Títol: Implementació Pràctica d’Algorismes Fonamentals

Autor: Marc Vinyals Pérez
Data: 15 de Juny de 2012
Director: Salvador Roura Ferret
Departament del director: Lenguatges i Sistemes Informàtics
Titulació: Enginyeria en Informàtica
Centre: Facultat d’Informàtica de Barcelona (FIB)
Universitat: Universitat Politècnica de Catalunya (UPC)
BarcelonaTech
# Contents

## 1 Introduction

1.1 Background and scope .............................................. 1  
1.2 Related work ...................................................... 1  
1.3 Procedures ......................................................... 2  
1.4 Notation ........................................................... 3  

## 2 Graphs

2.1 Topological sorting .................................................. 5  
   - Timus 1280 Topological Sorting .................................. 5  
2.2 Shortest paths ...................................................... 5  
   - UVa 1063 Marble Game\(^1\) ..................................... 5  
   - ICPC 5893 LatticeLand ........................................... 6  
   - Timus 1254 Die Hard ............................................. 6  
   - Timus 1487 Chinese Football ..................................... 7  
2.3 Maximum flow ...................................................... 7  
   - Timus 1277 Cops and thieves .................................... 7  
   - Timus 1533 Fat Hobbits .......................................... 8  
   - UVa 1049 Remember the A La Mode\(^2\) ........................ 8  
2.4 Maximum Matching .................................................. 9  
   - Timus 1099 Work Scheduling .................................... 10  
2.5 Dynamic programming ............................................. 10  
   - Timus 1169 Pairs ................................................ 10  

## 3 Data Structures

3.1 Hash Table .......................................................... 11  
   - UVa 12435 Consistent verdicts .................................. 11  
3.2 Search tree .......................................................... 11  
   - Timus 1613 For Fans of Statistics ............................... 11  
3.3 RMQ ................................................................. 12  
   - Timus 1846 GCD 2010 ............................................. 13  

## 4 Strings

4.1 Parsing ............................................................... 15  
   - Timus 1102 Strange Dialog ...................................... 15  

---

\(^1\)ICPC 3807: World Finals 2007  
\(^2\)ICPC 3562: World Finals 2006
## Contents

<table>
<thead>
<tr>
<th>Time</th>
<th>Problem</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>Like Comparisons</td>
<td>15</td>
</tr>
<tr>
<td>16</td>
<td>Trie</td>
<td>16</td>
</tr>
<tr>
<td>16</td>
<td>Japanese Sorting</td>
<td>16</td>
</tr>
<tr>
<td>16</td>
<td>Code generators</td>
<td>16</td>
</tr>
<tr>
<td>16</td>
<td>Stack Machine Programmer</td>
<td>16</td>
</tr>
<tr>
<td>17</td>
<td>Salary</td>
<td>17</td>
</tr>
<tr>
<td>17</td>
<td>Periodical Numbers</td>
<td>17</td>
</tr>
<tr>
<td>17</td>
<td>Hieroglyphs</td>
<td>17</td>
</tr>
</tbody>
</table>

### 5 Algebra

#### 5.1 Combinatorics
- Boxes | 19
- Ents | 19
- Warehouse Problem | 20
- Cutting a Painted Polygon | 20

#### 5.2 Number Theory
- Don’t Ask Woman about Her Age | 21
- Happy 10/9 Day | 21
- Sum of Sequential Numbers | 21
- Square Root | 21
- Square Country | 22
- Quad | 23
- Pairs of Integers | 23
- Bipartite Numbers | 23
- Film Rating | 24
- Intelligence Data | 24
- Binomial coefficients | 25
- Chase | 25

#### 5.3 Linear algebra
- Nikifor | 25
- Winnie the Pooh | 26
- Bookmakers | 26
- Net Loss | 27

#### 5.4 Linear programming
- Lead or Gold | 28

### 6 Calculus

#### 6.1 Numerical integration
- Suspense | 29

#### 6.2 Root finding

---

3ICPC 3564: World Finals 2006  
4ICPC 4124: World Finals 2008  
5ICPC 3001: World Finals 2004
## Contents

<table>
<thead>
<tr>
<th>Problem</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timus 1621</td>
<td>Definite Integral</td>
<td>30</td>
</tr>
<tr>
<td>ICPC 6027</td>
<td>Curvy Little Bottles</td>
<td>31</td>
</tr>
</tbody>
</table>

### 7 Optimization

#### 7.1 Greedy algorithms
- Timus 1113    | Jeep                                      | 33   |
- Timus 1131    | Copying                                   | 33   |
- Timus 1161    | Stripies                                  | 33   |
- Timus 1587    | Flying Pig                                 | 33   |
- UVa 1019      | Light Bulbs\(^6\)                         | 33   |
- Timus 1534    | Football in Gondor                        | 33   |
- Timus 1483    | Table Football                             | 33   |
- Timus 1535    | The Hobbit or There and Back Again         | 33   |

#### 7.2 Dynamic programming
- Timus 1167    | Bicolored Horses                           | 33   |
- Timus 1592    | Chinese Watches                           | 33   |

#### 7.3 Exhaustive search
- UVa 1047      | Zones\(^7\)                                | 36   |

### 8 Geometry

#### 8.1 Basic formulae
- Timus 1111    | Squares                                   | 39   |
- Timus 1168    | Radio Stations                            | 39   |
- Timus 1093    | Darts                                     | 40   |
- Timus 1151    | Radiobeacons                              | 40   |
- Timus 1482    | Triangle Game                             | 40   |
- UVa 1058      | Grand Prix\(^8\)                          | 41   |
- Timus 1550    | Dean’s Pyramid 3                          | 41   |

#### 8.2 Intersections
- UVa 1029      | Heliport\(^9\)                            | 42   |
- Timus 1451    | Beerhouse Tale                            | 42   |
- UVa 1081      | The Return of Carl\(^10\)                 | 42   |
- ICPC 5887     | Unchanged Picture                         | 42   |
- ICPC 5859     | Encircling Circles                        | 42   |
- ICPC 6035     | Shortest Flight Path                      | 42   |

#### 8.3 Basic constructions
- UVa 1029      | Heliport\(^9\)                            | 44   |
- Timus 1451    | Beerhouse Tale                            | 44   |
- UVa 1081      | The Return of Carl\(^10\)                 | 44   |
- ICPC 5887     | Unchanged Picture                         | 44   |
- ICPC 5859     | Encircling Circles                        | 44   |
- ICPC 6035     | Shortest Flight Path                      | 44   |

#### 8.4 Convex hull
- Timus 1185    | Wall                                      | 50   |
- Timus 1281    | River Basin                               | 50   |

\(^6\)ICPC 2722: World Finals 2003
\(^7\)ICPC 3278: World Finals 2005
\(^8\)ICPC 2377: World Finals 2007
\(^9\)ICPC 2994: World Finals 2004
\(^10\)ICPC 4447: World Finals 2009
## Contents

<table>
<thead>
<tr>
<th>Source</th>
<th>Problem Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timus 1538</td>
<td>Towers of Guard</td>
<td>51</td>
</tr>
<tr>
<td>UVa 1084</td>
<td>Deer-Proof Fence(^{11})</td>
<td>52</td>
</tr>
<tr>
<td>ICPC 5891</td>
<td>A Classic Myth: Flatland Superhero</td>
<td>52</td>
</tr>
<tr>
<td>8.5</td>
<td>Half-plane intersection</td>
<td>52</td>
</tr>
<tr>
<td>Timus 1681</td>
<td>Brother Bear’s Garden</td>
<td>53</td>
</tr>
<tr>
<td>Timus 1062</td>
<td>Triathlon</td>
<td>54</td>
</tr>
<tr>
<td>8.6</td>
<td>Arrangements</td>
<td>54</td>
</tr>
<tr>
<td>UVa 1065</td>
<td>Raising the Roof(^{12})</td>
<td>55</td>
</tr>
<tr>
<td>8.7</td>
<td>Sweeping algorithms</td>
<td>55</td>
</tr>
<tr>
<td>Timus 1103</td>
<td>Pencils and Circles</td>
<td>55</td>
</tr>
<tr>
<td>UVa 12426</td>
<td>Counting Triangles</td>
<td>56</td>
</tr>
<tr>
<td>Timus 1097</td>
<td>Square Country 2</td>
<td>56</td>
</tr>
<tr>
<td>Timus 1065</td>
<td>Frontier</td>
<td>57</td>
</tr>
<tr>
<td>Timus 1556</td>
<td>Multishot in the Secret Cow Level</td>
<td>57</td>
</tr>
<tr>
<td>UVa 1077</td>
<td>The Sky is the Limit(^{13})</td>
<td>58</td>
</tr>
<tr>
<td>8.8</td>
<td>Segment intersection</td>
<td>58</td>
</tr>
<tr>
<td>UVa 1035</td>
<td>Tree-Lined Streets(^{14})</td>
<td>59</td>
</tr>
<tr>
<td>8.9</td>
<td>Voronoi diagram</td>
<td>59</td>
</tr>
<tr>
<td>UVa 1039</td>
<td>Simplified GSM Network(^{15})</td>
<td>64</td>
</tr>
<tr>
<td>8.10</td>
<td>k-d tree</td>
<td>64</td>
</tr>
<tr>
<td>ICPC 5908</td>
<td>Tracking RFIDs</td>
<td>65</td>
</tr>
<tr>
<td>8.11</td>
<td>Optimization</td>
<td>65</td>
</tr>
<tr>
<td>Timus 1332</td>
<td>Genie Bomber</td>
<td>65</td>
</tr>
<tr>
<td>Timus 1075</td>
<td>Thread in a Space</td>
<td>65</td>
</tr>
<tr>
<td>Timus 1170</td>
<td>Desert</td>
<td>66</td>
</tr>
<tr>
<td>8.12</td>
<td>Binary and ternary search</td>
<td>66</td>
</tr>
<tr>
<td>UVa 1026</td>
<td>The Solar System(^{16})</td>
<td>67</td>
</tr>
<tr>
<td>Timus 1256</td>
<td>Cemetery Guard</td>
<td>67</td>
</tr>
<tr>
<td>Timus 1583</td>
<td>Cheese</td>
<td>68</td>
</tr>
<tr>
<td>UVa 1082</td>
<td>Conduit Packing(^{17})</td>
<td>68</td>
</tr>
<tr>
<td>8.13</td>
<td>Other algorithms</td>
<td>69</td>
</tr>
<tr>
<td>Timus 1768</td>
<td>Circular Strings</td>
<td>69</td>
</tr>
<tr>
<td>Timus 1444</td>
<td>Elephpotamus</td>
<td>69</td>
</tr>
<tr>
<td>Timus 1130</td>
<td>Nikifor’s Walk</td>
<td>70</td>
</tr>
</tbody>
</table>

### 9 Games

| 9.1 Min-max | 71 |

\(^{11}\)ICPC 4450: World Finals 2009  
\(^{12}\)ICPC 3809: World Finals 2007  
\(^{13}\)ICPC 4127: World Finals 2008  
\(^{14}\)ICPC 3000: World Finals 2004  
\(^{15}\)ICPC 3270: World Finals 2005  
\(^{16}\)ICPC 2729: World Finals 2003  
\(^{17}\)ICPC 4448: World Finals 2009
Contents

Timus 1163  Chapaev ................................................. 71

10  Trivial ................................................................. 73
  10.1 Simulations ....................................................... 73
  ICPC 5886  The Grille ............................................... 73
  ICPC 5880  Vigenère Cipher Encryption ........................... 73
  Timus 1083  Factorials!!! ......................................... 73
  10.2 Pranks ............................................................ 73
  Timus 1164  Fillword ................................................ 73
  10.3 Horrible IO ....................................................... 74
  Timus 1187  Statistical Trouble ................................... 74
  Timus 1274  Fractional Arithmetic ................................. 74

11  Economical analysis .................................................. 75

12  Conclusions ........................................................... 77
  12.1 Conclusions ...................................................... 77
  12.2 Future directions ................................................. 77
1 Introduction

1.1 Background and scope

The Technical University of Catalonia (UPC) started taking part in the ACM ICPC programming competition in 2002. Since then teams of UPC students have achieved some impressive results, qualifying for the World Finals in 7 out of 9 occasions.

These competitions require, as well as an amount of ingenuity, solid knowledge of a set of fundamental algorithms. These algorithms must be known not only in a black box way so that they can be simply used but in a white box way so that they can be adapted and an ad-hoc variation can be programmed for a particular problem.

The first goal of this project is to identify fundamental, non-trivial algorithms that may be useful to know in a competitive environment and to provide the tools for UPC contestants to understand them thoroughly. Precisely defining the objectives is therefore an objective of the project in itself.

At these competitions it is allowed to use a library of algorithms, chosen and prepared by the contestants, within a length restriction. It is therefore useful to have readily available code listings that implement the aforementioned algorithms. These implementations must be written in either C++ or Java, with a bias toward C++ for efficiency and practicality reasons, must be independent of external libraries and be correct. It is also desirable for the implementations to be efficient, easy to write, understandable and extensible, in this order.

The second objective of the project is to provide implementations for some algorithms that are not currently included in the library and that satisfy the above restrictions as closely as possible.

The library will contain a diverse set of algorithms spanning across many branches of computer science and mathematics, with non trivial implementations, and frequently used because of their importance.

1.2 Related work

Many universities that take part in the ICPC have their own training methods and materials. Because of the competitive nature of this event not all of the information is published, so each team must develop its own library.
1 Introduction

Some collaborative efforts exist, such as the public problem repositories and online judges described below or events like the Petrozavodsk training camp for teams in that region. It should be noted that Jutge.org is a complementary project developed within UPC.

This project does not start from scratch but builds up on a solid base of algorithms studied by previous UPC teams, and will likely continue to be used during the following years.

The part of the project described in this report corresponds to the current year and was a joint effort of the CFIS students Pol Mauri, Félix Miravé and Marc Vinyals, directed by the LSI professor Salvador Roura. The problem solutions and analysis exposed in this report are, however, the sole work of the author unless it is otherwise stated. Some of the tested code was originally written by Alex Álvarez, Javier Gómez, Josep Àngel Herrero, Ricardo Martin and possibly other students.

1.3 Procedures

The procedure used for identifying the set of interesting algorithms was to select a wide enough sample of past problems used in contests, attempt to solve the problems and notice both repeating techniques and gaps in our library.

The procedure used for testing the correctness and efficiency of the problems was to submit the solutions to online judge services such as the University of Valladolid Online Judge (UVa), the International Collegiate Programming Contest Live Archive (ICPC) and the Ural Federal University Archive (Timus).

In addition, to test the correctness and efficiency of the new implementations added to the library, an extensive set of test cases was developed and used locally. Correctness could then be determined by manually inspecting the output, comparing it to the output of a different, easier to program (but usually inefficient) program and ensuring that in-program assertions held.

The following sections will contain a brief analysis of most of the problems tested, notes on the algorithms used to solve them and a more detailed explanation of newly implemented algorithms.

With the objective of providing examples to algorithms, problems are grouped by the key algorithm used in its resolution. This means that problems that involve a reduction or more than one algorithm may not be in the obvious location a priori.

Original problem statements are usually long and involve real-life situations. The process of correctly interpreting them, although non-trivial, is not of much interest and is left beyond the scope of this project. For the sake of simplicity, problem descriptions will be as brief as possible, even if this means introducing some mathematical notation not present in the original statement which should be considered as part of the resolution procedure.
1.4 Notation

The goal of this report is to justify why the selected implementations did work and provide some insight deeper than what a mere description of an algorithm would. Code listings are not provided, but an experienced programmer should be able to reproduce them as most of the implementation quirks are explicitly stated.

1.4 Notation

- $[a..b]$ denotes the closed interval $[a, b] \subset \mathbb{Z}$.
- $[a] = [1..a]$.
- $\mathbb{N} = \mathbb{Z}^+ \cup \{0\}$.
- $n$ is usually the size of the problem.
- $o, O$ and $\Theta$ classes are abused as is usual in big-oh notation.
- $\subset$ inclusions are not proper. $\subseteq$ inclusions are.
- $G(V, E)$ is a graph with nodes $V$ and edges $E$. Either $|V| = n$ and $|E| = m$ or $|V| = v$ and $|E| = e$.
- $\langle v_1, v_2 \rangle$ is an edge from $v_1$ to $v_2$.
- $\Sigma$ is an alphabet.
- $\chi_S$ is the characteristic function of the set $S \subset \mathbb{R}$.
- if $p$ and $q$ are points, $pq$ is the vector from $p$ to $q$.
- if $u$ and $v$ are vectors, $\langle u, v \rangle$ and $\|u\|$ are the euclidean scalar product and norm, respectively.
2 Graphs

Many problems can be modelled using a graph; this includes road networks but also more abstract relations among elements. This is a well known area and complete descriptions of the briefly mentioned algorithms can be found in [5].

2.1 Topological sorting

A topological sorting of the nodes of a DAG is an order where a node always comes after its ancestor. The standard algorithm greedily selects any node without parents as the next element in the order and conceptually erases it from the graph. In practise, a parent count is maintained.

Timus 1280 Topological Sorting

Statement Given a graph $G$ with $n$ nodes and $m$ edges and a order of the nodes, find whether the order is topologically consistent.

Solution Slight variation of the standard topological sorting algorithm. Replace the node selection part for a check of the number of parents of a node.

2.2 Shortest paths

Breadth first search The breadth first search algorithm explores a graph by visiting at each step the set $V_t$ of nodes at distance $t$ from an source $s$. $V_0 = \{s\}$ and $V_{t+1} = n(V_t) - V_{t-1}$. Nodes are visited in distance to $s$ order. Its cost is $O(e)$.

UVa 1063 Marble Game$^1$

Statement A $n \times n$ board with $m$ marbles, $m$ holes and $w$ walls between cells is given. A move is started by moving all the marbles not in a hole toward the same direction. A marble stops when it falls into an empty hole or hits the board limits, a wall or a still marble. The winning position is that where each marble is in its matching hole. Find the minimum amount of moves needed to reach the winning position.

$^1$ICPC 3807: World Finals 2007
2 Graphs

Solution  Encode the state as whether each marble is not in its hole and, if so, its position. Run a breadth first search on the implicit graph of states, finding transitions on the fly by simulating the conditions defined by the problem statement and pruning impossible branches where a marble falls into a non-matching hole.

The graph of possible states has a size of \((n^2)^m\), but most of them are not reachable from the initial position. Even though computing the whole graph is not feasible, computing the reachable set is.

Caveats  Some marbles may already be inside a hole at the initial position.

Notes  Félix Miravé and Salvador Roura collaborated in solving this problem.

ICPC 5893 LatticeLand

Statement  A \(w \times h\) lattice and \(f\) segments whose endpoints lie in the lattice are given. A move consists of choosing an integer acceleration vector with modulo \(\leq 1\), adding the acceleration to the velocity and adding the velocity to the position. A move would leave the lattice or intersect a segment is invalid. Find the minimum amount of valid moves needed to move between two given points with 0 starting and ending speed.

