El Producte Lexicogràfic i el Producte Boustrofedònic en l'Unrank*

Xavier Molinero Albareda

Departament de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya
Campus Nord — Mòdul C6–209
Jordi Girona Salgado, 1–3
08034 Barcelona, Spain
molinero@lsi.upc.es

30 de Juny de 1998

Resum

Donada una classe d'estructures combinatòries $\mathcal{A}$, fixada una talla n i considerant un ordre total definit prèviament, la funció que ens determina l'i-èsim objecte corresponent s'anomena `unrank`. Estudiarem els avantatges i inconvenients d'utilitzar el producte boustrofedònic vers el lexicogràfic en la funció `unrank`. Veurem que tant teòrica com experimentalment, el producte boustrofedònic és considerablement millor en promig que el producte lexicogràfic.

Mots claus: Generació Determinista de Productes de Classes d'Estructures Combinatòries, Producte boustrofedònic, Producte Lexicogràfic, Funció `Unrank`, `Combstruct`.

1 Introducció

Donada una classe d'estructures combinatòries $\mathcal{A}$ [5, 6] i fixada una talla n, notarem per $\mathcal{A}_n$ el conjunt d'objectes de la classe $\mathcal{A}$ que tenen talla n. Recordem que donada una classe $\mathcal{A}$, fixada una talla n i considerant un ordre total predefinit, la funció `unrank` ens dóna l'i-èsim objecte de la classe i talla corresponents [7]. Per la seva banda, la funció `draw` [3, 4, 8] ens retorna amb probabilitat uniforme un objecte qualsevol de la classe $\mathcal{A}_n$.

En la generació aleatòria s'utilitza el producte boustrofedònic perquè el cost promig és menor que utilitzant el producte lexicogràfic [6]. En aquest treball veurem que en la funció `unrank` passa exactament el mateix. Així doncs, estudiarem les classes provinents d'un producte de talla determinada; és a dir, classes de la forma $\mathcal{A}_n = (\mathcal{B} \times \mathcal{C})_n$.

A la segona secció definim el que entenem per producte lexicogràfic i boustrofedònic i les relacions d'ordre que indueixen. Al tercer apartat presentem els algorismes d'`unrank` en ambdós ordres i calculem els costos respectius. A la quarta secció mostrem alguns resultats

*Aquest treball ha estat subvencionat per ESPRIT LTR Project no.20244 — ALCOM-IT (WP 4.5).

obtinguts experimentalment en implementar els nostres algorismes en Maple [1, 2]. A la cinquena secció considerem un pre-procés que ens permet de millorar el cost del nostre algorisme. Acabarem fent una reflexió sobre tot l'estudi realitzat a nivell teòric i pràtic.

2 Ordre Lexicogràfic i Ordre boustrofedònic

Considerem els elements de talla n d'una certa classe $\mathcal{A}$, que denotarem per $\mathcal{A}_n$, definida per un producte de dues classes, $\mathcal{A}_n = (\mathcal{B} \times \mathcal{C})_n$. Aleshores d'ordre lexicogràfic ve donat per l'isomorfisme

$$\mathcal{A}_n \cong \mathcal{B}_0 \times \mathcal{C}_n + \mathcal{B}_1 \times \mathcal{C}_{n-1} + \ldots + \mathcal{B}_n \times \mathcal{C}_0;$$

mentre que l'ordre boustrofedònic ve donat per l'isomorfisme

$$\mathcal{A}_n \cong \mathcal{B}_0 \times \mathcal{C}_n + \mathcal{B}_n \times \mathcal{C}_0 + \mathcal{B}_1 \times \mathcal{C}_{n-1} + \mathcal{B}_{n-1} \times \mathcal{C}_1 + \mathcal{B}_2 \times \mathcal{C}_{n-2} + \mathcal{B}_{n-2} \times \mathcal{C}_2 + \ldots$$

Conseqüentment, si tenim dos elements d'aquesta classe,

$$\alpha_1 = (\beta_1, \gamma_1),\ \ \alpha_2 = (\beta_2, \gamma_2)\ \in\ \mathcal{A}_n$$

en l'ordre lexicogràfic tindrem que

$$\begin{array}{lll} \alpha_1 < \alpha_2 & \iff & \beta_1 < \beta_2 \lor \\ & & (\beta_1 = \beta_2\ \land\ \gamma_1 < \gamma_2) \end{array}$$

mentre que l'ordre boustrofedònic será més complexe perquè s'han de tractar les talles de les quatre components

$$\begin{array}{lll} \alpha_1 < \alpha_2 & \iff & \min\{|\beta_1|, |\gamma_1|\} < \min\{|\beta_2|, |\gamma_2|\} \lor \\ & & (\min\{|\beta_1|, |\gamma_1|\} = \min\{|\beta_2|, |\gamma_2|\}\ \land\ \beta_1 < \beta_2)\ \lor \\ & & (\min\{|\beta_1|, |\gamma_1|\} = \min\{|\beta_2|, |\gamma_2|\}\ \land\ \beta_1 = \beta_2\ \land\ \gamma_1 < \gamma_2) \end{array}$$

Per tant, en principi el producte boustrofedònic serà més complexe de tractar (en el sentit de l'ordenació) pel fet que cal considerar les talles de cadascuna de les quatre components.

3 Algorismes i Costos

L'algorisme de l'`unrank` en el producte lexicogràfic és prou senzill i conegut [7]. El procés per a calcular l'i-èsim objecte de la classe $\mathcal{A}_n = (\mathcal{B} \times \mathcal{C})_n$ ve donat per

function *UnrankLexicograficProduct*$(Prod(\mathcal{B}, \mathcal{C}), i)$ **is**
  find j *such that* $\sum_{l=0}^{j-1} b_l \cdot c_{n-l} < i + 1 \leq \sum_{l=0}^{j} b_l \cdot c_{n-l}$
  $i := i - \sum_{l=0}^{j-1} b_l \cdot c_{n-l}$
  $ii := i$ **div** c_{n-j}
  $r := i$ **mod** c_{n-j}
  return $\langle unrank(\mathcal{B}, ii), unrank(\mathcal{C}, r)\rangle$
end

Tractem ara el producte boustrofedònic. Definim amb la variable $\rho_{n,j}$ les sumes parcials de les cardinalitats en l'ordre boustrofedònic; és a dir, $\rho_{n,0} = b_0 \cdot c_n$, $\rho_{n,1} = \rho_{n,0} + b_n \cdot c_0$, $\rho_{n,2} = \rho_{n,1} + b_1 \cdot c_{n-1}$, $\rho_{n,3} = \rho_{n,2} + b_{n-1} \cdot c_1$, ... Ara ja podem implementar l'algorisme

```
function UnrankBoustrofedonicProduct(Prod(B, C), i) is
    find j such that ρ_{n,j-1} < i + 1 ≤ ρ_{n,j}
    i := i − ρ_{n,j-1}
    (* We assume ρ_{n,j} = ρ_{n,j-1} + b_k · c_{n-k} *)
    ii := i div c_{n-k}
    r := i mod c_{n-k}
    return ⟨unrank(B, ii), unrank(C, r)⟩
end
```

Estudiem el cost total, comptant el nombre de comparacions i operacions aritmètiques necessàries. Tanmateix, per simplificar la notació, considerarem que la nostra talla n és un nombre parell. Denotem per $\Gamma_L \mathcal{A}_n$ el cost de calcular la talla de la primera component en el producte lexicogràfic en tots els objectes d'una determinada classe $\mathcal{A}_n = (\mathcal{B} \cdot \mathcal{C})_n$. El fet que calculem aquesta talla seqüencialment ens permet obtenir directament el seu valor,

$$\Gamma_L \mathcal{A}_n = \sum_{j=0}^{n} (j+1) \cdot b_j \cdot c_{n-j} \sim \sum_{j=0}^{n} j \cdot b_j \cdot c_{n-j}$$

De forma anàloga, denotem el cost mitjà en el cas boustrofedònic per $\Gamma_B \mathcal{A}_n$,

$$\begin{aligned}
\Gamma_B \mathcal{A}_n &= \sum_{j=0}^{\lfloor n/2 \rfloor} (2j+1) \cdot b_j \cdot c_{n-j} + \sum_{j=0}^{\lfloor n/2 \rfloor - 1} 2(j+1) \cdot b_{n-j} \cdot c_j + \sum_{j=0}^{n} 1 \cdot b_{n-j} \cdot c_j \\
&= 2 \cdot \sum_{j=0}^{n} (j+1) \cdot b_j \cdot c_{n-j} + 2 \cdot \sum_{j=n/2+1}^{n} (n - 2j + \tfrac{1}{2}) \cdot b_j \cdot c_{n-j} \\
&= 2 \cdot \sum_{j=0}^{n} (j+1) \cdot b_j \cdot c_{n-j} - 4 \cdot \sum_{j=0}^{n/2-1} (j + \tfrac{3}{4}) \cdot b_{j+n/2+1} \cdot c_{n/2-j-1}
\end{aligned}$$

on la tercera suma de la primera igualtat ve donada per un cost addicional per determinar si l'element que busquem és de la forma $b_j \cdot c_{n-j}$ o $b_{n-j} \cdot c_j$.