Solution  Encode the state as a position and a velocity. Note that the speed in \(x\) can be negative and its absolute value must satisfy \(1 + 2 + \cdots + v_x + \cdots + 2 + 1 \leq w\), this is \(v^2 \leq w\). So the state is an element of \([w] \times [h] \times [2\sqrt{w} + 1] \times [2\sqrt{h} + 1]\). Run a breadth first search on the implicit graph of states, finding transitions on the fly and checking their validity.

Note that there are \(n = O((wh)^{3/2})\) nodes and \(m = O(4n) = O(n)\) edges, each of which takes \(O(f)\) time to check for validity. This gives a worst case runtime of \(O((wh)^{3/2}f)\).

Dijkstra  Dijkstra’s algorithm explores a non-negatively weighted graph in increasing order of costs by greedily visiting the least cost node reachable from the set of already visited nodes. Using a heap for the queue of nodes to visit yields an implementation with a cost of \(O(e \log e)\).

Timus 1254 Die Hard

Statement  A \(n \times m\) grid with valid cells is given. It is allowed to move to the 8 adjacent cells at speed \(v\). Distances between cells are the euclidean distance of their centres. A starting cell and a list of cells to visit is given. Find the minimum time to visit the cells in the given order, skipping unreachable cells.
2.3 Maximum flow

**Solution** For each cell in the list find the minimum distance from the current cell using Dijkstra’s algorithm on the implicit graph. If the cell is reachable, add the distance to the total and update the current cell. Return $d/l$. The cost of the algorithm is $O(ke \log e)$, where $e < 8mn$ is the number of edges. The $k$ factor can be reduced to $\min(k, mn)$ by reusing computations starting from the same node.

**Floyd-Warshall** Floyd-Warshall’s algorithm computes the all-pair shortest paths by observing that $d(i,j) = \min_k d(i,k) + d(k,j)$. Its cost is $O(v^3)$.

**Timus 1487 Chinese Football**

**Statement** The adjacency matrix of a DAG with $n$ nodes is given. $q$ pairs of nodes are given. For each pair, find whether a common ancestor exists.

**Solution** A simple solution is to compute the set of ancestors of all the nodes using Floyd’s algorithm on the transposed graph. The answer to each query is whether the intersection of the ancestor sets is empty. Use bitsets to represent sets.

**Caveats** The original statement of this problem is plain wrong. It asks to find whether the first node is an ancestor of the second, which is a different problem.

2.3 Maximum flow

The Ford-Fulkerson algorithm computes the maximum flow from a source $s$ to a sink $t$ by repeatedly finding an augmentative path, considering the maximum flow through that path and updating the graph with the residual network. The Edmonds-Karp implementation uses a BFS step for finding an augmentative path. Dinic’s variation of the algorithm, which we are using, improves the running time by considering many paths at once.

**Timus 1277 Cops and thieves**

**Statement** Given a graph $G$ with $n$ nodes, weights $w_i$ of the nodes, a list of edges, and nodes $s$ and $t$, find whether it is possible to choose a subset $S \subset V - \{s,t\}$ with cost $\leq k$ such that $s$ and $t$ are disconnected in $G - V$.

**Solution** Consider the standard reduction to introduce nodes with capacities in a max-flow: replace each node $x$ with two nodes $x_i, x_o$, connected with a directed edge $\langle x_i, x_o \rangle$ with capacity $w_x$. Replace edges $\langle x, y \rangle$ for $\langle x_o, y_i \rangle$. The problem is reduced to finding if a cut of cost $\leq k$ from $s_o$ to $t_i$ exists. Compute the maximum flow, which is equal to the minimum cut, and compare it to $k$. 
Caveats \( S \) is not \( V - (n(s) \cup n(t)) \), as the original statement may suggest.

**Timus 1533 Fat Hobbits**

**Statement** Consider a set \( X \) with \( n \) elements. The adjacency matrix of the graph of a partial order of \( X \) is given. Find a maximum subset \( S \subset X \) such that no element in \( S \) is bigger than another.

**Solution** This is the maximal antichain problem in a DAG. By Dilworth’s theorem [16], the problem is the dual of the minimal path cover problem and can be reduced to the min flow problem by assigning a minimum capacity of 0 to edges and 1 to nodes and considering the crossing edges in the residual graph cut. The minimum flow problem can itself be reduced to maximum flow by changing minimal capacities into negated maximum capacities and finding an initial feasible negative flow.

The cost of this approach is \( O(ve) = O(n^3) \).

**Caveats** Set default capacities for the maximum flow algorithm to \(-\infty\), not 0.

**Notes** Elisabeth Rodriguez collaborated in solving this problem.

**Minimum cost flow** Letting the path finding step in the Ford-Fulkerson algorithm be Dijkstra’s algorithm yields a flow with a minimum cost per unit of flow among all the maximum flows. In order to ensure non-negative edge costs, potentials are added as in Johnson’s algorithm.

**UVa 1049 Remember the A La Mode**

**Statement** Consider \( a_i \) nodes of type \( A_1 \ldots A_m \) and \( b_j \) nodes of type \( B_1 \ldots B_n \). The cost \( c_{ij} \) of matching two nodes of types \( A_i \) and \( B_j \) is given for all matchable types \( i, j \). \( \sum_i a_i = \sum_j b_j \). Find the minimum and maximum cost among maximum bipartite matchings.

**Solution** Reduce the problem to the maximum flow in minimum cost problem. Build a graph with nodes \( s, A_i, B_j \) and \( t \). All edges \( \langle s, A_i \rangle \) exist and have capacity \( a_i \) and cost 0. All edges \( \langle B_j, t \rangle \) exist and have capacity \( b_j \) and cost 0. Edges \( \langle A_i, B_j \rangle \) exist if the types are matchable and have infinite capacity and cost \( c_{ij} \). The minimum cost is the minimum cost of a maximum flow from \( s \) to \( t \). Conversely, the minimum cost of a maximum flow on the graph with the costs set to \(-c_{ij} \) gives minus the maximum cost.

---

1ICPC 3562: World Finals 2006
2.4 Maximum Matching

Given a graph \( G(V, E) \) a matching is a subgraph \( M \subset G \) with maximum degree 1. This is, each node is adjacent to at most another. A matching is maximum if \( |M| \) is maximum among all the possible matchings of \( G \) [3, 11].

Given a graph \( G \) and a matching \( M \subset G \), an alternating path for \( M \) is a path \( P = v_1 \ldots v_K \) such that \( \langle v_{2k}, v_{2k+1} \rangle \in M \) and \( \langle v_{2k-1}, v_{2k} \rangle \notin M \). This is a path whose edges alternate between edges in and out of the matching.

Given a graph \( G \) and a matching \( M \subset G \), an augmenting path for \( M \) is an alternating path \( P = v_1 \ldots v_{2K} \) such that \( v_1, v_{2K} \notin M \). This is an alternating path of odd size whose endpoints are not in the matching.

Given a graph \( G \) and a matching \( M \subset G \), a blossom is an alternating odd cycle. The only vertex not in \( M \) is the base of the blossom.

It is easy to see that \( M \oplus P \) is a matching of size \( |M| + 1 \). Furthermore, Berge’s theorem proves that a matching is maximum iff no augmentative path exists. This suggests an algorithm: start with any matching \( M \). While an augmenting path exists, set \( M = M \oplus P \). Return \( M \).

Unfortunately finding an augmenting path is not always easy. As we can handle connected components separately we will work in a connected graph for simplicity. Start from any unmatched node and run a breadth first search, bicolouring nodes and allowing only edges in \( G - M \) from odd nodes and edges in \( M \) from even nodes. If the colouring is consistent and an augmenting path exists, this algorithm will find it.

If the colouring is not consistent, then we will have found an odd cycle. Moreover, this cycle will be a blossom. Set \( G' \) as \( G \) with the blossom contracted. By the blossom shrinking lemma, an augmenting path exists in \( G \) iff an augmented path exists in \( G' \), so we can recursively find and possibly contract all the blossoms. In this case the graph would be bipartite and the bicolouring algorithm would work.

It is possible to construct an augmenting path in \( G \) from the path in \( G' \). Say the blossom is \( w_1 \ldots w_k \), where \( w_1 \) is the base, and it was contracted to \( w \). If it does not touch the blossom, leave it unmodified. Otherwise the augmenting path is either \( wv_2 \ldots v_K \) or \( v_1 \ldots w_iw_{i+2} \ldots w_K \).

In the first case choose a node \( w_j \) such that \( \langle w_j, v_2 \rangle \in E \), then extend the augmenting path to \( w_1 \). Note that one out of the two possible alternating paths has the right parity and that as \( w \notin M' \), \( w_1 \notin M \) so \( w_1 \) can be a starting point for an augmenting path.

In the second case exactly one of \( \langle v_i, w \rangle \) or \( \langle w, v_{i+2} \rangle \) is in \( M' \), say \( \langle v_i, w \rangle \). But the only node that can be matched to a node outside the blossom is the base \( w_1 \). Choose another node such that \( \langle w_j, v_{i+2} \rangle \in E \). Replace \( w \) by \( v_1 \), the alternating path within the blossom of the right parity and \( v_j \).

By starting with any matching such as the empty one and successively finding augmenting paths and combining them with the matching the cardinality of the matching will
strictly increase until the maximum is reached. The size of the matching is $O(v)$, at most $O(v)$ blossoms exist, and each bicolouring takes $O(e)$ time. An upper bound for the algorithm runtime is $O(v^2e)$.

Note that the blossom shrinking lemma is an existence result. If an augmenting path is actually found, then it may be necessary to shrink all the blossoms again. Improvements to Edmonds’ algorithm exist that avoid these repeated calculations. One such example is Vazirani’s algorithm.

An heuristic that significantly reduces the average runtime is to find a feasible matching using a greedy algorithm (try to match nodes with fewer unmatched neighbours first) and then find augmenting paths.

**Timus 1099 Work Scheduling**

**Statement** Given a graph $G$ with $n$ nodes and a list of edges $(i, j)$, output a matching of $G$ with maximum cardinality.

**Solution** Straightforward maximum matching.

**2.5 Dynamic programming**

**Timus 1169 Pairs**

**Statement** Find a connected undirected graph with $n$ nodes and $k$ pairs of 1—connected nodes.

**Solution** Assume $G$ is a solution, and consider its 2—connected components. Node pairs in different 2ccs add 1 to the number of 1—connected pairs, while pairs in the same 2cc add nothing. The number of pairs depends only on the size of 2ccs and is $\sum_{i<j} |C_i| \cdot |C_j|$.

Thus, a partition of the nodes in 2ccs can be found using the recurrence

$$s(n, k) = \begin{cases} [n - i_0, n] \cup s(n - i, k - i(n - i)) & \exists i_0 = \min_{i \neq 2} s(n - i, k - i(n - i)) \neq \emptyset \\ \emptyset & \text{otherwise} \end{cases}$$

Finally, map each 2cc to a cycle and join consecutive cycles with a single edge.

**Caveats** Note that the graph of size 1 is 2-connected and no graph of size 2 (the 2-path and the 2-cycle) is 2-connected.
3 Data Structures

3.1 Hash Table

A hash table stores <key, value> elements. Elements can be inserted, erased and accessed by key in $O(1)$ time. The STL provides an implementation.

UVa 12435 Consistent verdicts

Statement Given a fixed sequence of $n$ points $p_i \in \mathbb{Z}^2$ and a variable radius $r$, define $x_i(r)$ as the number of points at distance from $p_i$ not greater than $r$. Find how many different sequences exist.

Solution Let $r$ range in $[0, \infty)$. For $r = 0$, $x_i(0) = \{0\}_n$. When $r$ equals the distance between some points $p_i$ and $p_j$, $x_i$ and $x_j$ are increased, this is the only case where some $x_i$ increases, and $x_i$ never decreases. If more than one pair of points share the same distance, all increases happen at the same time. The number of different sequences is, thus, the number of different distances plus one. An algorithm that counts the cardinality of the set of squared distances can run in $O(n)^2$ using a hash set.

Caveats The STL standard hash set is apparently too slow; use a custom hash of size $N = n^2 + 1$ and function $x \mod N$.

3.2 Search tree

A binary search tree stores <key, value> elements sorted by key. Elements can be inserted, erased and accessed by key in $O(\log n)$ time. The first element can be accessed in $O(1)$ time. The elements can be traversed in order in $O(n)$ time. The STL provides a red-black tree implementation that is enough for most purposes; if custom capabilities such as order numbers are needed then the more simple AA tree implementation can be used.

Timus 1613 For Fans of Statistics

Statement A sequence $\{x_i\}$ of $n$ numbers is given. Answer $q$ queries of the type $x \in [a, b]?$, with $[a, b] \subset [n]$. 
3 Data Structures

Solution  Maintain a dictionary from numbers to indexes in a vector of trees. For each number, maintain a search tree \( S(x) \) with the positions it appears in. The cost is \( O(\log n) \) per insertion. For each query, find \( S(x) \) and \( z = \min\{i \in S(x), i \geq a\} \) in \( O(\log n) \). The answer to the query is whether \( z \leq b \).

Caveats  A RMQ structure that counts the number of appearances of each number at each interval is too slow.

3.3 RMQ

A Range Minimum Query structure, also called an interval tree, is a data structure that provides fast updates and queries to a function defined over a list of numbers as long as the function is easily computable from two sublists. An example of such a function is \( \min \), and hence the name of the structure.

A RMQ structure for a list up to size \( n \) has \( \lceil \log_2 n \rceil \) levels, where the the \( k \)-th level has \( n/2^k \) entries, each storing the value of \( f \) over an interval of the list of size \( 2^k \). The total size of the structure is thus \( \Theta(n) \).

Let \( f^k(i) \) denote the value of \( f \) over the interval \([x_i \ldots x_i + 2^k]\). We will require \( i \equiv 0 \mod 2^k \). The notation \( f \) will be abused to denote both \( f \) and the merge step that computes \( f \) on an interval from its values in two subintervals.

To update a value in the list, and this includes inserting or deleting it, first update the value at level 0. So when updating \( x_i \), set \( f^0(i) = f(x_i) \). Then propagate the value upwards in the structure. Let \( r(i, k) = i - (i \mod 2^k) = (i >> k) << k \). Compute \( f^{k+1}(r(i, k+1)) = f(f^k(r(i, k+1), f^k(r(i, k+1)+2^k)) \). \( \Theta(\log n) \) merge steps are needed.

To query the value of \( f \) over an interval, break it recursively into smaller intervals while traversing the structure upwards. The answer to a query for the interval \([i, j)\) is \( q^0(i, j) \), where

\[
q^k(i, j) = \begin{cases} f(f^k(i), p^k(i + 2^k, j)) & i \not\equiv 0 \mod 2^k \text{ and } i + 2^k < j \\ p^k(i, j) & \text{otherwise} \end{cases}
\]

\[
p^k(i, j) = \begin{cases} f(f^k(j - 2^k), q^{k+1}(i, j - 2^k)) & j \not\equiv 0 \mod 2^k \text{ and } i < j - 2^k \\ q^{k+1}(i, j) & \text{otherwise} \end{cases}
\]

and \( p^k(x, x) = q^k(x, x) = f(\emptyset) \). Again \( \Theta(\log n) \) merge steps are needed.

In the \( f = \min \) case, this means setting \( f^0(i) = x_i \) and letting the merge step be taking the minimum of two values.
Timus 1846 GCD 2010

Statement A sequence of insertions and deletions of integers to a set is given. At each step, find the gcd of the integers in the set.

Solution Maintain a dictionary from numbers to positions in the sequence. Maintain a RMQ structure where the function computed for each interval is the gcd of the numbers in this interval. Initially the interval is set to zero. On each insertion, replace a zero by the number and store the position in the interval in a hash map. On each deletion, query the dictionary for an appearance of the number in the interval and replace it by a zero.

Caveats Factoring and maintaining sets of prime factors is too slow.
4 Strings

Problems on strings are related to matching, prefixes, suffixes, grammars and automata.

4.1 Parsing

Timus 1102 Strange Dialog

Statement  Consider the grammar $S \rightarrow (A^*B^*)^*, A \rightarrow (\text{out} | \text{output} | \text{puton})^*, B \rightarrow (\text{in} | \text{input} | \text{one})^*$. Given $n$ strings, decide if they are in the language the grammar produces.

Solution  The grammar is equivalent to $S \rightarrow (\text{out} | \text{output} | \text{puton} | \text{in} | \text{input} | \text{one})^*$, which is equivalent to $S \rightarrow (\text{out} \lambda | \text{put} \lambda | \text{on} \lambda | \text{e}) | \text{in} \lambda | \text{put} \lambda | \text{on} \lambda | \text{e}) | \text{puton} | \text{one})^*$ which can be solved greedily because prefixes are different.

Timus 1177 Like Comparisons

Statement  A restricted regular expression syntax is defined. Given $n$ string and regular expression pairs of length $m$, find whether the strings match their corresponding regular expressions.

The syntax is as follows: $\%$ matches any number of any characters. _ matches a single character. A group $[S]$ matches any character in the set $S$. A group $[^S]$ matches any character not in the set $S$. A triple $c_1 - c_2$ in a group represents the interval $[c_1, c_2]$. Other characters in a group represent themselves (including $\^$, $[|)$, $\-$, _ and $\%$). Group delimiters are paired greedily. Special sequences don’t share characters. Strings are delimited by ’ characters; a ’ inside a string is represented as ”.

Solution  Deciding the expressible languages can be done with a dynamic programming approach, where the state is a pair of indexes $(i, j)$, $i$ for the tested word and $j$ for the regexp element, where an element is either a multi-length matcher or a set of characters. If the element at $j$ is a multi-length matcher, then try to match all the possible lengths: the recurrence is $\bigvee_{k=1}^m f(k, j + 1)$. Otherwise, the recurrence is $s_i \in S_j \lor f(i + 1, j + 1)$. The complexity of this approach is $O(m^3)$. 

15
Parsing the regular expression is tricky but can be done with an implicit automaton
with states group and nogroup. No backtracking is needed provided that it is allowed
to look ahead to check whether a [ will be matched and needs to be treated as a group
delimiter or a regular character. The complexity of this part is $O(m)$.

Caveats  Make sure that chars are unsigned. Note that it is possible for the DP state
to be $(m, j)$.

4.2 Trie

A trie is $|\Sigma|$-ary tree. Each node corresponds to a character, and words are stored as
paths from the root to a terminal node.

Timus 1546  Japanese Sorting

Statement  An order between strings is defined\(^1\) as a variation on the lexicographically
smaller order, where consecutive digits are considered as a whole number. Numbers pre-
cede characters. Ties are broken by amounts of leading zeros, lexicographically greater.
Some strings with a total of $m$ characters are given. Sort them by this order.