Tot seguit anem a estudiar assimptòticament els avantatges o inconvenients que s'esdevenen quan utilitzem l'ordre lexicogràfic o bé l'ordre boustrofedònic. Per fer-ho considerem el següent *cost diferència*,

$$\Gamma_{L,B} \mathcal{A}_n \;:=\; \Gamma \mathcal{A}_n \;\equiv\; \Gamma_L \mathcal{A}_n - \Gamma_B \mathcal{A}_n.$$

Amb l'ajud d'un programa algebraic (com el MAPLE), podem emplenar la següent taula, tot suposant que $\mathcal{A}_n = (\mathcal{B} \times \mathcal{C})_n$,

$\Gamma\mathcal{A}_n$	$\mathcal{B}$ és polinòmic $(b_j \sim j^b)$
$\mathcal{C}$ és polinòmic $(c_j \sim j^c)$	n **parell** $c \gg b \Rightarrow \Gamma\mathcal{A}_n \sim -2(n-1)^c$ $b = c \Rightarrow \Gamma\mathcal{A}_n \sim (n-7)\cdot(n-1)^b$ $c \ll b \Rightarrow \Gamma\mathcal{A}_n \sim (n-5)\cdot(n-1)^b$ n **senar** $c \gg b \Rightarrow \Gamma\mathcal{A}_n \sim -2\cdot n^c$ $b = c \Rightarrow \Gamma\mathcal{A}_n \sim \left(-\frac{1}{2}\right)^c\cdot(2n+1)\cdot\left(n+\frac{1}{2}\right)^b + \left(\frac{1}{2}\right)^c\cdot(2n-3)\cdot\left(n-\frac{1}{2}\right)^b$ $c \ll b \Rightarrow \Gamma\mathcal{A}_n \sim \left(-\frac{1}{2}\right)^c\cdot(2n+1)\cdot\left(n+\frac{1}{2}\right)^b + \left(\frac{1}{2}\right)^c\cdot(2n-3)\cdot\left(n-\frac{1}{2}\right)^b$
$\mathcal{C}$ és exponencial $(c_j \sim \gamma^j$, on $\gamma > 1)$	$\Gamma\mathcal{A}_n \sim -2\cdot c^{n-1}$
$\mathcal{C}$ és super-exponencial $(c_j \sim j!)$	n **parell** $\Gamma\mathcal{A}_n \sim (n-2)\cdot n^b$ n **senar** $\Gamma\mathcal{A}_n \sim (2n+1)\cdot\left(n+\frac{1}{2}\right)^b\cdot\sqrt{\pi}$

$\Gamma\mathcal{A}_n$	$\mathcal{B}$ és exponencial $(b_j \sim \beta^j$, on $\beta > 1)$
$\mathcal{C}$ és exponencial $(c_j \sim \gamma^j$, on $\gamma > 1)$	n **parell** $c \gg b \Rightarrow \Gamma\mathcal{A}_n \sim -c^n$ $b = c \Rightarrow \Gamma\mathcal{A}_n \sim (n-3)\cdot c^n$ $c \ll b \Rightarrow \Gamma\mathcal{A}_n \sim (n-2)\cdot b^n$ n **senar** $c \gg b \Rightarrow \Gamma\mathcal{A}_n \sim -c^n$ $b = c \Rightarrow \Gamma\mathcal{A}_n \sim 2n\cdot b^n$ $c \ll b \Rightarrow \Gamma\mathcal{A}_n \sim (2n+1)\cdot b^{n+1/2}\cdot c^{-1/2}$
$\mathcal{C}$ és super-exponencial $(c_j \sim j!)$	n **parell** $\Gamma\mathcal{A}_n \sim (n-2)\cdot b^n$ n **senar** $\Gamma\mathcal{A}_n \sim \left(\sqrt{b\pi}\cdot n + (\sqrt{b\pi}-1)\cdot(n-1)\right)\cdot b^n$

$\Gamma\mathcal{A}_n$	$\mathcal{B}$ és super-exponencial $(b_j \sim j!)$
$\mathcal{C}$ és super-exponencial $(c_j \sim j!)$	n **parell** $\Gamma\mathcal{A}_n \sim n \cdot a_n$ n **senar** $\Gamma\mathcal{A}_n \sim n^{3/2} \cdot \sqrt{\pi} \cdot a_n$

Per qualsevol aclariment sobre els càlculs assimptòtics o per veure els gràfics experimentals que verifiquen aquestes dades, només cal demanar-los a l'autor del present article.

La conclusió principal que en podem extreure d'aquesta taula és el fet que en general el *cost promig* en el cas del producte lexicogràfic és superior que en el cas boustrofedònic tot i que no augmenti l'ordre. Així doncs, la utilització de l'ordre boustrofedònic en el producte pot fer disminuir el cost.

4 Resultats experimentals

Tots els algorismes i experiments d'aquesta secció s'han fet en MapleV (Release 4.00a) amb una SUN-ULTRA-2. La implementació detallada dels algoritmes es pot demanar a l'autor.

Tot seguit anem a veure quins resultats en podem extreure experimentalment.

Considerem les classe dels arbres binaris, $\mathcal{B} = Union(Z, Prod(\mathcal{B}, \mathcal{B}))$, i la classe dels arbres binaris-triaris, $\mathcal{M} = Union(Z, Union(Prod(\mathcal{M}, \mathcal{M})), Prod(\mathcal{M}, Prod(\mathcal{M}, \mathcal{M}))))$. El gràfic 1 represesenta el resultat d'aplicar l'algorisme del producte lexicogràfic i boustrofedònic sobre la classe $Prod(\mathcal{B}, \mathcal{B})$. L'eix d'abscisses indica la talla tractada, mentre que l'eix d'ordenades indica el cost mitjà (en segons) obtingut després de fer 1000 iteracions en un `unrank` triat a l'atzar amb probabilitat uniforme tant pel cas lexicogràfic (línia superior) com pel cas boustrofedònic (línia inferior). Els gràfics 2 i 3 són anàlegs al gràfic 1 però considerant les classes $Prod(\mathcal{B}, \mathcal{M})$ i $Prod(\mathcal{M}, \mathcal{B})$, respectivament.

Tanmateix, recordant que

$$M_n \sim \frac{3^{n+1/2}}{2\sqrt{\pi n}} \quad \text{i} \quad B_n = \frac{1}{n+1}\binom{2n}{n} \sim \frac{4^n}{(n+1)\cdot\sqrt{\pi n}},$$

podem comprovar que el *promig calculat teòricament del cost diferència* també ens diu que és millor utilitzar el producte boustrofedònic vers el lexicogràfic en el primer i el darrer exemple (diferenixen en una quantitat lineal). Per la seva banda, en el segon exemple l'ús del producte lexicogràfic és lleugerament millor que el boustrofedònic (únicament difereixen en una constant).

L'autor té uns gràfics experimentals que corroboren aquestes dades. Si algú hi està interessat només l'hi ha de demanar.

5 Utilització de Taules

En tot el treball presentat fins al moment hem suposat que fixada una classe $\mathcal{A}_n$ la seva cardinalitat ens ve donada. Ara bé, el càlcul d'aquesta cardinalitat requereix un cost

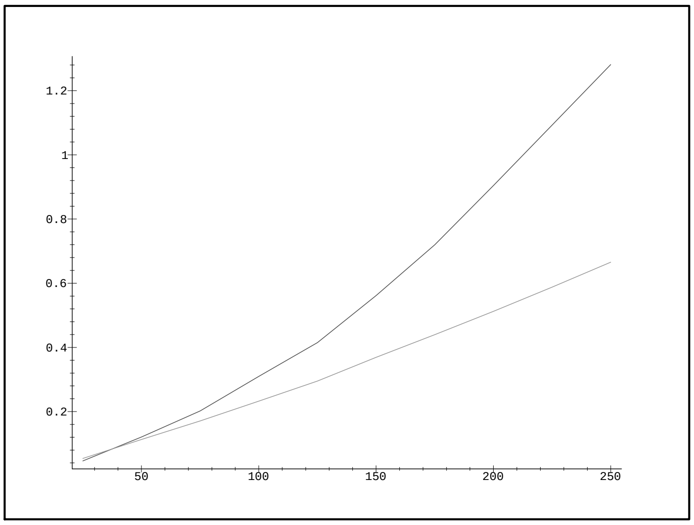

Figura 1: Considerem la classe $Prod(\mathcal{B}, \mathcal{B})$, on $\mathcal{B}$ és la classe dels arbres binaris definida per l'equació $\mathcal{B} = Union(Z, Prod(\mathcal{B}, \mathcal{B}))$. L'eix d'abscisses indica la talla considerada, mentre que l'eix d'ordenades indica el cost mitjà (espressat en segons) de fer l'`unrank` en l'esmentada classe pel cas lexicogràfic (línia superior) i boustrofedònic (línia inferior). S'han considerat 1000 iteracions de l'`unrank` triades a l'atzar amb probabilitat uniforme.