Solution  Insert the strings into a variation of a trie. In addition to standard nodes,
this data structure can contain nodes indicating that the next $n$ nodes are digits.
So, when parsing a string, if a digit is found first count the number of consecutive zeros
and then the number of consecutive digits. Add the number appropriately and insert
the position and number of leading zeros into a temporary vector. Add this vector to
the leaf node.
When traversing the trie follow the numerical branches first, ordered by number of digits,
and character branches afterwards. Maintain a stack of traversed characters. At leaves
sort the zeros vectors and print the strings, reconstructing skipped zeros.

4.3 Code generators

ICPC 5883  Stack Machine Programmer

Statement  Given two lists, $x$ and $y$, produce a program in a stack machine assembly
language that on input $x_i$ produces output $y_i$.

\(^1\)It is not defined in the original problem and must be deduced by generating sequences and comparing
them with a black box algorithm.
4.4 Other algorithms

Solution  Consider the polynomial \( p_i = \prod_{j \neq i} (x - x_j) \) and \( q_i = p_i(x) / p_i(x_i) \cdot y_i \). \( q_i(x) = y_i \) and \( q_i(x_j) = 0 \). So \( f(x) = \sum_i q_i(x) \) is a function that produces the desired output. Output a program that evaluates it on the given input; this is the sum of assembly language implementations of Horner’s algorithm hardcoded to evaluate each of the polynomials \( q_i \).


4.4 Other algorithms

Timus 1123  Salary

Statement  Given a number \( x \) of \( n \) digits, compute the smallest number \( y \geq x \) that is a palindrome.

Solution  Copy the reversed left half of \( x \) to the right half of \( x \). If the resulting number is smaller than \( x \), sum 1 to the left half and copy it again.

Timus 1558  Periodical Numbers

Statement  Two infinite periodic strings \((s_1)^*\) and \((s_2)^*\) are given. Find the (minimum) period and (minimum) non-periodic part of the arithmetic sum of the strings.

Solution  Let \( n = |s_1| \), \( m = |s_2| \). The maximum period length is bounded by \( \text{lcm}(n, m) \leq nm \), and the maximum non-periodic part length is bounded by \( \text{lcm}(n, m) + 1 \), the 1 accounting for a possibly different first carry. Generate \( 3nm + 1 \) digits of the sum and find the minimum period by exhaustively searching all the possible lengths and starts. This brute force approach has a cost of \( O((nm)^3) \).

Timus 1545  Hieroglyphs

Statement  A list of \( N \) 2-character words is given. Print all the words that begin with a given character.

Solution  Store whether a word appears in the list in an matrix of size \( |\Sigma|^2 \). Print the words that appear in the given row.
5 Algebra

This chapter includes some parts of mathematics that are generally classified as algebra: combinatorics, number theory, linear algebra and linear programming.

5.1 Combinatorics

There are no specific algorithms for combinatorics problems at this level. Most of the problems involve clever approaches to counting problems, and these require ad-hoc algorithms.

Timus 1114 Boxes

**Statement**  Count how many different ways are there to put at most $A$ red balls and $B$ blue balls into $N$ boxes.

**Solution**  Standard dynamic programming. The recurrence is

$$f(A, B, N) = \sum_{i=0}^{A} \sum_{j=0}^{B} f(A - i, B - j, N - 1)$$

Timus 1537 Ents

**Statement**  A multiset of numbers is defined as follows: $X^{(1)} = \{2\}$. $X^{(i)} = \bigcup_{x \in X^{i-1}} \{x + 1, 2x\}$. $X = \bigcup_{i \in \mathbb{Z}^+} X^{(i)}$. Find $|a \in X| \mod k$.

**Solution**  Standard dynamic programming. Define $e(a) = |a \in X| \mod k$. The recurrence is

$$e(a) = \begin{cases} 
  e(a - 1) + e(a/2) & a \equiv 0 \mod 2 \\
  e(a - 1) & a \equiv 1 \mod 2
\end{cases}$$
5. Algebra

**Timus 1107 Warehouse Problem**

**Statement**  Two distinct multisets of elements in \([n]\) are defined to be similar if one can be obtained by deleting an element of the other or swapping an element of the other for one in \([n]\). Given \(k\) sets and \(m > n\), find a mapping \(S_i \mapsto j, j \in [m]\) such that if \(f(S_i) = f(S_j)\), then \(S_i\) is not similar to \(S_j\).

**Solution**  Map \(S_i\) to \(\sum_{x \in S_i} x \mod n + 1\). Assume a missing element is an instance of a new element 0, so all sets can be considered to be of the same size and the similarity condition is to swap an element (maybe with 0). Since all elements of 0..\(n\) belong to different classes modulo \(n + 1\), a single swap will always send a set to a different mapping.

**Timus 1181 Cutting a Painted Polygon**

**Statement**  The \(n\) vertexes of a convex polygon are painted in red, green and blue. All colours are present at least once and consecutive vertexes have different colours. Find a triangulation whose triangles have vertexes of different colours, if it exists.

**Solution**  Let us show by induction that the triangulation exists and how to construct it. The adjacency condition will be satisfied by always using edges with different coloured endpoints. The hard condition will be to keep vertexes of all 3 colours. For \(n = 3\), the polygon is a 3 coloured triangle. Otherwise, consider any two consecutive vertexes, coloured \(a\) and \(b\).

If no other \(a\) coloured vertex exists, then the other vertexes are painted alternating \(b\) and \(c\), so a triangulation with all of its edges starting from \(a\) satisfies the condition. Same if no other \(b\) vertex exists.

If other vertexes of colours \(a\) and \(b\) exist, consider the adjacent vertexes to \(a\) and \(b\). If either of them is \(c\), removing the triangle \(abc\) leaves us with a polygon of \(n - 1\) vertexes with 3 different colours, which can be triangulated by the induction hypothesis.

If no adjacent vertex is \(c\), then they are \(b\) and \(a\) in this order. By hypothesis, a vertex of colour \(c\) exists. Removing the triangle \(abc\) leaves us with two polygons. Clearly they have a \(c\) vertex. In the first polygon, the vertex adjacent to the \(a\) vertex is coloured \(b\), same with the second. By induction hypothesis they both can be triangulated.

5.2 Number Theory

Number theory includes both classic arithmetic and modern number theory.
Timus 1104 Don’t Ask Woman about Her Age

Statement Given a number’s $n$-digit representation $m(k)$ in base $k$, find the minimum $k$ such that $k - 1 \mid m$. It is known that $k \leq K$.

Solution Evaluate the polynomial $\sum a_i k^i$ for all $k > \max a_i$ and check if $k - 1$ divides it. Using Horner’s rule and taking modulo $k - 1$ after each step has a $\Theta(nK)$ cost.

Solution $k \equiv 1 \mod k - 1$. Thus, $\sum a_i k^i \equiv \sum a_i \mod k - 1$. By evaluating the sum only once, the cost is reduced to $\Theta(n + K)$.

UVa 12431 Happy 10/9 Day

Statement Compute $d\ldots d_n(b)$ mod $m$.

Solution $d\ldots d_n(b) = d \sum_{i=0}^{n-1} b^i = d(b^n - 1)/(b - 1)$. Since $(b - 1)$ may not be invertible modulo $m$, compute instead $b^n - 1$ modulo $m(b-1)$ and divide by $b-1$. $b^n \mod m(b-1)$ can be computed in $O(\log n)$.

Caveats $m(b-1)$ can be up to $10^{14}$, so the exponentiation function will need 128-bit integers to compute $x^2 \mod m(b-1)$.

Timus 1120 Sum of Sequential Numbers

Statement Given $n$, find $a > 0$ and the maximum $p > 0$ such that $n = \sum_{i=0}^{p-1} a + i$.

Solution $\sum_{i=0}^{p-1} a + i = ap + p(p - 1)/2$. So $a = (n - p(p - 1)/2)/p$ is a monotonically decreasing function of $p$. Iterating through all $p$ and checking whether $a$ is a positive integer gives an $O(\sqrt{n})$ algorithm.

Timus 1132 Square Root

Statement Compute the square root of $k$ numbers $a_i \in \mathbb{Z}_{p_i}^*$, with $p_i$ prime.
5 Algebra

Solution  Let \( g \) be a generator of \( \mathbb{Z}_p^* \). If \( a = g^r \), then \( a^{1/2} = \pm g^{r/2} \) if \( 2 \mid r \) and no solution exists if \( 2 \nmid r \).

There are \( \varphi(p-1) \) generators of \( \mathbb{Z}_p^* \), which is \( \Omega(\sqrt{p}) \), where the worst case (fewer generators) happens when \( (p-1) \) is a product of two primes (a RSA number).

The number of tries until an element of \( K \subset [n] \) is found by picking elements of \([n]\) at random without replacement is modelled by a random variable that follows a negative hypergeometric distribution of parameters \( 1, k, n \) [17]. Its expected value is

\[
E(n, k) = \frac{1}{\frac{k+1}{n+1}} = \frac{n + 1}{k + 1}
\]

Thus the expected number of tries until a generator is found is \( O(\sqrt{p}) \). To test whether a number is a generator, check that \( g^{(p-1)/q} \neq 1 \) for all primes \( q \) such that \( q \mid p - 1 \). For each \( q \) we need \( O(\log p) \) operations, and there are \( O(\log p) \) such \( q \)s. Thus the expected time to find a generator is \( O(\sqrt{p} \log^2 p) \).

The discrete logarithm problem is believed to be hard (it is a fundamental part of the Diffie-Hellman key-exchange algorithm). There is a \( O(\sqrt{p} \log p) \) algorithm ([10], discrete logarithm). Assume \( x = g^r \). Dividing \( r \) by \( s \) we get \( r = as + b \), so \( b = r - as \). \( g^b = x g^{-as} \). \( a \) and \( b \) can be found by generating the two lists \( \{g^b\}_b, \{g^{-as}\}_a \), sorting them and checking for a common element in time \( \Theta(s \log s + p/s \log p/s) \). Setting \( s = \sqrt{p} \) gives a runtime of \( \sqrt{p} \log p \), which is still exponential with respect to the number of bits but fast enough for this problem.

Solution  An algorithm for finding square roots in \( \Theta(\log p) \) exists ([9], square root modulo \( p \)) but was not tested.

Notes  Daniel Remon, Salvador Roura and Pol Mauri collaborated in solving this problem.

Timus 1073 Square Country

Statement  Find the minimum number of squares whose sum is \( n \).

Solution  By Lagrange’s four square theorem the solution is less or equal than 4. Pre-compute the solutions for \( f(n) = 1, 2 \) and use a dynamic programming approach with the recurrence \( f(n) = \min_{i \leq n} 1 + f(n - i^2) \).

Caveats  A straightforward solution may overflow the recursion stack.
5.2 Number Theory

UVa 12443 Quad

Statement  Compute how many real numbers below \( n \) are the square root of the norm of a prime Lipschitz integer.

Solution  The norm of a Lipschitz integer is always an integer. By multiplicativity of the norm, if a Lipschitz integer has prime norm it is prime. If it has composite norm, then choose a factorization of the norm. By the unique factorization for primitive Hurwitzian quaternions theorem\(^4\), a factorization into Hurwitz quaternions exists so that their norms are the chosen factors. By unit translation, this factorization induces a factorization into Lipschitz integers. So, a prime Lipschitz integer has prime norm.

Note that this implication does not hold for the Gauss integers: if a Gauss integer has prime norm it is prime, but Gauss primes with composite norm exist.

The problem has been reduced to computing \( \pi(n^2) \).

Since many queries are given, a good algorithm is to compute the list of primes up to \( \max n^2 \) using the Eratosthenes sieve and return the position of \( n^2 \) in this list with a binary search.

Caveats  A straightforward sieve is too slow. Instead, the given solution computes a partial sieve using the first \( I \) primes, replicates the chunk of size \( \prod_{i \leq I} p_i \), and then starts the sieve from \( p_{I+1} \). \( I \) should be chosen so that the chunk has a size of about \( n \). Another optimization is to only consider numbers congruent to \( \pm 1 \mod 6 \).

Timus 1189 Pairs of Integers

Statement  Given an integer \( n \), compute the pairs \( \langle a, b \rangle \) such that \( a + b = n \) and \( b \) is obtained by removing a digit from \( a \).

Solution  Write \( a = 10^z x + y, b = 10^z \lfloor x/10 \rfloor + y \). Then \( 2y \equiv n \mod 10^z \) and \( x = \lceil (n - 2y)/10^z \cdot 10/11 \rceil \). Iterating through \( z \) and the two possible solutions for \( y \) between 0 and \( 10^z (n/2) \) and \( (n + 10^z)/2 \) gives a \( O(\log n) \) solution.

UVa 1051 Bipartite Numbers\(^1\)

Statement  A bipartite number is defined as a number of the form \( s \ldots s t \ldots t, s \neq t \).

Given an list of numbers, for each number find the smallest bipartite number strictly divisible by it.

\(^1\)ICPC 3564: World Finals 2006
5 Algebra

Solution  We will solve the problem for each number $x$ by generating bipartite numbers in order modulo $x$ and testing if any is zero. If $x$ is already bipartite, skip the first found number.

To generate bipartite numbers in order first set the size, this is $n + m$, smaller first. Then set the first number $s$, smaller first. Then decide if $t$ will be smaller or greater than $s$. If $t$ will be smaller, then set $m$, smaller first, and finally set $t$, smaller first. If $t$ will be greater, then set $n$, smaller first, and finally set $t$, smaller first.

To compute a bipartite number modulo $x$, first precompute $f(n) = \frac{10f(n - 1) + 1}{n}$ mod $x$ using the recurrence $f(n) = 10f(n - 1) + 1$, adding one element at each iteration of the size. The bipartite number is $s(f(m + n) - f(n)) + tf(n) \mod x$.

For performance reasons a special case should be handled separately: if $x \equiv 0 \mod 10^a$, with $a > 1$, then fix $t = 0$ and $n \geq a$.

Timus 1484 Film Rating

Statement  Let $x$ be a sequence of $n$ elements in $[10]$. $f(x) = \sum_{i=1}^{n} x_i / n$ rounded to 1 decimal places. Given $f$, $g$ and $n$, find the minimum $m$ such that a sequence $y$ of $m$ elements exists with $f(xy) \leq g$ for all $x$ satisfying $|x| = n$ and $f(x) = f$.

Solution  If $f(x) \leq g$, the solution is $m = 0$. Otherwise add a sequence of $m$ ones. Let us show how to compute $m$. First compute the maximum possible $s = \sum x_i$. If must hold that $s/n + 0.05 < f$ and $s \leq 10n$. The maximum integer that satisfies it is $\min(\lceil (f + 0.05)n \rceil - 1, 10n)$. Then compute the minimum possible $m$. It must hold that $(s + m)/(n + m) + 0.05 < g$. This is $m > (s + g + 0.05)/(g - 0.95)$. The minimum integer that satisfies it is $\lceil (s + g + 0.05)/(g - 0.95) \rceil$.

Caveats  Work with fixed precision numbers (2 decimal digits) instead of floating point numbers.

Timus 1539 Intelligence Data

Statement  $n$ positive fractions are given rounded to $d$ decimal places. Compute their minimum common denominator.

Solution  Given a valid denominator $m$, it must hold that for each fraction $x_i$ an integer $n_i$ exists such that $x_i - 5 \cdot 10^{-d-1} \leq n_i/m < x_i + 5 \cdot 10^{-d-1}$. This is $m(x_i - 5 \cdot 10^{-d-1}) < m(x_i + 5 \cdot 10^{-d-1})$. Iterate through all denominators in increasing order until a valid one is found.
Caveats  Work with fixed precision numbers \((d + 1\) decimal digits).

**ICPC 5900 Binomial coefficients**

**Statement**  Given \(m > 2\), find all the pairs \(n,k\) such that \(\binom{n}{k} = m\).

**Solution**  Assume \(k \leq n - k\). Given \(k\), a rough bound for \(n\) is \(m = \binom{n}{k} \geq \left(\frac{n-k}{k}\right)^k\). So, if a solution for \(k\) exists, \(n \in [k + 1, \sqrt[m]{m}]\), and a binary search will find it. A rough bound for \(k\) is \(\binom{n}{k} \geq 2^k\), so it is enough to check \(k \in [1, \log_2 m]\).

Caveats  To avoid overflows while computing \(\binom{n}{k}\) iteratively, abort the computation if a partial result exceeds \(m\).

**Timus 1541 Chase**

**Statement**  Given a fraction \(m/n \leq 3\), \(1 \leq m,n \leq 50\), \((m,n) = 1\), find a set of less than 20 different fractions \(1/x_i\) such that \(\sum_i x_i = m/n\), \(1 \leq x_i \leq 100000\).

**Solution**  Compute the divisors of \(1080n\). Choose them greedily in decreasing order ensuring that \(\sum d_i \leq 1080m\). The set of fractions \(d_i/1080n\) satisfies the conditions. This can be proved by exhaustive search, and 13 fractions are always enough.

**5.3 Linear algebra**

**Timus 1041 Nikifor**

**Statement**  Given \(m\) vectors \(v_i \in \mathbb{Z}^n\) and costs \(c(v_i) \in \mathbb{Z}^+\), find a subset of vectors that is a basis for \(\mathbb{R}^n\) and minimizes the sum of costs.

**Solution**  Let us show that sorting the vectors by cost and greedily adding them to a growing linearly independent set is an optimum solution; call it \(\langle v_j \rangle\). Assume \(\langle w_j \rangle\) is an optimum solution. Sort them by cost and let \(J\) be the first index where they differ. \(v_J \in \langle w_1 \ldots w_n \rangle\), but \(v_J \notin \langle w_1 \ldots w_{J-1} \rangle = \langle v_1 \ldots v_{J-1} \rangle\), so \(v_J\) can replace some \(w_J\), \(J > J\) in \(\langle w_j \rangle\) and it will still be a basis. By construction \(c(v_J) \leq c(w_j)\) for all \(j \geq J\), and since \(\langle w_j \rangle\) is an optimum the costs must be equal and the new basis is a new optimum with one more vector in common with \(\langle v_j \rangle\). By induction, all the vectors can be replaced and \(\langle v_j \rangle\) is also an optimum.

A solution that checks the linearly independent set condition is a Gaussian elimination form that adds the vectors sequentially in increasing cost order. The \(k\)th vector in the set is transformed so that it has zeros in the components \([0..k]\). Partial pivoting of the rows is needed. Equivalently, think of the \(m\) vectors as a \(n \times m\) matrix.
5 Algebra

Caveats  Solutions with IEEE 64 bit floating point numbers and partial pivoting are apparently not stable enough but an exact solution with arbitrary precision integers is needed. Note that complete pivoting cannot be applied because the column order is given by the costs.

Timus 1561 Winnie the Pooh

Statement  Consider a set of integer variables and equations, initially empty. $n$ instructions are given. An instruction can either add a new variable, add an equation of the type $d_0 + \sum a_i x_i = d_1 \mod 7$, or ask for the result of $d_0 + \sum a_i x_i$. Answer the queries according to the information given so far.