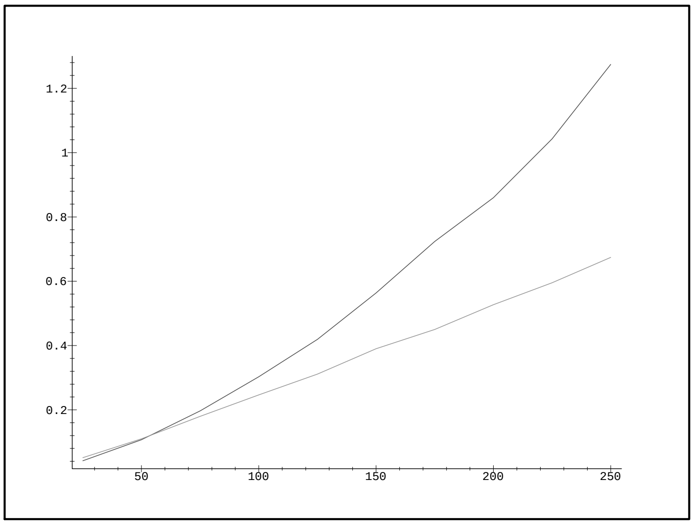

Figura 2: Considerem la classe $Prod(\mathcal{B}, \mathcal{M})$, on $\mathcal{B}$ és la classe dels arbres binaris definida per l'equació $\mathcal{B} = Union(Z, Prod(\mathcal{B}, \mathcal{B}))$ i $\mathcal{M}$ és la classe dels arbres unaris-binaris, $\mathcal{M} = Union(Z, Union(Prod(\mathcal{M}, \mathcal{M})), Prod(\mathcal{M}, Prod(\mathcal{M}, \mathcal{M}))))$. L'eix d'abscisses indica la talla considerada, mentre que l'eix d'ordenades indica el cost mitjà (espressat en segons) de fer l'`unrank` en l'esmentada classe pel cas lexicogràfic (línia superior) i boustrofedònic (línia inferior). S'han considerat 1000 iteracions de l'`unrank` triades a l'atzar amb probabilitat uniforme.

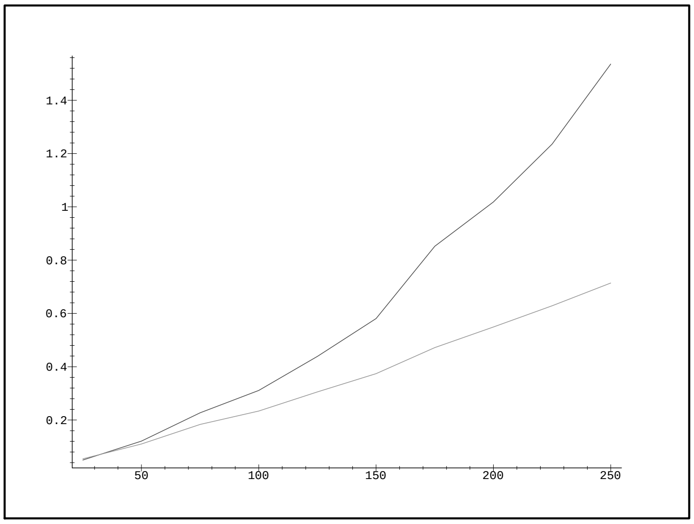

Figura 3: Considerem la classe $Prod(\mathcal{M}, \mathcal{B})$, on $\mathcal{M}$ és la classe dels arbres unaris-binaris, $\mathcal{M} = Union(Z, Union(Prod(\mathcal{M}, \mathcal{M})), Prod(\mathcal{M}, Prod(\mathcal{M}, \mathcal{M}))))$, i $\mathcal{B}$ és la classe dels arbres binaris definida per l'equació $\mathcal{B} = Union(Z, Prod(\mathcal{B}, \mathcal{B}))$. L'eix d'abscisses indica la talla considerada, mentre que l'eix d'ordenades indica el cost mitjà (espressat en segons) de fer l'`unrank` en l'esmentada classe pel cas lexicogràfic (línia superior) i boustrofedònic (línia inferior). S'han considerat 1000 iteracions de l'`unrank` triades a l'atzar amb probabilitat uniforme.

addicional que caldria tenir present.