Solution  Consider variables and coefficients as members of the field $\mathbb{Z}_7$. Keep a square matrix and a vector in gaussian form for the lineal system $Ax = b$.

To add a variable, add a row and a column with zeros to the matrix.

To add an equation, add it as a row and perform gaussian elimination on the row. If any of the columns, say the $i$-th, does not zero out, then we can swap the new row with the $i$-th. If all of them are zero and the corresponding entry in $b$ is also zero, the equation is redundant and we can ignore it. If the entry in $b$ is not zero, we derived a contradiction so all the following answers will be a contradiction.

To answer a query of type $z = \sum c_i x_i$, add the row and again perform gaussian elimination, setting the entry in $b$ to 0 and $z + 0$ conceptually. If the resulting row in $A$ is zero and $z'$ in $b$, we have derived the equation $0 = z + z'$, so we can answer $z = -z'$. Otherwise we have an equation of the type $c'x = z + z'$ that is consistent with all the previous equations for any value of $z$, so the answer is undefined.

Timus 1582 Bookmakers

Statement  A triple $\langle a_1, a_2, a_3 \rangle$ is given. Find $\max \{x \in \mathbb{R}^3 : \sum x_i = k \} (\min_i a_i x_i)$.

Solution  The optimal solution must satisfy $a_1 x_1 = a_2 x_2 = a_3 x_3$ or a better solution would exist. Indeed, suppose $a_1 x_1 < a_2 x_2 \leq a_3 x_3$. Then $a_1 (x_1 + \varepsilon) = a_2 (x_2 - \varepsilon)$ for some $\varepsilon > 0$ would be a strictly better solution.

So we want to solve the equation

$$
\begin{pmatrix}
  a_1 & -a_2 & 0 \\
  0 & a_2 & -a_3 \\
  -a_1 & 0 & a_3 \\
  a_1 & a_2 & a_3 
\end{pmatrix}
\begin{pmatrix}
  x_1 \\
  x_2 \\
  x_3 
\end{pmatrix} =
\begin{pmatrix}
  0 \\
  0 \\
  k
\end{pmatrix}
$$
5.4 Linear programming

which gives

\[
x = \frac{1}{a_1a_2 + a_2a_3 + a_3a_1} \begin{pmatrix} a_2a_3 \\ a_3a_1 \\ a_1a_2 \end{pmatrix}
\]

and the final answer

\[
\frac{a_1a_2a_3}{a_1a_2 + a_2a_3 + a_3a_1}
\]

Notice that the answer is the harmonic mean, so there may be a more beautiful proof involving it.

UVa 1074 Net Loss²

**Statement** Given a polynomial \( g(x) \), find a piecewise linear continuous function \( f(x) \) with two segments joined at \( x = k \) that minimizes \( \int_{-1}^1 f(x)g(x)dx \).

**Solution** The search space is a vector space \( F \) generated by the functions \( f_1 = \chi_{[-1,k]}(x-k), f_2 = \chi_{[k,1]}(x-k), f_3 = 1 \). The function to minimize is the scalar product in \( L_2 \), so the function we are looking for is the orthogonal projection of \( g \in L_2 \) into \( F \).

To compute the projection, let us first find an orthogonal basis for \( F \), using the Gram-Schmidt process. \( \langle f_1, f_2 \rangle = 0 \), so \( f'_1 = f_1 \) and \( f'_2 = f_2 \). \( \|f'_1\|^2 = \frac{1}{3}(1+k)^3 \). \( \|f'_2\|^2 = \frac{1}{3}(1-k)^3 \). \( \langle f'_1, f_3 \rangle = -\frac{1}{2}(1+k)^2 \). \( \langle f'_2, f_3 \rangle = \frac{1}{2}(1-k)^2 \). So \( f'_3 = 1 + \frac{3}{2(1+k)}f_1 + \frac{3}{2(k-1)}f_2 \).

The coefficients of \( \pi_F(g) \) in this orthogonal basis are \( c_i = \langle f'_i, g \rangle / \|f'_i\|^2 \). Compute them using the exact antiderivative of a polynomial and Barrow’s rule. The final result is

\[
\begin{cases}
  (c_1 + \frac{3c_3}{2(1+k)}) (x-k) + c_3 & -1 \leq x \leq k \\
  (c_2 + \frac{3c_3}{2(k-1)}) (x-k) + c_3 & k < x \leq 1
\end{cases}
\]

5.4 Linear programming

Linear programming problems are optimization problems of the type \( \min c^\top x, \ Ax \preceq b \), with \( c, x \in \mathbb{R}^n \), \( A \in M_{m \times n}(\mathbb{R}) \), \( b \in \mathbb{R}^m \), where the (in)equality relations are row-wise and can be different for each row.

The simplex algorithm, even though with a theoretical run time exponential in the number of variables, is fast enough in practice for solving linear programming problems.

The simplex algorithm solves problems of the type \( \min cx, Ax = b, x \leq 0 \). More general problems can be reduced to it by introducing slack variables. e.g. a constraint of the type \( Ax \leq b \) would be transformed to \( Ax + y = b, y \geq 0 \).

²ICPC 4124: World Finals 2008
At a bird’s eye, the algorithm keeps a list of \( m \) variables allowed to be non-zero, called the basis. The submatrix \( B \subset A \) associated to these variables is required to be invertible and so \( x = B^{-1}b \) is a feasible but not necessarily optimal solution. The matrix \( B^{-1} \begin{pmatrix} A & b \\ c \end{pmatrix} \) is also kept and called the tableau. At each step the algorithm selects a variable to enter the basis using the \( c \) row in the tableau, selects a variable to leave the basis using the entering variable column in the tableau, and updates the tableau to reflect the new basis. When no more steps can be performed, the solution is optimal. For more details on how and why this steps are performed, see [12, 13].

To find an initial basis, \( m \) artificial variables are added with \( B = I_m \) and the cost function is changed to \((0, 1, \ldots, 1)^T\). Simplex iterations are run until a minimum is found. If this minimum is zero, the algorithm can continue with the original costs after flagging some variables (including artificial) to be unfeasible and thus not eligible for entering the basis. Otherwise, no solution exists.

The variation of the algorithm we added to the library is the revised primal, full canonical tableau, two phases, full artificial basis in phase I, explicit inverse form, with optional reinversions[12]. Ties in entering and leaving variable selection are broken up using Bland’s rule[18].

A similar version of the simplex algorithm already existed in our library, but it did not correctly handle some degenerate cases with infeasible variables. Additionally, its stability was improved.

A general caveat on using the simplex algorithm with floating point numbers: multiplying the system so that \( A \) has an operator norm close to 1 will lead to more stable intermediate calculations.

**UVa 802  Lead or Gold**

**Statement** Given a list of \( n \) triples \( \langle a_i, b_i, c_i \rangle \) and a target \( \langle a, b, c \rangle \) find whether a combination \( \sum x_i \langle a_i, b_i, c_i \rangle, x_i \in \mathbb{Z}^+ \) exists with ratios equivalent to the target.

**Solution** By multiplying the solution coefficients by the lcm of the denominators, we can assume that an integer solution exists iff a rational solution exists. The problem is now reduced to a standard lineal programming \( \min cx, \ Ax = b, \ x \geq 0 \), with \( A = \begin{pmatrix} a_i \\ b_i \\ c_i \end{pmatrix} \),

\[
b = \begin{pmatrix} a \\ b \\ c \end{pmatrix}, \ c = (1).\] The costs are irrelevant, so they are set to 1 to avoid precision issues.
6 Calculus

This chapter includes analysis-related parts of mathematics: numerical methods for integration, zero-finding, extrema-finding and function approximation.

6.1 Numerical integration

The simplest but yet accurate numerical integration method is the compound trapezes method, which breaks the integration interval into \( n \) intervals of width \( h = (b - a)/n \) and approximates the function by a polynomial of degree 0. This is

\[
I = (b - a) \sum_{i=1}^{n} f(a + (i - 1/2)(b - a))
\]

UVa 1036 Suspense!

Statement  Consider two vertical segments of heights \( 3j - 2 \) and \( 3t - 2 \) whose lower endpoints are at the \( x \) axis and are at distance \( d \) from each other. Points at heights \( 3k + 1 \) may be of types \( b, c \) or none. An horizontal line at height \( h \in [1, 3 \min(j, t) - 2] \) is feasible if \( h \) is not in the interval \( (b - 3, b + 1/2) \) for any point \( c \) or it is not in the interval \( (b - 1/2, b + 3) \) for any point \( b \). A convex parabola is feasible if it passes through the tops of the segments, its minimum is at height \( 1 + h \) and the line at \( h \) is feasible. Find the maximum length of such a parabola.

Solution  If a parabola is feasible, lowering its minimum towards the nearest half integer is still feasible and is longer, so we can restrict the search to half integers. Thus, start at \( h = 1 \) and check if the line is feasible. A line at a half integer height is not feasible if either of the two \( ([h/3] + 1) \)-th points is of type \( c \) and either of the two \( ([h/3] + 1) \)-th points is of type \( b \). Increase \( h \) by \( 1/2 \) until a feasible line is found.

It remains to compute the length of the parabola. Assume its minimum is at \((0, 0)\). Then the parabola is of the form \( y = f(x) = Ax^2 \). Say its endpoints are \((-p, a)\) and \((q, b)\). We know \( 3j - h - 1 = a = f(-p), 3t - h - 1 = b = f(q) \), and \( d = p + q \). Then, \( p = \sqrt{a/A}, q = \sqrt{b/A}, d = \sqrt{a/A} + \sqrt{b/A}, \) so \( A = ((\sqrt{a} + \sqrt{b})/d)^2 \).  

---

\(^1\)ICPC 3001: World Finals 2004
The length of a general curve \( y = f(x) \) along the interval \( x \in [p, q] \) is
\[
\int_p^q \sqrt{1 + f'(x)^2} \, dx.
\]
To get a geometric intuition (but not a proof!), think about applying Pythagoras’ theorem on the segment \((x, f(x)), (x + dx, f(x + dx))\). The formula applied to our parabola yields
\[
\int_p^q \sqrt{1 + (2Ax)^2} \, dx.
\]
The antiderivative is not straightforward, so it’s easier to compute the integral numerically with the rectangles method.

**Notes** Félix Miravé collaborated in solving this problem.

### 6.2 Root finding

The Weierstrass method, also known as the Durand-Kerner method, is an iterative root finding method based on the fixed point principle. It can approximate all the roots of a polynomial that has only simple roots. Note that roots will be complex in the general case, so complex arithmetic is needed.

Let
\[
P(x) = \prod_{i=1}^d (x - x_i). \quad x_i = x - p(x)/\prod_{j \neq i} (x - x_j),
\]
so this suggests the iterative method
\[
x_i^{(n+1)} = x_i^{(n)} - \frac{p(x_i^{(n)})}{\prod_{j \neq i} (x_i^{(n)} - x_j^{(n)})},
\]

It appears to be folklore that this method converges globally for initial values \(x_i^0\) in the disk \(1 + \max_i |g_i|\) and spaced enough. Known theoretical results such as [15] are weaker.

Actually the method is usually implemented using the Gauss-Seidel variation, where \(x_i^{(n+1)}\) is computed from \(x_j^{(n+1)}\) for \(j < i\) and from \(x_j^{(n)}\) for \(j \geq i\).

**Timus 1621 Definite Integral**

**Statement** Compute \(\int_{-\infty}^{\infty} \frac{dx}{P(x)}\), where \(P(x)\) is a polynomial of degree exactly 4 with no real roots and \((P(x), P'(x))\) constant.

**Solution** Since \(P(x)\) has no real roots, its complex inverse \(1/P(z)\) has no poles in the real line. Consider the closed curve \(\gamma(r) = [-r, r] \cup re^{i[0,\pi]}\), a real segment plus an upper semicircle. Consider the integral \(I(r) = \int_{\gamma(r)} \frac{dz}{P(z)} = \int_{-r}^{r} \frac{dz}{P(z)} + \int_{|z|=r, \Im(z) \geq 0} \frac{dz}{P(z)} \). \(|P(z)| = O(|z|^4)\), so the second part of the integral is \(o(|z|^{-3})\). Taking the limit as \(r \to \infty\), we get \(\lim_{r \to \infty} I(r) = \int_{-\infty}^{\infty} \frac{dz}{P(z)}\). But since \(P(x)\) has only simple roots, \(1/P(z)\) has only simple poles, and so for \(r\) large enough (so that \(\gamma\) encloses all the roots of \(P(x)\)), the Cauchy integral formula \(I(r) = 2\pi i \sum_{P(w)=0, \Im(w)=0} \frac{\gamma(w)}{P'(z)}\) holds. It only remains to compute the roots of \(P(x)\). This reduction is standard in complex analysis.
An exact formula, known as Ferrari’s formula, exists to compute the roots of a quartic polynomial, but it is not practical. Instead we can use the Weierstrass method and find a good numerical approximation.

Caveats  IEEE 64 bit floats and the C++ standard complex library are apparently not precise enough for this approach. Instead, IEEE 128 bit floats can be emulated with the java.math.BigDecimal library and a custom complex class.

ICPC 6027 Curvy Little Bottles

Statement  Let $p : [a, b] \to \mathbb{R}^+$ be a degree $n$ polynomial. Let $V(c, d)$ be the volume produced by revolving $p$ around the $x$ axis in the interval $[c, d] \subset [a, b]$. Compute $V(a, b)$ and a list $x_1 \ldots x_m$ such that $x_0 = a, x_m \leq b, m \leq 8$ and $V(x_i, x_{i+1}) = k$.

Solution  $V(c, d) = \pi \int_c^d p(x)^2 dx$. Compute $p^2$ with the standard $O(n^2)$ multiplication algorithm and a primitive. Compute $V$ using Barrow’s rule. $V$ is monotonically increasing, so we can compute the list $x_i$ with a binary search.
7 Optimization

Optimization problems involve minimizing a function in a domain and finding either the minimum value or a certificate.

7.1 Greedy algorithms

Greedy algorithms greatly reduce the search space by choosing always the local minimum.

Timus 1113 Jeep

Statement  Minimize the fuel a Jeep needs to travel a given distance according to the rules of the Jeep Problem [19].

Solution  Each additional drum is more fuel inefficient, so the optimal solution is to use as few drums as possible (proof omitted). This suggests an algorithm that adds unit drums until a last possibly fractional drum can be added.

Timus 1131 Copying

Statement  Consider the empty graph of $n$ nodes and a set $A = v_0$. At each step, at most $k$ disjoint edges from $A$ to $V - A$ can be added. Compute the minimum number of steps for $A$ to cover the graph.

Solution  By adding as many edges as possible, the size of $A$ grows exponentially until the growth rate reaches $k$, then linearly. So, sequentially compute $\sum_{i=0}^{m} 2^i$ until $2^i > k$ or $\sum_{i=0}^{m} 2^i \geq n$. The solution is $\lceil (n - \sum_{i=0}^{m} 2^i)/k \rceil$. The cost of the algorithm is $O(\log k)$.

Timus 1161 Stripies

Statement  Given a set of $n$ positive integers, the following transform can be applied: choose two numbers $a$, $b$, remove them from the set, and add $2\sqrt{ab}$ back. Minimize the sum of the set after applying some transforms.

Optimization

Solution  A transform never increases the sum: $2\sqrt{ab} \leq 2(a + b)/2$, so it is optimal to apply $n - 1$ transforms. Also, the largest number should be under as many square roots as possible. An algorithm that satisfies this conditions is to insert all the numbers in a heap and successively perform the transform with the two top elements, inserting the result back into the heap. Its cost is $O(n \log n)$.

Timus 1587  Flying Pig

Statement  A list $\{x_i\}$ of $n$ integers with $|x_i| \leq 3$ is given. Find the maximum product of the numbers in any non empty interval.

Solution  Split the problem at the positions where $x_i = 0$, since all products that include a zero are zero. For each interval, check if the product of all the numbers is positive. If so, this is the maximum. Otherwise, at least a negative number exists. Conceptually erase numbers from the right until a negative number is found. The product in the interval is now positive. Same starting from the left. All other intervals with a positive product are a subset of either of these, so one of them is the maximum.

To compute the product of an interval, count the number of 2s and 3s in absolute value and the sign parity. Use the java.math.BigInteger library to compute the appropriate power of 3, shift it left, and set the sign.

Take the maximum among all the positive intervals. If none exists, then the answer is the maximum number in the original interval.

UVa 1019  Light Bulbs\textsuperscript{1}

Statement  A $n$-bit number $a$ can be transformed by the $n$-bit number $c$ into $b = a \oplus \bigoplus_{i \in [n]} c_i (2^{i-1} + 2^i + 2^{i+1})$ (ignoring bits 0 and $n + 1$). Given $a$ and $b$, minimize the number of set bits of $c$.

Solution  Assume $c_1 \ldots c_i$ are fixed. Then $b_i = a_i \oplus c_{i-1} \oplus c_i \oplus c_{i+1}$, so $c_{i+1}$ is also known. By induction, all $c_i$ are fixed but $c_0$. The proposed algorithm chooses $c_0$, computes $c_i$ iterating, and finally, when $c_n$ is set, checks that $b_n = a_n \oplus c_{n-1} \oplus c_n$ or reports that the choice is invalid. Its cost is $O(n)$.

Caveats  Binary to decimal conversions are better left to the java.math.BigInteger library.

\textsuperscript{1}ICPC 2722: World Finals 2003
7.2 Dynamic programming

**Timus 1534 Football in Gondor**

**Statement**  A player plays $n$ matches, scores $k$ goals and is scored $l$ goals. The score of a game is 3,1 or 0 if the number of goals for is greater, equal or less than the number of goals against. Find the maximum and minimum possible scores.

**Solution**  The maximum score is to win $\min(k, n-1)$ games 1–0, draw $\max(0, n-1-k)$ games 0–0 and distribute the remaining goals in a single match. The minimum score is to either lose $\min(l, n-1)$ games 0–1, draw $\max(0, n-1-l)$ games 0–0 and distribute the remaining goals in a single match or to draw all the games should this be possible.

**Timus 1483 Table Football**

**Statement**  $n$ players play exactly once against each other. The score of a game is 3,1 or 0 if a player wins, draws or loses. Teams are sorted by total score and ties not broken. Find the minimum possible score of the first player and the maximum possible score of the last player.

**Solution**  A first player must achieve at least $N/n$ points, where $N$ is the total number of points awarded. When all the matches are draws $N$ is minimum, and all players can achieve exactly $N/n = n-1$ points, so the solution is optimum.

A last player must achieve at most $N/n$ points. When all the matches are wins $N$ is maximum, so for $n$ odd all players can achieve exactly $3(n-1)/2$ points and the solution is optimum. For $n$ even a last player cannot reach $n/2$ wins, otherwise the total number of wins would be at least $n^2/2 > n(n-1)/2$ which is the number of games. However, it can achieve $n/2 - 1$ wins and a draw, for a total of $N/n = 3n/2 - 2$ points. As we showed that any solution with more points is impossible, this solution is optimum.

**Timus 1535 The Hobbit or There and Back Again**

**Statement**  Consider the graph $K_n$ with edge costs $c((i,j)) = ij$. Find a minimum and a maximum hamiltonian circuit starting at 1.

**Solution**  If $n \equiv 0 \mod 2$, a minimum circuit is $1, n-1, 3, n-3, \ldots, n/2, n/2+1, n/2-1, n/2 + 3, n/2 - 3 \ldots n$ and a maximum circuit is $1, 3, \ldots, n-1, n, n-2, \ldots, 2$. If $n \equiv 1 \mod 2$, a minimum circuit is $1, n-1, 3, n-3, \ldots n$ and a maximum circuit is $1, 3, \ldots, n, n-1, n-3, \ldots, 2$ (proof omitted).

7.2 Dynamic programming

Dynamic programming algorithms reduce the search space by using a recurrence.
7 Optimization

Timus 1167 Bicolored Horses

Statement  Given an ordered sequence of \( n \) bits, find a partition \( \mathcal{P} \) of \([n]\) into exactly \( k \) non-empty intervals such that \( \sum_{I \in \mathcal{P}} c(I) \) is minimal, where \( c(I) := |\{i \in I : x_i = 0\}| \cdot |\{i \in I : x_i = 1\}| \).

Solution  Standard dynamic programming. The recurrence is

\[
f(n,k) = \min_{i \in [n-1]} c([i+1,n]) + f(i,k-1)
\]

To compute \( c([a,b]) \) in constant time, compute previously \(|\{i \in [j] : x_i = 0\}|\) and \(|\{i \in [j] : x_i = 1\}|\).

Timus 1592 Chinese Watches

Statement  \( n \) numbers in \([m]\) are given. \( d(x,y) \) is defined as the canonical (smallest non negative) element of the class \([y-x] \mod m\). Find an element \( y \in [m] \) that minimizes \( \sum_i d(x_i,y) \).

Solution  Compute \( \sum_i d(x_i,1) \) looping through all \( x \). Compute the sum for the other elements of \([m]\) using the recurrence \( \sum_i d(x_i,y+1) = \sum_i d(x_i,y) + n - m |\{i : x_i = y\}| \).

This is, each element’s cost increases in 1 but those for which the new cost would be \( m \), for which the cost decreases in \( m - 1 \). Precompute \(|\{i : x_i = y\}|\). The cost of this approach is \( O(n+m) \).

7.3 Exhaustive search

An exhaustive search traverses all the search space. Its complexity is asymptotically the same as a backtracking.

UVa 1047 Zones

Statement  The cardinality of \( n \) sets is given. The cardinality of their \( m \) intersections is given in PIE form, this is considering the intersection regions as a partition. Choose \( k \) sets so that the cardinality of their union is maximum.

\[ ^2 \text{ICPC 3278: World Finals 2005} \]
7.3 Exhaustive search

Solution  Represent the set of sets $S$ as a bit pattern $p$, where a set $s_i \in S$ iff the $i$-th bit of $p$ is set. Patterns and sets will be identified. Iterate through all patterns of length $n$ and $k$ ones. For each set $S$, compute its disjoint union size as $\sum_{s_i \in S} |s_i|$ and then subtract the number of elements counted more than once. The repeated elements are those in each intersection $R_j$, and they have been counted as many times as the number of common sets in the intersection and the set, this is $|s_i \in R_j \cap S|$. Since they should be counted only once, subtract $r_j \cdot (|R_j \cap S| - 1)$.

The cost of each iteration is $O(n + m)$, since set operations take constant time. The $|\cdot|$ operation corresponds to the amd64 instruction POPCNT. The total cost of the algorithm is then $O(\binom{n}{k} n + m) = O(2^n(n + k))$. 
8 Geometry

8.1 Basic formulae

The followed approach for geometric problems is to use a point data structure which
stores its coordinates and has some basic operations to work with. Because of the
ability of C++ to redefine operators, this provides a familiar syntax at the cost of some
extra allocations.

It is generally wise to avoid floating point computations because of rounding errors. So,
it is important to know which operations work with integers and use an integer data
type when possible or which do not and use approximate comparisons instead.

The operations addition, subtraction, multiplication by a scalar, scalar product, wedge
product and squared modulo are standard and work with integers.

The operation division by a scalar is exact if the divisor is a power of 2. The operations
modulo and argument generally would result in irrational numbers, so their floating
point output should be considered as approximate.

Point-line distance

\[
\begin{align*}
\left| \frac{|ab \land ap|}{\|ab\|} \right| = \\
d(p, ab) = \|ap\| \sin \alpha
\end{align*}
\]

Do not divide by \(\|ab\|\) when comparing distances to the same line or using the signed
distance to know whether a point lies to the left or to the right of the oriented line.

Point-segment distance

Similar to point-line but checking that the projection of \(p\) lies inside \(ab\).

\[
\begin{align*}
d(p, ab) = \begin{cases}
  d(p, ab) & 0 \leq ab \cdot ap \leq \|ap\|^2 \\
  \min\{d(p, a), d(p, b)\} & \text{otherwise}
\end{cases}
\end{align*}
\]

Angle between vectors

\[
\begin{align*}
\sin \alpha &= \frac{u \land v}{\|u\|\|v\|}, \\
\cos \alpha &= \frac{u \cdot v}{\|u\|\|v\|}, \\
\tan \alpha &= \frac{u \land v}{u \cdot v}
\end{align*}
\]

To actually compute \(\alpha\) use \(\text{atan2}()\).

Timus 1111 Squares

Statement

Given \(n\) squares and a point \(p\), sort the squares by distance from \(p\) to them.
8  Geometry

**Solution**  For each square, first check if \( p \) lies inside it with the *always left* algorithm and return 0 if true. Otherwise set the distance to the minimum of the segments of the square.

**Caveats**  Avoid square roots in vertex computations by performing a \( \pi/4 \) rotation (\( \sin \pi/4 = \cos \pi/4 = \sqrt{2}/2 \)) and a \( 1/\sqrt{2} \) scaling in the same operation.

**Caveats**  Avoid square roots by using the squared distance.

---

**Timus 1168  Radio Stations**

**Statement**  A set of integer points in \( \mathbb{R}^3 \) such that the projection into the \( z = 0 \) plane is a \( m \times n \) lattice is given. \( k \) such points are selected and \( k \) closed balls are centred on them. Find all integer points (not strictly) in the intersection of the balls whose projections fall inside the lattice and are not lower than the initial points.

**Solution**  For each point in the lattice, compute the maximum and minimum possible heights by iterating for each ball. Additionally consider the initial point for the minimum bound. The boundary heights can be computed as \( z_k \pm \sqrt{r_k^2 - \Delta x^2 - \Delta y^2} \). Skip the point if the square root is not defined for any ball. This gives an \( O(mnk) \) algorithm.

---

**Timus 1093  Darts**

**Statement**  A disk in \( \mathbb{R}^3 \) is given as centre \( c \), normal vector \( n \) and radius \( r \). A curve is given parametrized as \( m(t) = s + vt - 5kt^2, \ t \geq 0 \). Compute whether the disk and the curve intersect.

**Solution**  Find the parameter values where the curve intersects the plane that contains the disk by solving for \( t \) the quadratic equation \( m(t) \cdot n = c \cdot n \). Return whether any solution \( t_i \) satisfies \( \| m(t_i) - c \|^2 < r^2 \) and \( t_i \geq 0 \).

**Caveats**  The equation can be degenerate.

---

**Timus 1151  Radiobeacons**

**Statement**  \( n \) points in \( [c]^2 \) with unknown coordinates exist. \( m \) points in \( [c]^2 \), are given as well as their \( \| \cdot \|_\infty \) distance to some of the unknown points. Find the coordinates of the unknown points or report ambiguity.
8.1 Basic formulae

Solution Each unknown point is in the intersection of some ball boundaries with known centres and radii. Notice that balls of radius \( r \) in \( \|\cdot\|_\infty \) are squares with sides parallel to the axis of length \( 2r \) in the euclidean plane, where we will actually work.

Consider each square as a union of segments. The intersection of unions of segments is again a union of segments, some of which can degenerate to points. Moreover, if care is taken to eliminate points that lie inside segments in the union, then the number of segments never increases.

The solution is then to keep a set of valid segments for each unknown point. For each given distance to it, compute the ball boundary (square), restrict it to \([c]^2\) and intersect it with the valid set, updating the set. Initialize the set with the first square. Finally check whether the valid set is a single point. The cost of this solution is \( O(mn) \) in time and \( O(n) \) in space.

Solution A simpler solution is to keep an array over the \([c]^2\) lattice for each unknown point that contains the number of ball boundaries that pass through a given point in the lattice. The unknown point location is the one that attains the maximum or unknown is more than one does. The cost is \( O(cn^2) \) in time and \( O(c^2 n) \) in space. However, it is simpler to implement and actually faster for small datasets.

Timus 1482 Triangle Game

Statement Two triangles \( A \) and \( B \) are given. Find a translation or rotation \( T \) such that \( T(A) = B \).

Solution Fix a permutation of the vertexes of \( B \). Let \( v = a_1a_2, w = b_1b_2 \) and \( d = a_1b_1 \). The angle of the transform is the angle \( \alpha \) between \( v \) and \( w \). If it is zero, the transform must be a translation of vector \( d \). Otherwise, it must be a rotation whose centre lies in the bisector of the segment \( d \). The rotation centre is at distance \( |d/2| / \tan(\alpha/2) \) to the midpoint of \( d \) (see figure 8.1). Apply the rotation as a composition of two translations and a rotation around the origin.

For each permutation find the associated transform. Test if it correctly maps \( A \) to \( B \). If so, output it and halt. If no permutation yields a valid transform, none exists.

UVa 1058 Grand Prix$^1$

Statement A polygonal line in the plane \( z = x \tan \theta \) is given. Find the minimum angle in absolute value of a rotation within the plane such that no segment in the resulting line points downwards in the given order.

$^1$ICPC 2377: World Finals 2007
8 Geometry

Solution  Rotate the problem $+\pi/2$ along the $z$ axis for convenience. Now consider rotations in the plane in the interval $[-\pi, \pi)$. A segment $v = pq$ points downwards iff $\theta > 0$ and $v_y < 0$. A segment with $v_y \geq 0$ is in a correct position but cannot be rotated too much. It imposes an upper bound of $\pi - \alpha$ on positive rotations and $\alpha$ on negative rotations. A segment with $v_y < 0$ needs to be rotated. It imposes a lower bound of $-\alpha$ on positive rotations and $\pi + \alpha$ on negative rotations.

Compute the maximum and minimum bounds for positive and negative rotation directions separately and choose the feasible direction with the smallest lower bound. If it is 0 or $\theta = 0$ or no feasible direction exists, report it.

Volumes

Timus 1550 Dean’s Pyramid 3

Statement  Find the volume of a square pyramid of height $h$ and side $w$, base centred at $O$, from which the intersection with a cylinder of radius $r$, infinite height, base centred at $(x, y, 0)$ is removed. The intersection does not contain any edge of the pyramid.

Solution  The volume of the intersection is equivalent to the volume of a cylinder of radius $r$ and height the intersection of the line $(x, y)$ with the pyramid. This height is $h(w/2 - \max(|x|, |y|))/(w/2)$. So the total volume is

$$\frac{hw^2}{3} - \pi r^2 h \left(1 - 2 \max(|x|, |y|) \right)/(w/2)$$

8.2 Intersections

Lines in point-vector form

$$(p, v) \cap (q, w) = p + \lambda v = q + \mu w$$
8.3 Basic constructions

\[ p \land w + \lambda v \land w = q \land w \implies \lambda = \frac{(q - p) \land w}{v \land w} \]

\[ p \land v = q \land v + \lambda w \land v \implies \mu = \frac{(q - p) \land v}{v \land w} \]

The intersection is a point if \( v \land w \neq 0 \); otherwise it is a line or the empty set. We call \( \lambda \) and \( \mu \) the intersection parameters. If \((p, v)\) is a ray, check that \( \lambda \geq 0 \). If it is a segment, check that \( 0 \leq \lambda \leq 1 \). Same for \((q, w)\) and \( \mu \).

**Lines in general form**

\[ A_1 x + A_2 y + C = B_1 x + B_2 y + D \implies (x, y) = \left( \begin{array}{c} A_1 \\ B_1 \end{array} \right)^{-1} \left( \begin{array}{c} C \\ D \end{array} \right) \]

**Planes in point-normal vector form**

\[ \pi(p, v) \cap \eta(q, w) = x + \lambda(v \land w) \]

\[ \begin{pmatrix} v^\perp \\ w^\perp \end{pmatrix} x = \begin{pmatrix} v^\perp p \\ w^\perp q \end{pmatrix} \]

The intersection is a line directed like \( v \land w \) that passes through a point \( x \) satisfying the plane condition for both \( \pi \) and \( \eta \). This point can be found by inverting one of the 3 \( 2 \times 2 \) submatrices of the \( 2 \times 3 \) matrix \( \begin{pmatrix} v^\perp \\ w^\perp \end{pmatrix} \). It is more numerically stable to choose that with the greater determinant in absolute value. If all of the determinants are zero, then \( v \land w = 0 \) and the intersection is not a line but a plane or the empty set.

**8.3 Basic constructions**

**Bisector of two points**

\[ \left( \frac{p + q}{2}, (q - p)^\perp \right) \]

**Bisector of two lines**

\[ \left( r \cap s, \frac{u}{\|u\|} + \frac{v}{\|v\|} \right) \]

Where \( r \cap s \) is the intersection of \( r \) and \( s \) or \((p + q)/2\) if it does not exist.
Circle passing through 3 points  Find the circumcentre of the triangle defined by the points. The naive way is to intersect the bisectors of two pairs of them. A more stable computation, according to [14] and rearranging, is

\[ D = 2(a \land b + b \land c + c \land a) \]
\[ u_x = \frac{a^2(b_y - c_y) + b^2(c_y - a_y) + c^2(a_y - b_y)}{D} \]
\[ u_y = -\frac{a^2(b_x - c_x) + b^2(c_x - a_x) + c^2(a_x - b_x)}{D} \]

Circle over a line, tangent to a line and passing through a point  Let \( p \) be the intersection of the two lines. Write the first line as \( p + \lambda v \) and the second as \( p + \mu w \). The solution’s distance to the line must be the same as the point’s. Thus,

\[ \frac{(\lambda v) \land w}{\|w\|} = \|p + \lambda v - q\| \]

Writing \( r = p - q \) and squaring,

\[ 0 = r^2 + 2\lambda rv + \lambda^2 \left(v^2 - \frac{v \land w}{w^2}\right) \]

Solve the quadratic equation for \( \lambda \).

Circle passing through 2 points and tangent to a line  Find the bisector of the points and reduce the problem to finding a circle over the bisector, tangent to the line and passing through any of the two points.

Circle passing through a point and tangent to 2 lines  Find the bisector of the lines and reduce the problem to finding a circle over the bisector, tangent to any of the two lines and passing through the point.

Circle tangent to 3 lines  Find the incentre of the triangle defined by the lines by intersecting the bisectors of two pairs of them.

UVa 1029 Heliport

Statement  Given a simple polygon, find the maximum radius of a circumscribed circle.

\(^2\)ICPC 2994: World Finals 2004
**Solution**  Given a candidate circle, it is possible to check whether it lies inside the polygon by checking whether its centre does, and whether it intersects the polygon by comparing the distance from the centre to each side with the radius.

Thus, given a candidate centre, its maximum radius can be found with a binary search and the former procedure.

Now, each maximum circle is either delimited by 3 segments of the polygon or can be translated until it is. Thus, the set of candidate circles is the set of circles delimited by 3 endpoints, 2 endpoints and a line, 2 lines and an endpoint or 3 lines. Iterating through all triples \((i, j, k)\) and checking the previous cases gives the full set.

The cost of this approach is \(O(n^4 \log m)\), where \(m\) is the polygon diameter.

**Caveats**  The centre finding procedure also produces a radius that should be valid, reducing the cost to \(O(n^4)\), but because of precision issues the binary search is needed. The code is unfortunately so numerically unstable that the search interval \([0, 1024]\) where halves are always exact will work but the interval \([0, 1000]\) will not.

**Timus 1451 Beerhouse Tale**

**Statement**  Given 3 points, find a point that minimizes the sum of distances to the points.

**Solution**  The problem is to find the Fermat point of a triangle. Let \(a\) be a vertex, \(v\) and \(w\) the side vectors from \(a\) ordered so that \(v \wedge w > 0\). If any of the angles in \(a\), \(b\) or \(c\) is greater than \(2\pi/3\) return the vertex. This check translates to checking \(v \cdot w < -\|v\|\|w\|/2\) for each vertex. Otherwise, rotate \(v\) by \(-\pi/3\) and \(w\) by \(\pi/3\). \(f\) is the intersection of the lines \((a + v)c\) and \((a + w)b\).

**Caveats**  A solution using the gradient method to minimize the sum of distances is too slow.

**Notes**  Josep Grané collaborated in solving this problem.

**UVa 1081 The Return of Carl\(^3\)**

**Statement**  Two points in the surface of an octahedron are given using polar coordinates. Find the shortest distance between the points within the octahedron.

\(^3\)ICPC 4447: World Finals 2009
8 Geometry

Solution  Number the upper faces as the quadrant they are contained in and the lower faces as the quadrant plus 4. Unroll the polyhedron as shown in figure 8.2. The shortest path between a point in the 0 face and any other point is the length of the straight segment joining them, which always exists because the unrolled polygon is star-shaped. Note that triangles 2,5 and 7 have 2 possible locations and triangle 6 has 6 possible locations. They represent different paths through the octahedron. Take the minimum of all locations as the minimum distance.

Assume the first point is on the 0 face; otherwise maybe flip the octahedron through the $xy$ plane and then rotate it along the $z$ axis. This is easy using polar coordinates. Let us see how to compute its position in the plane.

Let $p$ be the result. Let $u$ be the upper vertex of the face. Let $v$ be the segment from $u$ to the base that passes through $p$. Consider the projection of the face into the $xy$ plane, so that $\theta$ is invariant (figure 8.3a). Let $l$ be the projection of $v$. $l$ is at an angle $\theta$, and $\|l\| = \sqrt{1 + \tan^2(\pi/4 - \theta)}$. Consider now the projection into the $lz$ plane, so that $\varphi$ is invariant (figure 8.3b). $p$ is at the intersection of $v$ and the line passing through $O$ with angle $\varphi$. In this projection $v$ has coordinates $(l, -\sqrt{2}/2)$. It only remains to compute $p = u + \lambda v$, where $\lambda$ is the intersection parameter, using the coordinates of the face plane: $u = (0, \sqrt{3}/2)$ and $v = (\tan(\pi/4 - \theta), -\sqrt{3}/2)$.