Per tal d'evitar aquest cost addicional, el que podem fer és emmagatzemar en unes taules les sumes parcials que es requereixen i després fer una cerca seqüencial sobre aquestes taules. En el cas del producte lexicogràfic de la classe $\mathcal{A} = Prod(\mathcal{B}, \mathcal{C})$ tindríem la taula

$$\begin{aligned} \pi_{n,0} &= b_0 \cdot c_n; \\ \pi_{n,j} &= \pi_{n,j-1} + b_j \cdot c_{n-j}, \quad \text{per a tot } 1 \leq j \leq n \end{aligned}$$

Així la taula esmentada només l'haurem de calcular una vegada i la podem considerar com un pre-procés del nostre algorisme.

Però l'inconvenient és que cal fer taules per a totes les talles i també per a totes les classes, la qual cosa ens porta a un pre-procés molt costós i a un requeriment considerable de memòria. Si tenim k classes del tipus producte i volem computar elements de talla n, aleshores el pre-procés requereix un cost i una memòria de l'ordre de $\mathcal{O}(k \cdot n^2)$.

L'algorisme implementat en Maple pel producte lexicogràfic el podem fer emmagatzemant aquestes dades en una simple taula global i modificant sensiblement el producte lexicogràfic presentat anteriorment. La implementació d'aquests algorismes, com tots els anteriors, es pot demanar a l'autor.

Tot seguit, en detallem alguns resultats experimentals obtinguts.

Tornem a considerar la classe dels arbres binaris. La Figura 4 tracta objectes de talla 100. L'eix d'abscisses mostra l'execució de l'`unrank` per un objecte triat a l'atzar entre 0 i $a_{100} - 1$ amb probabilitat uniforme i l'eix d'ordenades mostra el seu cost. La corba amb menys ordenades correspon a l'algorisme amb el producte lexicogràfic utilitzant un pre-procés d'emmagatzemar les dades en una taula addicional. La corba del mig correspon

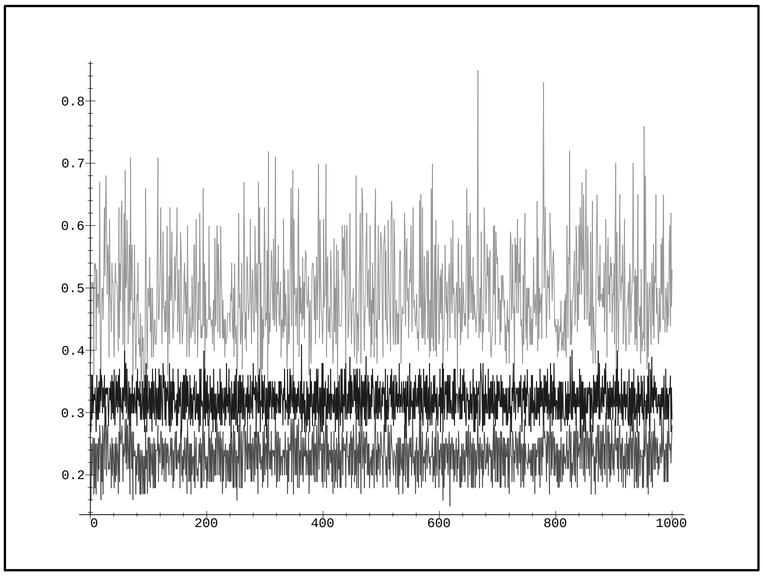

Figura 4: Gràfic en fer 1000 iteracions de l'`unrank` amb pre-procés de les taules (corba inferior) i sense aquest pre-procés (corba superior), i amb la rutina `draw` (corba del mig). Totes les execucions s'han fet per objectes de talla 100 de la classe $\mathcal{A} = Prod(\mathcal{B}, \mathcal{B})$, on $\mathcal{B} = Union(Z, Prod(\mathcal{B}, \mathcal{B}))$

a l'execució del `draw` (rutina de la llibreria `combstruct` en la qual es genera aleatòriament un element d'una determinada classe amb una talla prefixada). La corba que mostra el cost màxim correspon a l'algorisme presentat a la secció anterior on s'utilitzava el producte en ordre lexicogràfic sense emmagatzemar les dades en cap taula. La Figura 5 considera la mateixa classe però amb objectes de talla $n = 300$. En aquest cas la corba inferior correspon a la rutina `draw` del Maple, mentre que la corba que es troba lleugerament a sobre d'aquesta correspon al producte lexicogràfic amb pre-procés d'emmagatzematge de dades, la corba superior és la de l'ordre lexicogràfic sense pre-procés. Finalment, la Figure 6 correspon a objectes de talla $n = 500$. La corba inferior fa referència al `draw`, la corba del mig és de l'algorisme que emmagatzema les dades prèviament, mentre que la corba superior correspon a l'algorisme lexicogràfic que no calcula una taula amb les sumes parcials. D'aquesta manera, podem concloure que efectivament la utilització del pre-procés és considerablement beneficiós sempre i quan requerim l'execució de l'`unrank` un nombre de vegades suficientment gran per contrarestar el cost addicional en el càlcul de les taules.

Tanmateix, si utilitzem taules de pre-procés encara millorarem més l'eficiència si fem una cerca dicotòmica en lloc de seqüencial. Així el cost de trobar la primera component en el cas del producte serà logarítmic. Però encara podem millorar el cost teòricament si utilitzem l'ordre boustrofedònic amb taules.

Malgrat tot, resultaria millor considerar el producte boustrofedònic i emmagatzemar les sumes parcials en una taula d'una manera estratègica per poder reduir al màxim el cost en promig. Per poder-ho fer considerem els valors

$$\pi_{n,i} = b_i \cdot c_{n-i}.$$

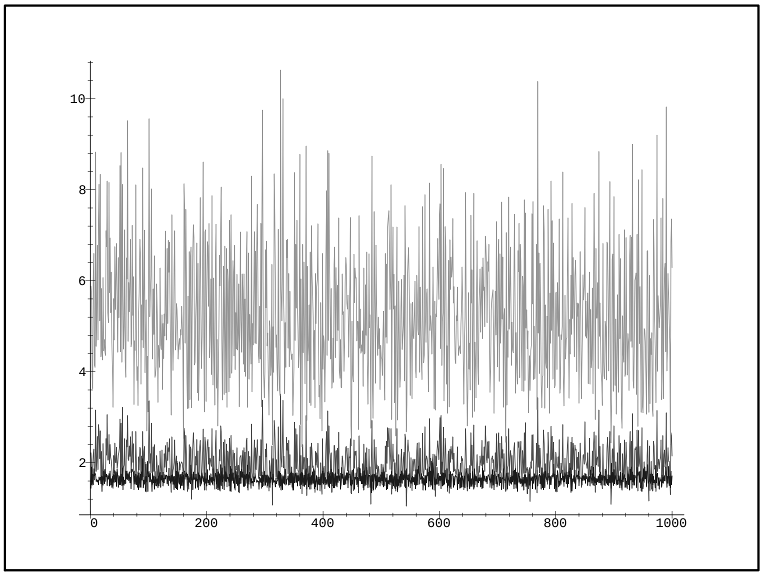

Figura 5: Gràfic en fer 1000 iteracions de l'`unrank` amb pre-procés de les taules i sense aquest pre-procés (corbes superiors solapades), i amb la rutina `draw` (corba inferior). Totes les execussions s'han fet per objectes de talla 300 de la classe $\mathcal{A} = Prod(\mathcal{B}, \mathcal{B})$, on $\mathcal{B} = Union(Z, Prod(\mathcal{B}, \mathcal{B}))$

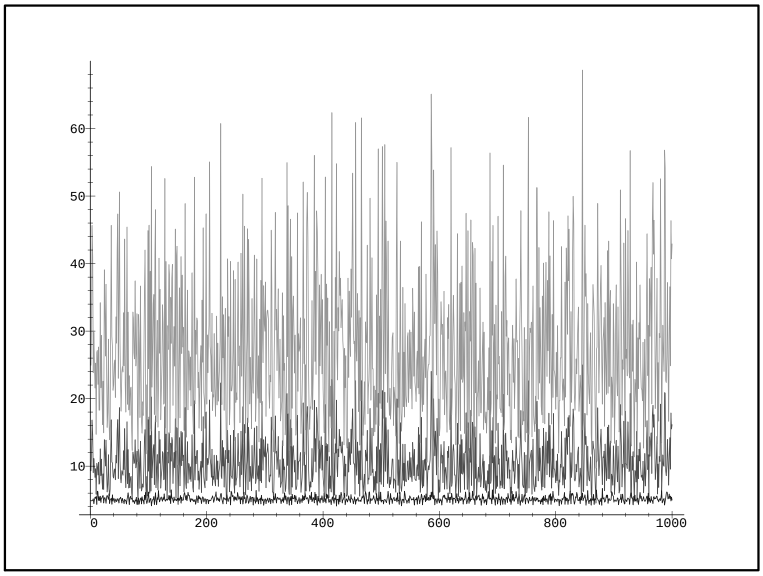

Figura 6: Gràfic en fer 1000 iteracions de l'`unrank` amb pre-procés de les taules (corba del mig) i sense aquest pre-procés (corba superior), i amb la rutina `draw` (corba inferior). Totes les execucions s'han fet per objectes de talla 500 de la classe $\mathcal{A} = Prod(\mathcal{B}, \mathcal{B})$, on $\mathcal{B} = Union(Z, Prod(\mathcal{B}, \mathcal{B}))$