Regarding the second point, also transform the octahedron but record which face the point was in. After computing its position in the plane, apply the plane rotations and
translations that would map the 0 face to an unrolling of the original face and compute the plane distance.

ICPC 5887 Unchanged Picture

Statement Two lists of plot operations (draw, move) are given. Find whether the figure they produce is the same modulo translation, rotation and scaling.

Solution For each figure, first compute its representation as a set of maximal segments by merging pairs of overlapping segments. Then make the segment sets canonical respect to translation and scaling. Finally test a set of possible rotations.

To merge segments, consider them oriented so that the second endpoint is always greater than the first in lexicographic order. Then, for each pair of segments, check if they lie in the same line. If so, they do not overlap iff the first segment's second endpoint is smaller than the second segment's first endpoint or vice versa. Otherwise merge the segments into a new segment whose endpoints are the minimum of the first endpoints and the maximum of the second endpoints.

To make the segment sets canonical, first translate the segments by minus the barycentre of the endpoints. This will eliminate translations by ensuring that the barycentre is always the origin. Next scale the segments by the reciprocal of the maximum segment length. This will eliminate scalings by ensuring that the maximum segment length is always the unit.

The rotations to test are those that send a fixed segment of the first canonical figure, say the first one, to a segment of the second canonical figure. This second segment must have the same length as the first and one of its endpoints must be at the same distance from the origin than the first endpoint of the first segment.

For each feasible rotation, apply the rotation to the second canonical set, sort the new set and compare it with the first. If they are equal, the original figures are equal. If no rotation yields an equality result, the figures are distinct.

ICPC 5859 Encircling Circles

Statement A set of $n$ circles $(c_i, r_i)$ is given. Consider the union of all the disks of radius $R$ that contain the set. Find the perimeter of the union.

Solution A circle $(c_i, r_i)$ is enclosed in the disk $(d_i, R)$ iff $\|c_i - d_i\| \leq R - r_i$. Thus, the locus of the enclosing centres for a given circle is the disk $(c_i, R - r_i)$. The locus of the enclosing centres for all the circles is the intersection of these new disks. From now on we will refer to the new disks only. The curve whose perimeter we want to find is the outer curve at distance $R$ from the border of the intersection.
To compute the perimeter of the intersection, consider a disk. Intersect its boundary with all the other disks’ and sort the resulting points by angle. This will induce a list of circular segments. If a segment is contained in all the disks, then it is part of the border.

To check if a circle contains a segment, by convexity it is enough to check if it contains its midpoint.

To compute circle intersections, first discard all the circles that contain another. Then check if any two circles are disjoint. Finally, given two circles, the angles of intersection in the first circle can be computed using the cosine theorem and are

\[ \theta = \pm \arccos \left( \frac{d^2 + r_1^2 - r_2^2}{2dr_1} \right). \]

The perimeter of the intersection is the sum of all the segments, and the final result is this plus \( 2\pi R \), since it is the parallel curve at distance \( R \).

**Caveats** The UVa judge for this problem is incorrect. It will only accept solutions that output exactly 3 correct decimal digits or 0.0.

## ICPC 6035 Shortest Flight Path

### Statement

Work in a sphere of radius \( R \). A set of \( n \) points \( A \) is given. The set of valid points \( B \) is defined to be those whose distance to \( A \) is at most \( r \), \( r < R\pi/2 \). A set of queries \( s, t \in A, c \in \mathbb{R} \) is given. The set of valid paths is defined to be those contained in \( B \) so that every subpath of length \( c \) contains at least one point of \( A \). Find the shortest valid path from \( s \) to \( t \).

### Solution

The set of valid points is \( B = \bigcup_{p \in A} B(p, r) \), the union of balls (in \( S^2 \)) of radius \( r \) centred at points in \( A \). Since \( r < R\pi/2 \) the balls can be enclosed in some hemisphere and are convex in the sense that geodesics between any two points are contained in the balls.

Let us show that a shortest valid path is a piecewise geodesic. Forget the \( c \) restriction momentarily. Parts of the shortest path in \( B \) are geodesics. So are those in \( \partial B \) that belong to a single ball (but notice that this is the empty set). So a shortest valid path is a union of geodesics that join at intersections of balls.

When adding the \( c \) restriction back, it can happen that a path is no longer valid. But then, if a valid path between \( s \) and \( t \) exists, it must contain some other point in \( A \), say \( p \). So, the path is the union of a path from \( s \) to \( p \) and a path from \( p \) to \( t \). Use induction.

The solution is then to compute a graph of distances between points in \( A \) and for each query run Dijkstra’s algorithm in the graph with edges with cost greater than \( c \) removed.

Let us now focus on how to compute the distance graph. Consider another graph whose nodes are all the possible geodesic endpoints, this is the points in \( A \) and the intersections of points in ball boundaries. For each pair of points compute the geodesic between them.
8.3 Basic constructions

and whether it is contained in $B$. If so, add it as an edge. Every shortest valid path is composed of edges in this graph, so running a shortest path algorithm from each point in $A$ over this graph, say Dijkstra’s, will return the shortest valid distances between pairs of points in $A$.

Let us now discuss the problem of intersecting two different circles in $S^2$. A circle on the sphere is uniquely determined by the intersection of plane and the sphere, so the intersection points, if they exist, lie in the intersection of the planes. The intersection of the planes is either a line, the empty set or a plane. The last case cannot happen because the circles are different, and so are the planes. If the intersection is a line, it remains to intersect it with the sphere, and this can be done by solving the quadratic equation $\|p + \lambda v\|^2 = \|p\|^2 + 2\lambda \langle p, v \rangle + \lambda^2 \|v\|^2 = R^2$ in $\lambda$.

To compute the intersection points of two balls $B(p,r)$ and $B(q,r)$ use the former procedure with the planes $(\cos(r/R)p, p)$ and $(\cos(r/R)q, q)$ (see figure 8.4).

![Figure 8.4: Ball boundaries in $S^2$ are circles in $\mathbb{R}^3$](image)

It only remains to find if the geodesics are valid. Given two non-antipodal points $p$ and $q$, the geodesic through them is the shortest arc of the (maximal) circle at the plane $(O, p \wedge q)$. Each ball in $B$ may cover a segment of the arc, and we need to check whether all the arc is covered.

To clarify the following argument, consider the $\mathbb{R}^3$ basis $\langle v = q - p, w, t = p \wedge q \rangle$ with $w$ chosen so that the basis is orthogonal and positive, centred at $O$. Consider the projection of the arc and the segments into the $v$ axis. If this were a bijection, the problem would be equivalent to an interval covering, which is easily solvable using a sorted list of enter/leave events. Unfortunately it is not: points symmetric respect to the $v$ axis are projected into the same coordinate. But not everything is lost: the $pq$ arc is contained in the upper half-space, so we can ignore the lower half-space by rearranging the endpoints of covering arcs. When processing an arc $xy$, first sort it so that $\langle x \wedge y, t \rangle > 0$. If both endpoints are in the lower half-space, this is $\langle x \wedge v, t \rangle < 0$, the arc can be discarded. If $x$ is in the lower half-space, the points in $xp$ can be discarded and the arc becomes $py$. If $y$ is in the lower half-space, the arc becomes $xq$. Note that we might discard more points than there were in the original arc.

For antipodal points, either no valid geodesic exists or there are infinitely many and one of them passes through another point in $A$, so in both cases they can be discarded.

The distance computing part works with a graph of $O(n^2)$ nodes. Checking the validity of each of the $O(n^4)$ edges costs $O(n)$. Running Dijkstra’s algorithm on the graph for
each of the $n$ points in $A$ has a cost of $O(n^4 \log n)$. The queries cost $O(n^2 \log n)$ each. So the total cost is $O(n^5 + n^5 \log n + qn^2 \log n) = O(n^5 \log n + qn^2 \log n)$.

Notes This problem was not solved by any team during the 2012 ICPC World Finals. It is not conceptually difficult but very tricky to implement because many subproblems are involved.

8.4 Convex hull

Given a set of points $P$, define its convex hull $H(P)$ as the intersection of all convex sets that contain $P$. Equivalently, $H(P)$ is the set of all convex combinations of points in $P$.

The most interesting computational property of the convex hull is a convex polygon whose vertexes are a subset of $P$; as a convex polygon traversing its vertexes clockwise leaves its interior at the left hand side. This is the key idea in the Graham scan algorithm.

The most practical implementation of the convex hull algorithm is however a modification of Graham’s scan. Sort the points by $x$ coordinate. We know that the leftmost and rightmost points in $P$, $p_0$ and $p_n$, belong to the convex hull. We know that when traversing the lower hull from $p_0$ to $p_n$ in increasing $x$ order all the points in $P$ will lie to the left of the hull. So, as in Graham’s scan, we can maintain the hull of the first $p_i$ points, $q_0 = p_0 \ldots q_h = p_i$ by iteratively checking that $q_{h-2}, q_{h-1}, q_h$ turn left and removing $q_{h-1}$ if not. Computing the upper hull can be done likewise by traversing it from $p_n$ to $p_0$ in decreasing order.

As in Graham’s scan, each point is added or removed once during a traversal, so the traversal cost is $\Theta(n)$ and the algorithm cost is dominated by the $\Theta(n \log n)$ sorting phase.

The only needed operations on points are subtraction and the wedge product, so it is safe to use integer coordinates when possible. This case is when the given points already have integer coordinates whose absolute value is less than $\sqrt{2^{m-1}/2} = 2^{m/2-2}$, where $m$ is the word size in bits. Thus, for 64 bit integers, coordinates should be less than $2^{30}$, or $10^{10}$, which is weaker.

Sorting points by $x$ coordinate The $x$-coordinate order is well defined for points in the general case. However, it is ambiguous if some points share the same $x$ coordinate. To fix this, imagine that we rotate the axis an angle $\varepsilon > 0$ and use the $x'$ order instead. Points with different $x$ coordinates will still be in the same relative order, and among points with the same $x$ coordinate, those with lower $y$ coordinates will come first.

The short recipe is $p <_x q$ iff $(p_x < q_x) \lor ((p_x == q_x) \wedge (p_y < q_y))$. The $y$-coordinate order consistent with this one is $p <_y q$ iff $(p_y < q_y) \lor ((p_y == q_y) \wedge (p_x > q_x))$. 

50
8.4 Convex hull

**Timus 1185 Wall**

**Statement** Given a \( n \) vertex polygon in the plane, find the length of the shortest curve that is at distance at least \( l \) from the polygon.

**Solution** The minimal curve is the outer parallel curve at distance \( l \). This is, given the convex hull of the polygon, translate its edges by a vector outwards normal to them and modulo \( l \). Join them with circle arcs of radius \( l \) centred at the vertexes. Its length is that of the convex hull’s plus \( 2\pi l \).

**Timus 1281 River Basin**

**Statement** \( n \) lists of points are given. A list \( X_i \) is a child of \( X_j \) if the last point of \( X_i \) belongs to \( X_j \). The area of a list is defined as the area of the convex hull of its points and its descendants points. Find the maximum area of a list.

**Solution** First, for each list, compute its parent. Only lists without parents need to be considered for the maximum area, so for each list compute its maximal ancestor and assign its points to it. A straightforward \( O(n^2) \) solution is fast enough, but it could be improved to \( O(n) \) with memoization. Finally, for each expanded list, compute its convex hull and its area.

**Caveats** For \( X_i \) to be a child of \( X_j \) the final point of \( X_i \) must belong to the set of points \( X_j \), not the polygonal line induced by \( X_j \) as the original statement suggests.

**Timus 1538 Towers of Guard**

**Statement** Given \( n \) points in general position, find a convex pentagon.

**Solution** By the Erdős-Szekeres theorem, a convex \( n \)-gon exists in every collection of \( N \) points for some finite \( N(n) \) \[7\]. Furthermore, the Erdős-Szekeres conjecture states that \( N(n + 2) = 2^n + 1 \). It is known to be true up to \( n = 6 \). So \( N(5) = 9 \). Discard all but the first 9 points and sort them. Compute the convex hull of all the \( \binom{\min(n,9)}{5} \) sorted groups of 5 points. If any of them has 5 points, report it in the hull order. Otherwise, no pentagon exists.

**Solution** Alternatively, check randomly chosen groups of 5 points until a pentagon is found or a limit is reached.

**Notes** The user MSDN@Timus collaborated in solving this problem.
UVa 1084 Deer-Proof Fence

**Statement**  
$n$ points are given. Find the minimum total perimeter of a set of curves such that every point is inside a curve and at distance at least $m$ from it.

**Solution**  
Given a subset of points, the minimum perimeter of a single curve that encloses them at distance at least $m$ is the parallel curve at distance $m$, whose perimeter is that of the convex hull plus $2\pi m$.

To compute the minimum perimeter of all the points identify subsets with bit patterns and use a dynamic programming approach, where $f(X) = \min(p(X), \min_{Y \subseteq X} f(Y) + f(X - Y))$. The time complexity is $O(n \log n + 2^n)$.

ICPC 5891 A Classic Myth: Flatland Superhero

**Statement**  
$n$ points are given. Find the minimum area of a parallelogram that encloses all the points.

**Solution**  
Work on the polygon defined by the convex hull of the points. Clearly a minimum parallelogram touches 4 vertexes.

Let us see that a minimum parallelogram defined by two sides and their opposite vertexes always exists. Suppose a minimum parallelogram exists with a pair of opposite sides neither of which is defined by a polygon side. Set the base parallelogram as one of the other sides. Rotate the pair of sides along the vertex they touch. The height of the parallelogram remains constant, while the base is a linear function of the slope. Thus, the area is a linear function of the slope, and its extrema are on the boundary of its domain, which is the point where the sides intercept another vertex.

The algorithm is: for each side of the polygon, compute its furthermost vertex. This can be done in $O(n)$ time using a rotating callipers approach, but since the final algorithm will be dominated by $O(n^2)$, a brute force method is preferred for simplicity.

Now, for each pair of sides, compute the parallelogram they define. Let $(p, v)$ be a side whose opposite vertex is $q$ and let $w$ be the vector of the other side. Then the parallelogram side vector associated to $w$ is $\mu w$, with $p + \lambda v + \mu w = q$. The area of the parallelogram is the wedge product of the parallelogram side vectors.

8.5 Half-plane intersection

The half-plane intersection problem is essentially the dual of the convex hull problem. If we know a point inside the intersection $p$, then then we can reduce half-plane intersection to convex hull in the following way:

---

4ICPC 4450: World Finals 2009
For each half-plane $\pi_i$, defined by its boundary $q_i + \lambda v_i$ and oriented so that if $p \in \pi_i$ then $p$ lies at the left side of the line, represent it as $\{x : w_i^T x \geq c_i\}$, where $w_i = v_i^\perp$ and $c_i = w_i^T q_i$. Associate to it the dual point $w_i/c_i$. Compute the convex hull of the dual points. Finally compute the intersection of the lines in the convex hull and its immediate neighbours in the convex hull. \[6, 2\] give good explanations of this.

Because of precision issues, it is better to think of dual points as points in the oriented projective space. From the computational point of view, it is enough to think of the points pairs of fractions $\langle p, z \rangle$ with $z \geq 0$ and operate with them as rational numbers.

If we do not know a point inside the intersection, not even if any exists, then no straightforward reduction exists. The orientation of the half-planes respect to the interior point is essential for the aforementioned algorithm to work. Instead, a modification of the Graham scan algorithm can be used.

As in Graham’s scan sort the half-planes by normal vector’s angle. Maintain the list of half-planes that bound the intersection and add new half-planes sequentially. At each step, check if the added half-plane contains the last vertex in the boundary. If it does not, then the last half-plane in the boundary is redundant. Once all the redundant half-planes have been removed, compute the new last vertex and actually add the half-plane to the boundary.

A special check needs to be done if two consecutive half-planes normal vectors have an angle of at least $\pi$. If this happens the solution is either unbounded or empty; a way to discriminate these cases is to check whether the new half-plane contains the last vertex in the boundary.

As in Graham’s scan, each half-plane is added or removed once during a traversal, so the traversal cost is $\Theta(n)$ and the algorithm cost is dominated by the $\Theta(n \log n)$ sorting phase.

**Sorting points by angle** To avoid computing slopes, which involve divisions, or angles, which involve Taylor series, and the corresponding loss of precision, it is better to check whether the cross product of two points is positive. This approach defines an order for points in an angle range of $[0, \pi)$, but is not transitive when the whole $[0, 2\pi)$ range is considered. Thus, a patch must be used: first sort by half-plane (upper or lower), then solve ties by cross product, and solve remaining ties by squared norm. The half-plane boundaries are defined according to the angle intervals: the upper half-plane includes the $x^+$ ray, while the lower half-plane includes the $x^-$ ray.

**Timus 1681 Brother Bear’s Garden**

**Statement** The $n$ vertexes of a strictly convex polygon $A$ are given, in counter-clockwise order. The set of segments joining vertexes at distance $k$ in the list divides the plane into regions. Compute the ratio between the area of the bounded regions $B$ and the area of $A$. 
8 Geometry

Solution  Compute instead $C = A - B$, the area of the unbounded region inside $A$. $C$ is composed of subregions each associated to an edge $e = p_i p_{i+1}$ of $A$ that bounds it as well as part of $\partial B$. Moreover, these regions are, by construction, the intersection of some half-planes: those defined by $\partial B$, oriented so that they contain a point in $e$, say $(p_i + p_{i+1})/2$, and $e$.

Timus 1062 Triathlon

Statement  Given a list of $n$ integer triples $(a_i, b_i, c_i) \in \mathbb{Z}^+^3$, for each triple compute whether a triple $(x, y, z) \in \mathbb{R}^+^3$ exists such that for all $j \neq i$, $x/a_i + y/b_i + z/c_i < x/a_j + y/b_j + z/c_j$.

Solution  Rewrite the conditions as finding whether the intersection of (strict) half-spaces $x > 0$, $y > 0$, $z > 0$ and $\{\pi_j\}$ is empty, where $\pi_j = \{x \in \mathbb{R}^3, (1/a_j - 1/a_i, 1/b_j - 1/b_i, 1/c_j - 1/c_i) \cdot x > 0\}$. Since all the boundary planes contain the origin, we can project to the plane $z = 1$ and reduce the problem to the intersection of half-planes $x > 0$, $y > 0$ and $\{\eta_j\}$, with $\eta_j = \{x \in \mathbb{R}^2, (1/a_j - 1/a_i, 1/b_j - 1/b_i) \cdot x + 1/c_j - 1/c_i > 0\}$.

Solution  The decision problem for half-space intersection can also be solved with the simplex algorithm by introducing slack variables and maximizing the slackness.

Caveats  Avoid divisions by using rational coordinates. To avoid overflow with rational operations, use the double type for numerators and denominators.

8.6 Arrangements

An arrangement of a set of lines is the partition of the plane in disjoint convex cells whose boundaries are the lines.

It is not clear which is the best data structure that represents an arrangement. We will use a modification of the twin edge structure, where an edge is split into two directed edges. This allows to distinguish a unique point (its origin) and face (bounded by the edge on the left side) for each edge.

The structure contains an entry for each intersection point, which contains the point itself and a list of edges. An edge contains its direction and a pointer to the next edge in the face it bounds.

An $O(n^2)$ incremental algorithm exists for constructing an arrangement from the lines that define it. However we will use an $O(n^2 \log n)$ algorithm for simplicity.

The algorithm is to first compute all the intersection parameters for each line, sort them and make them unique. Same with intersection points. For each line consecutive
intersection parameters define a directed edge. Since points are sorted, the list index
where the edge belongs can be found in $O(\log n)$. Finally sort the edges in each list by
angle and for each edge find its opposite. The next edge in traversal order is the next
to its opposite in angle order, so pointers are set.

**UVa 1065 Raising the Roof**

**Statement**  A list of $n$ points in $\mathbb{R}^3$ and a list of $m$ triangles whose vertexes are these
points are given. Find the area of the parts visible from $z = \infty$. Triangle interiors do
not intersect.

**Solution**  Project the points into the plane $z = 0$ and build the arrangement of the
triangle edges considering them as lines instead of segments. Each of the bounded
regions is convex, its boundary does not intersect the boundary of any triangle and
belongs to at most one triangle when viewed from $z = \infty$.

So, for each region $S_i$, consider its barycentre $g$. For each triangle $T_j$, check if $g \in \pi(T_j)$,
and if so lift it to the triangle’s plane. $S_i$ belongs to the triangle with the high-
est $z$-coordinate of $g$ lifted. If it exists, then the lifted region contributes an area of
$A(S_i) \|N_j\| / \|N_{zj}\|$ where $A(S_i)$ is the area of the region and $\|N_j\| / \|N_{zj}\|$ is a correction
factor.

The correction compensates the projection distortion and is computed from the normal
of the plane a triangle belongs to. To see why it works, consider the orthogonal basis
of the plane formed by a maximum and minimum gradient unit vectors. The maximum
gradient vector is projected to a vector whose length is the same as the projection of the
normal vector to the $z$ axis. The minimum gradient vector remains unchanged.

There are at most $O(n^2)$ regions and finding the visible triangle costs $O(n)$, so the cost
of this algorithm is $O(n^3)$.

**8.7 Sweeping algorithms**

**Timus 1103 Pencils and Circles**

**Statement**  Given a list of $n$ points, no 3 collinear and no 4 cocircular, find a circle
defined by 3 points such that the number of points strictly inside the circle equals the
number of points outside or report that it does not exist.

---

5ICPC 3809: World Finals 2007
Solution For any two points \( p, q \), we claim that a third point such that the condition is satisfied always exists. Each check costs \( \Theta(n) \), and there are \( n - 2 \) candidates, so this gives a \( \Theta(n^2) \) algorithm.

Let us verify the claim. Fix two points. Consider the set of circles passing through \( p \) and \( q \) with signed radius ranging along \( \mathbb{R} \). The signed radius is defined to be the opposite of the radius iff the centre lies to the left of the line \( pq \). For \( r = -\infty \), all the points left of \( pq \) are in the circle, while for \( r = \infty \) all of them are outside. When a point is swept by this circle and changes regions, its radius is the circumradius of the three points. Since no 4 points are cocircular, different events have different radii. By continuity, the regions will be balanced at some event.

Solution This proof suggests an improvement for the algorithm to run in \( \Theta(n \log n) \) time. Fix two points. Compute the number of points left of \( pq \). Compute all the circumradii and sort them. At each event, add or remove a point depending on whether it lies right or left of \( pq \).

Notes Salvador Roura collaborated in solving this problem.

UVa 12426 Counting Triangles

Statement Given the \( n \) vertexes of a convex polygon and a constant \( K \), compute how many triangles defined by three vertexes of the polygon have area \( \leq K \).

Solution After fixing two vertexes \( i, j \), the height of the points in the interval \( (j, i) \) is a concave function, so we can find its maximum with a ternary search and then its two intersection points with \( K \) with two binary searches. This gives an \( O(n^2 \log n) \) algorithm, but it was not tested.

Solution Actually, the height function decreases monotonically when \( j \) moves from \( i + 1 \) to \( i - 2 \), while the base length \( ij \) is a concave function. An algorithm that updates the two intersection points by moving the first intersection point to the left and moving the second intersection point to the right has an amortized cost of \( O(n^2) \), improving the previous runtime (proof omitted).

Notes Salvador Roura collaborated in solving this problem.

Timus 1097 Square Country 2

Statement Given a square \( R \) of side \( L \) and cost 1 and \( n \) squares \( r_i \subset R \) of side \( l_i \) and cost \( c_i \), non overlapping, find a square \( S \subset R \) of side \( A \), such that \( \max_{i: r_i \cap S \neq \emptyset} c_i \) is minimum. Squares are parallel to the axes.
8.7 Sweeping algorithms

Solution If $S$ is an optimal solution, moving it until it hits the boundary of an external square is also optimal. Thus, we can sweep the square first horizontally and then vertically and consider only the coordinates where a square $r_i$ would touch $S$.

First sort candidate coordinates $x$ and $y$ independently. Then for each $x$ start processing events by $y$ coordinate and update the costs inside the rectangle accordingly. There are $4n^2$ events and updating the costs takes $O(\log n)$, giving a $O(n^2 \log n)$ algorithm.

Timus 1065 Frontier

Statement The list of $n$ vertexes of a convex polygon is given. $m$ interior points are given. Find the minimum perimeter of all subsets of vertexes such that the polygon they define is not degenerate and strictly contains all the interior points.

Solution If $m = 0$ the solution is the perimeter of three consecutive non-aligned vertexes (proof omitted).

Otherwise, for each pair of vertexes, compute their distance and whether the oriented edge is feasible by checking that all the interior points lie to their left. The distance calculation is a straightforward $O(n^2)$ computation. The feasibility calculation can be done in $O(mn)$ by keeping a pair of vertexes. If the edge is feasible advance the second vertex, and if it is not advance the first vertex.

Finally, for each vertex $v_s$, compute the minimum perimeter of a polygonal line starting and finishing at $v_s$ and composed of feasible edges. The computation is standard dynamic programming, with the recurrence

$$p(s, l) = \min_{i: (s+l, s+l+i) \in F} d(s + l, s + l + i) + p(s, s + l + i)$$

Timus 1556 Multishot in the Secret Cow Level

Statement A point $q$ and $n$ disks of radius $r$ are given. Fix an angle $\alpha$ and consider $k$ particles spawning from $q$ moving in the direction defined by $\alpha + \arctan(i/(k-1) - 1/2)$. If a point (not strictly) intersects a disk a score of is incremented by 1 and the point continues moving with probability $p$. If a point intersects 5 disks it always stops moving. Maximize the expected value of the score over $\alpha$.

Solution Consider each moving point as a ray and define its score as $\sum_{i=0}^{m-1} p^i$, where $m = \min(5, \{|i, c \cup r \neq \emptyset\}|)$. The angles where the score may change are those where a ray is tangent to a disk.

For each disk, compute its left and right tangent angles $\tau_0$ and $\tau_1$ respect to $q$. Add $\tau_0 + k\pi$ to an «in» list and $\tau_1 + k\pi$ to an «out» list, with $k \in \{-2, 0, 2, 4\}$ to avoid modular arithmetic. Sort the lists in $O(n \log n)$. 

57
We will now sweep $\alpha \in [0, 2\pi]$. For each ray we will maintain the index of the first in angle after the ray and the index of the first out angle not after the ray. The difference between them is the number of disks the ray intersects. Start setting $\alpha = 0$ and finding the indexes with a binary search (upper_bound and lower_bound) in $O(\log n)$. At each step find the angle a ray has to move to reach the next in angle and take the minimum as the step length. Update the indexes with a linear search in amortized cost $O(1)$.

**UVa 1077 The Sky is the Limit**

**Statement**  A list of isosceles vertical wedges is given. Compute the upper perimeter of the union.

**Solution**  Consider both wedge parts as segments. Sort the segments lexicographically by leftmost endpoint. For each segment, compute the intersections with other segments such that the new segment either turns upwards (to the left) or shares an endpoint. Sweep the plane to the right, keeping a segment index and the position inside the segment. At each step, find the intersection with the least position bigger than the current position, and update the current segment with its counterpart bigger than the current position.

**8.8 Segment intersection**

The Bentley-Ottmann algorithm for segment intersection is an instance of a sweeping line algorithm. Even though the worst case $o(n^2)$ lower bound cannot be improved, this algorithm achieves an output-sensitive $O(n + k) \log n$ upper bound.

The algorithm maintains a list of segments ordered in increasing $y$ order and traverses a set of events in increasing $x$ order. Segments can only intersect with their neighbours in the list, so this reduces the number of intersections to test.

Changes to the list happen when a new segment enters the list (insert), a segment leaves the list (erase) or some segments intersect (swap). These are the events we need to handle.

On an insert event, two new possible intersections arise: one involving the new segment and its upper neighbour and one involving the new segment and its lower neighbour.

On an erase event, a new possible intersection between the neighbours of the erased segment must be considered.

On a swap event, two new possible intersections arise: one involving the lowest segment and the highest segment’s upper neighbour and one involving the highest segment and the lowest segment’s lower neighbour. Note that more than two segments may intersect at the same point.

---

*ICPC 4127: World Finals 2008*
Since different types of event can coincide at the same point, we will represent an event as a point and the indexes of segments that must be inserted or erased. To keep the order consistent, first find all the segments that intersect and swap them in the list. Then add the segments to insert and finally remove the segments to erase.

UVa 1035 Tree-Lined Streets\textsuperscript{7}

**Statement** Given \( n \) segments in the plane, find the maximum amount of points that can be laid off inside the segments such that for each point the distance to intersections points in the same segment is at most \( d \) and the distance to other points in the same segment is at most \( e = 2d \). Endpoints belong to exactly one segment.

**Solution** Because \( e \leq 2d \), segment parts between two intersection points are independent. Compute the sorted list of intersection parameters with Bentley-Ottmann’s algorithm. For each interval \( [\lambda_i, \lambda_{i+1}] \), \( \lceil \|s\| \Delta \lambda - 2d/e \rceil \) points are added. If any intersection exists, then for the intervals \( [0, \lambda_1] \) and \( [\lambda_1, 1] \), \( \lceil \|s\| \Delta \lambda - d/e \rceil \) points are added. Otherwise, \( \lceil \|s\|/e \rceil \) points are added.

**Solution** An easier solution is to intersect all the pairs of segments and sort the intersection parameters.

### 8.9 Voronoi diagram

The Voronoi diagram of a set of points \( \{p_i\} \) is a partition of the plane into regions \( V(p_i) \) such that \( q \in V(p_i) \) iff \( d(q, p_i) = \min_j d(q, p_j) \). We will call the points sites and the regions cells.

Fortune’s algorithm is a sweep line algorithm that computes the Voronoi diagram in \( \Theta(n \log n) \). \textsuperscript{1} provides an informal but very readable explanation.

As other sweep line algorithms, Fortune’s algorithm processes events in the order they are found by a horizontal line moving downwards through the plane. At each event, an auxiliary structure called the beach line is updated. This structure is the bottom part of the union of the parabolas defined by the sites and the sweep line.

Points in a parabola are at the same distance from a site than the sweep line. In particular, they are closer to the site than any other point lying below the sweep line. So we can know the Voronoi cell of any point in the beach line.

Events are related to parabola segments entering and leaving the beach line. A new parabola segment enters when the sweep line finds a point. We will call this a site event.

\textsuperscript{7}ICPC 3000: World Finals 2004
A parabola segment disappears when it is reduced to a point, and this happens at the circumcircle of three sites. We will call this a circle event.

Let us now focus on the implementation details.

Each part of the beach line is a parabola segment, and so it will be represented by the indexes of site that defines the parabola and the sites whose parabolas bound the segment at the left and right sides. If a side is unbounded, the index -1 will be used instead. Note that a left bound does not imply that the site is actually to the left of the current site.

A segment must be able to report its endpoints when the sweep line is at a given position. To do so we will use an auxiliary function that, given two points and a horizontal line, returns the point that is equidistant from the points and the line. It is an easy but subtle subproblem.

Let $p_1$ and $p_2$ be the points. Let $r : y = w$ be the line. Let $y_i = p_i^y - w$. Assume $q$ is the solution. Let $x_1 = q^x - p_1^x$, $x_2 = p_2^x - q^x$, $k = x_1 + x_2$. Let $l = d(q, r) = d(q, p_1) = d(q, p_2)$ (see figure 8.5). It holds that

$$(y_1 - l)^2 + x_1^2 = l^2 = (y_2 - l)^2 + x_2^2$$

from where

$$x_1 = \begin{cases} 
-ky_1 \pm \sqrt{y_1 y_2 (y_2 - y_1)^2 + k^2} \\ y_1 \neq y_2 \\
\frac{k}{2} \\ y_1 = y_2 
\end{cases}$$

We must still choose the sign of the square root in the first case. Compute the two possible $x_1$ values. The sequence of appearances of the parabolas in a beach line would be $HLH$, where $H$ is the higher parabola and $L$ is the lower. So, if the left bound site is higher than the right bound site, the solution we are looking for is the left intersection, that is the smallest $x_1$. Otherwise, the solution is the biggest $x_1$.

We will store these segments in a C++ multiset of `beach` structs. The comparison function sorts segments by left endpoint and resolves ties by right endpoint. The order may change as the sweep line advances, but points where this could happen are exactly the circle events.
8.9 Voronoi diagram

The event queue must support insertion and extraction of the topmost event. So we will store them in a C++ priority queue of event structs. An event contains the coordinates where it happens and its type. The $x$ coordinate is used only to break ties. A site event also contains the index of the site that originates it. A circle event contains the parabola segment that will disappear and the point it will be contracted to.

It only remains to describe which operations happen at each event.

At a site event, an existing segment is split into two parts and a new segment is inserted in between. To find the segment to split, do a binary search with a virtual segment whose endpoints are hardcoded to be the site event’s.

In the general case, a single segment $ABC$ is found, and we will replace it by the segments $ABX$, $XBC$ and $XBC$.

If the site’s $y$ coordinate is the same as the segment’s site (e.g. points at the same height on figure 8.6a), by the construction order its $x$ coordinate is bigger, so the new segment is at the right. In this case only two segments appear: $ABX$ and $BXC$.

If the new segment would appear at the intersection of two segments, say $ABC$ and $BCD$, we will replace them by the segments $ABX$, $BXC$ and $XCD$. Additionally, a new Voronoi vertex must be added for the points $BXC$ as described below for circle events.

If the new segment would appear at the intersection of more than two segments (e.g. the lowest point on figure 8.6b), that is a site event at the same position as a circle event, process the circle event first, do as in the former case, and erase incorrectly added edges.

Alternatively, the two former cases can be avoided by applying a random rotation to the sites before the algorithm starts. No more degenerate cases exist.

The newly inserted segments may disappear in the future, so we must add new circle events for them to the queue.

At a circle event we will find all the $k$ segments that get contracted to the same point. It is possible that none exists because the candidate has been split or than more than one exists in a vertex of degree more than 3.

If $k > 0$, the contracted segments will be the $k$ consecutive triples in a string $A_0 \ldots A_{k+1}$. Expand it to the $k + 2$ segments in the string $A_{-1} \ldots A_{k+2}$. We will replace them by
two segments: $A_{-1}A_0A_{k+1}$ and $A_0A_{k+1}A_{k+2}$. Additionally we will add information to the Voronoi diagram: a new vertex and the information that an edge between cells $A_i$ and $A_{i+1}$ exists and one of its endpoints is this vertex.

Again, since we inserted new segments, we must find their possible circle events.

The function that creates circle events given 3 points computes their circumcircle $(u, r)$ and sets the coordinate of the event to $(u_x, u_y - r)$.

Finally, actual edges should be computed from the information we gathered at circle events. If two cells share an edge with two endpoints, we are done. If they share only one endpoint, then the sites are in the convex hull and the cells are infinite, so the edge is a ray in the direction of the bisector, oriented to the opposite side of the barycentre of the points respect to the line joining the two sites.

Note that the information about whether an edge exists is actually the cell adjacency graph, and so it can be used to build the Delaunay triangulation: two sites are connected by a segment iff their sites share an edge.
Figure 8.8: Delaunay triangulation of 100 randomly distributed points
UVa 1039 Simplified GSM Network

Statement A list of $B$ points $b_i$, a graph of $C$ points $c_i$ and $R$ straight line edges and a list of $Q$ queries $(c_i, c_j)$ are given. Edges are straight lines. The cost of an edge is the number of changes from a point $b_i$ to a point $b_j$ being the nearest to $x$ as $x$ moves along the edge segment. For each query, find the minimum cost of going from $c_i$ to $c_j$ in the given graph.

Solution Note that the cost of a segment is the number of Voronoi edges of \{$b_i$\} it crosses. So to build the graph compute the Voronoi diagram and the number of intersections with each edge. To answer the queries run a straightforward Dijkstra’s algorithm.

Solution An asymptotically worst but easier to implement solution is to find all the candidate crossing points for each segment, find the nearest point to the endpoints and sweep along the segment, updating the Voronoi cell at crossing points.

8.10 $k$-d tree

A $k$-dimensional tree is a data structure based on the binary tree where the comparison function depends on the depth of a node. Each node stores a $k$-dimensional point, and the comparison function at depth $h$ is $<_{h \mod k}$, where $p < q$ iff $p_d < q_d$, this is the comparison function on the $d$-th coordinates.

The graphical meaning is that each node splits the search space $S$ in two half-spaces $H^-$ and $H^+$ at a hyperplane passing through the point $H$.

To insert a point into the tree, proceed as with a regular binary tree, keeping track of the depth to choose the comparison function.

The operation a $k$-d tree is best known for is the closest point query. Given a point $q$, find the set of points $p \in T$ such that $d(q, p) = d(q, T)$.

At each node, the $q$ belongs to a half-space, say $H^+$. It is likely that $d(q, S) = d(q, H^+)$. Indeed, $d(q, H^-) \geq d(q, H)$, so a search algorithm can explore the $H^+$ branch first and if $d(q, H^+) < d(q, H)$ there is no need to explore the $H^-$ branch. The expected cost is logarithmic on the size of the tree. Note that by choosing the hyperspaces to be perpendicular to an axis, computing $d(q, H)$ turns into a trivial operation.

There exist worst-case scenarios where the cost is linear, though. One such case happens when all the points lie in the boundary of a ball.

*ICPC 3270: World Finals 2005*
8.11 Optimization

ICPC 5908 Tracking RFIDs

**Statement** A list of \( s \) points \( s_i \) not closer than \( r \), a list of \( w \) segments \( w_k \) and a list of \( p \) points \( p_j \) are given. Find the points \( s_i \) that are at a distance from \( p_j \) at most \( r \) minus the number of intersections of the segment \( s_i p_j \) and each of the \( w_k \).

**Solution** Build a \( k \)-d tree for the points \( s_i \). For each \( p_j \), query the tree for the points that are at distance at most \( r \) from \( p_j \). Note that at most 7 points can satisfy this condition for a given \( p_j \), so the cost of a query is asymptotically the same as a closest point query. For each returned \( s_i \), check if it satisfies the complex proximity condition by intersecting \( p_j s_i \) with all the segments \( w_k \).

8.11 Optimization

Timus 1332 Genie Bomber

**Statement** Given a fixed list of \( n \) disks \( c_i \) of radius \( r \) and a variable disk \( C \) of radius \( R \), find the maximum number of disks that simultaneously fit inside \( C \).

**Solution** If \( r < R \) no disk fits. Otherwise reduce the problem to finding the maximum number of points inside a disk of radius \( R - r \). The maximum is at least 1. If it is greater than 1, then consider a maximum disk. Translate it until an interior point lies on its border. Rotate it around this point until a different interior point lies on its border. Since no interior point has crossed the border, this is also a maximum disk. Therefore it is enough to test disks with two points on their border.

This gives a \( \Theta(n^3) \) algorithm to solve the problem: test all the disks that lie on the bisector of two points and are at distance \( R \) of them. Testing a disk has cost \( \Theta(n) \) and there are \( \Theta(n^2) \) such disks.

Timus 1075 Thread in a Space

**Statement** Two points \( A, B \) and a closed ball \( S(C, r) \) in \( \mathbb{R}^3 \) are given. \( A, B \notin S \). Find the minimum distance of a curve from \( A \) to \( B \) in \( \mathbb{R}^3 - S \).

**Solution** If \( AB \cap S = \emptyset \) then the solution is \( \|AB\| \). Otherwise, the minimal curve is contained in the plane \( ABC \) and is composed of two segments from \( A \) and \( B \) respectively and tangent to the circle \( ABC \cap S \), joined by an arc.

To do the actual computations, first reduce the problem to the points in \( \mathbb{R}^2 \) \( A(-x, 0), B(d - x, 0), C(0, -y) \), where \( d = \|AB\|, x = AB \cdot BC/d \) and \( y = \|AB \wedge BC\|/d \) (see
If \( y > r \) or \( A \) and \( B \) lie to the same side of the \( y \) axis, the solution is \( d \). Otherwise, let \( T \) be the tangency point to \( S \) of the line though \( A \). Then, \( \| AT \| = \sqrt{\| AC \|^2 - r^2} \), and the angle \( TCO = ACO - ACT = \arccos(y/\|AC\|) - \arccos(R/\|AC\|) \).

![Figure 8.9: Projection to the ABC plane](image)

**Timus 1170 Desert**

**Statement**  Given a set of \( n \) disjoint rectangles \( R_i \) with associated costs \( c_i \), the cost of a segment \( S \) is defined as \( c := \sum_i c_i |S \cap R_i| + c_0 |S \setminus \bigcup_i R_i| \). Find the minimum cost of a segment centred at \( O \) with length \( l \) in the first quadrant. All rectangles are (strictly) contained in the first quadrant and the disk \((O, l)\).

**Solution**  Since the cost is a piecewise linear function of the slope, the solution lies on one of the non-smooth points, and they are a subset of the angles defined by rectangle vertexes.

So, for each vertex, intersect the segment that passes through \( O \) and the vertex with all the rectangles and compute the accumulated length \( l' \) and cost \( c' \). Set \( c = c' + (l - l')c_0 \).

An easy way to compute the intersection length is to compute the intersection parameter of the segment with the sides of the rectangle (as lines), sort them, check that the low and left lines are before the up and right ones (return 0 otherwise), and return the difference between elements 3 and 2.

**8.12 Binary and ternary search**

The binary search paradigm can be used in two modes.

One is known as the bisection method, and is a numerical method to find the root of a monotonic function, or equivalently evaluate its inverse function.

The other is finding the extremal value of the parameter of a problem \( P(\alpha) \) for which its decision version is easy to solve.

Ternary search is a numerical method to find the minimum of a convex function.
UVa 1026  The Solar System\footnote{ICPC 2729: World Finals 2003}

**Statement** A point $P(t)$ moves along an ellipse centred at $O$, with axes $(a,0)$, $(0,b)$, \(a > b\). Its speed is defined according to Kepler’s law: proportional to the area it sweeps respect to the positive locus. The period is also defined according to Kepler’s law: \((T/T')^2 = (a/a')^3\). Another ellipse with axes \(a', b'\) and period \(T'\) is given. \(P(0) = (a, 0)\). Compute \(P(t_0)\).

**Solution** Compute \(T = T'\sqrt{(a/a')^3}\). Observe that \(P(t + T) = P(t)\) and \(P(T - t) = \bar{P}(t)\) and reduce the problem to \(0 \leq t_0 \leq T/2\) by taking \(t_0 \mod T\). The locus is \((c, 0)\), with \(c = \sqrt{a^2 - b^2}\). Parametrize the ellipse as \(P(\alpha) = (a \cos \alpha, b \sin \alpha)\). Then the swept area from 0 to \(\alpha\) is \(A(\alpha) = ab\alpha/2 - bc\sin\alpha/2\) (see figure 8.10), which is a monotonically increasing function. Indeed, \(A(\alpha)' = b/2(a - c \cos \alpha) > 0\). Additionally we know that \(A(T/2) = ab\pi/2\), so \(A(t_0) = ab\pi t_0/T\). Thus, \(\alpha(t_0) = A^{-1}(A(t_0))\), where \(A^{-1}\) is the inverse of the area function respect to \(\alpha\) and is computed with a binary search.

![Figure 8.10: Swept area](image)

Timus 1256  Cemetery Guard

**Statement** The distances between three points \(A, B, C\), \(r_1 = \|AB\|\), \(r_2 = \|AC\|\), \(r_3 = \|BC\|\) and a radius \(r\) are given. Find the minimum length of a closed curve with endpoints at \(A\) and whose distance from \(B\) and \(C\) is at most \(r\).

**Solution** Let \(b\) and \(c\) be the disks \((B, r)\) and \((C, r)\). The curve has to intersect \(b\) and \(c\). If \(A \in b \cap c\), this is \(r_1, r_2 \leq r\), the solution is a point so return 0. If \(A \in b\), the solution is a segment from \(A\), directed as \(AC\), and with length \(r - r_2\), so return \(2r - r_2\). Same if \(A \in c\).

If none of this happens, we can solve the triangle \(ABC\). Let \(A = (0, 0), B = (0, r_1), C = (x, y)\), with \(x = (r_1^2 + r_2^2 - r_3^2)/(2r_1)\) (cosine theorem) and \(y = \sqrt{r_2^2 - x^2}\) (Pythagoras theorem). The solution is to approach the boundary of \(b\) with a straight segment,
continue to the boundary of \( c \) with a straight segment, and return to \( A \) with a straight segment.

To compute the angle at which the curve intersects \( b \) and \( c \) we can do two nested ternary searches. Since the solution has to be inside the triangle \( ABC \), we can define the boundaries for the searches to be the angles where \( ABC \) intersects \( b \) and \( c \) respectively.

**Solution**  It is also possible to use Fermat’s principle: given that circles have negative curvature, the minimum solution satisfies the reflection principle, that is the incident angle equals the reflected angle.\(^8\). The solution could then be improved to a single binary search, but this was not tested.

**Timus 1583 Cheese**

**Statement**  A cheese is defined as a rectangular parallelepiped with \( n \) interior disjoint balls removed. A cheese of dimensions \( 10 \times 100 \times 10 \) is given. A list of planes \( y = k_i \) is constructed as follows: \( k_0 = 0 \). Each element \( k_i \) is defined as the value such that the volume of the cheese between the planes \( y = k_i - 1 \) and \( y = k_i \) is exactly 500 and then is rounded to the nearest \( 10^{-6} \)th integer. If no \( k_i \) exists, the list ends at \( k_i - 1 \). Find the list.

**Solution**  The volume of the cheese up to \( y = k \) is \( 100k \) minus the volume of some balls. For each each ball \( S_i(c, r) \), compute \( d = k - c_y \). If \( d > r \), the ball is completely inside the cheese portion and its volume \( 4\pi r^3/3 \) must be subtracted. If \( d < r \), the ball is completely outside and it must be discarded. Otherwise, the volume of the ball between \(-r \) and \( d \) must be subtracted. This volume is exactly

\[
\int_{-r}^{d} \pi r^2(y)dy = \pi \int_{-r}^{d} (r^2 - y^2)dy = \pi r^2(r + d) - \frac{\pi}{3} (r^3 + d^3)
\]

We now have a function that quickly computes \( V(k) \), so it is possible at each step to find \( k \) such that \( V(k) = V(k_{i-1}) + 500 \) up to a reasonable precision, say \( 10^{-9} \), with a binary search. Set \( k_i = 10^{-6} \cdot \lceil 10^{6}k + 1/2 \rceil \), update \( V(k_i) \) and proceed to the next step.

**UVa 1082 Conduit Packing\(^{10}\)**

**Statement**  The radii of 4 circles are given. Find the minimum radius of a circle that can enclose them so that no circle strictly intersects.

\(^{10}\text{ICPC 4448: World Finals 2009}\)
8.13 Other algorithms

Solution  An optimal solution exists with all the circles tangent to the enclosing circle and at least another one (proof omitted). Given an order and the radius $R$ of the enclosing circle, it is possible to test if the circles fit: emplace the first circle centred at $(R - r_1, 0)$ (in polar coordinates). Emplace the second at $(R - r_2, \alpha_2)$, where $\alpha_2$ is such that both circles are tangent. Emplace the third so $\alpha_3$ is tangent to either the first or the second circle, whichever gives a bigger value of $\alpha_3$ (if we took the minimum $\alpha_3$, the third circle would intersect the circle with the maximum $\alpha_3$). And so on.

Finally check that the circles would still fit if we tried to fit them again in a second round considering all the circles in the first round. That is, the new virtual circles should be at an angle at most $2\pi$ from the old real ones.

To compute $\alpha_i - \alpha_j$, use the cosine theorem on a triangle with sides $R - r_i$, $R - r_j$, $r_i + r_j$.

Iterate for all the orders and do a binary search using the former decisional procedure.

8.13 Other algorithms

Timus 1768  Circular Strings

Statement  Given a list of $n$ points, decide whether they are the (ordered) vertexes of a regular $n$-gon.

Solution  A regular $n$-gon is fully determined by its centre, radius and a rotation. Assume that the given points are indeed the vertexes of a $n$-gon and compute their barycentre $G$ and radius $r$. Fix a distinguished point $p_0$. Now validate the assumption by checking that all the points are at distance $r$ from $G$ and the angle $p_0 G p_i$ is $\pm 2\pi i / n$.

Caveats  Precision.

Timus 1444  Elephpotamus

Statement  Given a list of $n$ points $\{p_i\}$, not all of them colinear, find a non-self-intersecting polygonal line $p_i p_{j_2} \ldots p_{j_m}$ that maximizes $m$.

Solution  A line with $m = n$ always exists. Translate the points by $-p_1$. Sort the points by angle, resolving ties by modulo. The line that joins two consecutive points $p_i, p_{i+1}$ will lay between its endpoints in the defined order unless the endpoints form an angle $\alpha_i \geq \pi$. In this case, set $j_2 = i + 1$ and $j_n = i$. If no such pair exists, set $j_2 = 2$. No two pairs with $\alpha_i \geq \pi$ exist because $\sum i \alpha_i = 2\pi$, and two pairs with exactly $\alpha + i = \pi$ would mean that all the points are collinear.
Timus 1130 Nikifor’s Walk

**Statement**  Given a list of $n$ vectors of length $|v_i| \leq L$, find a map $f : [n] \to \{\pm 1\}$ such that $|\sum_{i=1}^n f(i)v_i| \leq \sqrt{2}L$ or report that none exists.

**Solution**  Given 3 vectors of length $\leq L$, it is possible to find a linear combination with coefficients $\pm 1$ of two of them with length $\leq L$. Indeed, $\|a + b\|_2^2 = \|a\|_2^2 + \|b\|_2^2 + 2ab$. Since $\|a\|_2^2, \|b\|_2^2 \leq L^2$, $\|a + b\|_2^2 \leq 2L^2(1 - \cos \alpha)$. For 6 vectors, at least one pair must have an angle $\alpha \leq \pi/3$, so $\cos \alpha \geq 1/2$ and $\|a + b\|_2^2 \leq L^2$.

An algorithm can maintain two vectors, add a third and merge two of them at each step. To keep track of the signs, consider each vector as a node in a tree that records which vectors it is built from and the sign of its right child (the left’s being always +). Once all vectors have been processed, merge the two remaining trees and traverse the resulting tree. This has a $\Theta(n)$ cost.

**Notes**  Salvador Roura collaborated in solving this problem.
9 Games

Games feature a state, a set of valid moves and two players, which alternate moves and play optimally.

9.1 Min-max

The min-max framework is useful for games where a player tries to minimize a function and its counterpart tries to maximize it. The score of an intermediate state can be defined recursively from the neighbouring states, and the problem is modelled by a graph.

Timus 1163 Chapaev

Statement A game uses $2n$ disks of the same radius, $n$ from each player, whose initial position in the plane is given. No disks are initially overlapping or tangent. A move consists of choosing a disk and an oriented direction and removing all the disks that intersect or are tangent to the disk sliding through the chosen direction, including the chosen disk. The first player that cannot move (because it has no disks left) loses the game. Find the winner.

Solution A game state is the set of alive disks and which player is expected to move. It will be represented by $2n + 1$ bits. A move is characterized by a disk and a direction. The set of disks that can potentially be erased during a move (if they were not erased yet) depends only on the initial position, so it can be represented by $2n$ bits. The set of states reachable from a given state is the bitwise and of the state and the erasable disks for each alive disk of a player and each direction.

To compute the set of directions for a disk $(P, r)$, first reduce the problem to intersection of disks of radius $2r$ and a ray spanning from $P$. Suppose we are rotating the ray along $P$. The set of intersected disks will change only at tangency points, so it is enough to choose the two tangency points for each other disk. The actual implementation uses the tangency point, a small perturbation to the left and a small perturbation to the right. To see why, imagine two tangent disks. If they both belong to the opposite player, touching the two of them is a good move. If some belongs to the current player, then a slight shift to avoid it is better.
10 Trivial

10.1 Simulations

The solution (omitted) is to do exactly what the statement says.

ICPC 5886 The Grille

Statement Decrypt a ciphertext using a turning grille cipher with a given key.

ICPC 5880 Vigenère Cipher Encryption

Statement Encrypt a plaintext using a Vigenère cipher with a given key.

Timus 1083 Factorials!!!

Statement Compute \( \prod_{0 \leq ik < n} n - ik \).

10.2 Pranks

These problems are harder to understand than to implement.

Timus 1164 Fillword

Statement Given a rectangular \( n \times m \) board filled with characters and \( p \) words, it is granted that the words can be laid out in the board using neighbouring characters (but maybe changing the direction after every character) and without characters being shared neither among different words or different positions of the same work. Output the letters that are unmapped to any word in any such layout in lexicographic order.

Solution The actual layout does not matter. Count how many characters of each type are in the board and subtract the number of characters in each word.
10 Trivial

10.3 Horrible IO

These problems have little algorithmic difficulty but many formatting quirks.

**Timus 1187 Statistical Trouble**

**Statement** Given a list of words $w_k$ and a list of queries $(i, j)$, compute the frequency table of characters $w(i)$ vs $w(j)$, including percents and marginal distributions. Percents must be rounded either up or down so that sums add up to 100%.

**Solution** The only interesting part is rounding. Given a list of integers $\{x_i\}$, first compute their sum $s$. Then compute $\{100x_i/s\}$, its sum $r$ and whether a value can be rounded: $s \nmid 100x_i$. Finally, for each value that can be rounded, add 1 to it. Stop after $100 - r$ additions.

**Timus 1274 Fractional Arithmetic**

**Statement** Compute the result of $a \star b$, with $a, b \in \mathbb{Q}$ and $\star \in \{+, -, \cdot, /\}$.

**Solution** Use the rational class over 64 bit integers plus formatting.
11 Economical analysis

This project could be developed by a professor working part-time and three students working full time. This report considers only five months of work, from January 2012 to May 2012, this is 22 weeks, and the salary costs would amount to 28160€, detailed in table 11.1.

<table>
<thead>
<tr>
<th>Salary</th>
<th>Hours</th>
<th>€/Hour$^1$</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Professor</td>
<td>440</td>
<td>34</td>
<td>14960</td>
</tr>
<tr>
<td>Student</td>
<td>2640</td>
<td>5</td>
<td>13200</td>
</tr>
<tr>
<td>Total</td>
<td>3080</td>
<td>-</td>
<td>28160</td>
</tr>
</tbody>
</table>

Table 11.1: Salary costs

$^1$ Source: 2011 UPC budget

By using a free development environment the software costs were reduced to 0.

By using publicly available problem repositories and judges, the testing costs were reduced to 0.

The infrastructure and access to books and research journals costs are considered as fixed costs of the university and are not computed in the final cost.

The final cost is therefore 28160€, detailed in table 11.2.

<table>
<thead>
<tr>
<th>Concept</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>Salaries</td>
<td>28160</td>
</tr>
<tr>
<td>Software</td>
<td>0</td>
</tr>
<tr>
<td>Testing</td>
<td>0</td>
</tr>
<tr>
<td>Total</td>
<td>28160</td>
</tr>
</tbody>
</table>

Table 11.2: Total costs
12 Conclusions

12.1 Conclusions

The UPC teams achieved an excellent result in this year edition of the ICPC: a team again qualified for the World Finals in Warsaw and it achieved the 36th position out of over 10000 teams from 2200 universities that took part in some of the contest rounds.

The set of developed algorithms and trained techniques proved to be sufficient for the problem set encountered in the world finals, so the objective finding goal of the project was successful.

All of the algorithms identified for inclusion in the library were understood, implemented conforming to the language, independence, efficiency and length restrictions and tested with positive results, so the code development goal was also successful.

12.2 Future directions

3D geometry is becoming a more important of ICPC-like contests and it would be a good direction to improve. In particular, basic constructions and intersections, data structures for polyhedra representation, a 3D convex hull algorithm and projections to a plane should be studied.

Randomized and interactive algorithms are still rarely needed, but they are sometimes a good alternative to more complex classical algorithms. In particular, randomized data structures such as the random tree or random treap, bloom filters and tailored hash functions should be studied.

Future students will likely expand this list.
Bibliography


Bibliography