Suposem que $n = 2^r$ (en cas de no ser cert podem prendre $m = 2^r$, on $r = \lfloor \log_2(n) \rfloor$) i anem guardant aquests valors en una taula ρ seguint la següent seqüència,

$$\begin{array}{l}
\rho'[\frac{n}{2} - 1] := \pi_{n,0} \\
\rho'[\frac{n}{2} + 1] := \pi_{n,0} \\
\\
\rho'[\frac{n}{4}] := \pi_{n,1} \\
\rho'[\frac{3n}{4}] := \pi_{n,n-1} \\
\\
\rho'[\frac{n}{8}] := \pi_{n,2} \\
\rho'[\frac{3n}{8}] := \pi_{n,n-2} \\
\rho'[\frac{5n}{8}] := \pi_{n,3} \\
\rho'[\frac{7n}{8}] := \pi_{n,n-3} \\
\ldots
\end{array}$$

Després considerem la taula ρ definida per

$$\begin{array}{rcl}
\rho_{n,0} & = & \rho'[1]; \\
\rho_{n,j} & = & \rho_{n,j-1} + \rho'[j], \quad \text{per a tot } 1 \leq j \leq n.
\end{array}$$

D'aquesta manera aconseguim saber que la primera component

té talla 0 o n amb 2 passos,
té talla 1 o $n-1$ amb 3 passos,
té talla 2 o $n-2$ o 3 o $n-3$ amb 4 passos,
té talla 4 o $n-4$ o 5 o $n-5$ o 6 o $n-6$ o 7 o $n-7$ amb 5 passos,
...

I com que en general el nombre d'elements de talla major en una mateixa classe és considerablement més gran, en promig podrem saber la talla de la primera component d'una manera més eficient (ràpida) que en els casos anteriors.

6 Conclusió

És conegut que la utilització del producte boustrofedònic en la funció `draw` vers el producte lexicogràfic millora el cost promig (bàsicament el redueix a la meitat tot i que no en disminueix l'ordre). En aquest treball hem comprovat que, tant analíticament com experimentalment, el producte boustrofedònic també té un cost promig sensiblement inferior (en general) que el producte lexicogràfic en la funció `unrank`.

D'altra banda, en el cas que es vulgui aplicar reiteradament la funció `unrank` dins d'una mateixa talla en una determinada classe, fóra convenient emmagatzemar en unes taules les sumes parcials que ens proporcionen la talla de la primera component. D'aquesta manera podríem aplicar una cerca dicotòmica per reduir substancialment el cost promig.

Agraïments

Agraeixo l'ajut d'en Conrado Martínez, tant pel seu suport i els seus suggeriments en l'estudi d'aquest treball, com pel seguiment i interés per a dur-lo a terme.

Referències

[1] Bruce W. Char: *Maple V library reference manual.* Pre-publication manuscript, Waterloo Maple 1991

[2] P. Dumas, X. Gourdon: *Une introduction à Maple.* Tech.Rep.num. 173, June 1995, INRIA.

[3] P. Flajolet and B. Salvy: *Computer Algebra Libraries for Combinatorial Structures.* INRIA Res.Rep.num. 2497, March 1995.

[4] P. Flajolet, B. Salvy and P. Zimmermann: *Lambda-Upsilon-Omega: The 1989 Cookbook.* Res.Rep.num. 1073, Institut National de Recherche en Informatique et en Automatique.

[5] P. Flajolet and R. Sedgewick: *The average case analysis of algorithms: Counting and generating functions.* INRIA Tech.Rep.num. 1888, April 1993 .

[6] P. Flajolet, P. Zimmermann and B. Van Cutsem: *A calculus for the random generation of combinatorial structures.* Submitted to TCS, 1993. Also available as Inria Res.Rep.num. 1830.

[7] X. Molinero: *Unranking of combinatorial structures.* Tech.Rep. LSI-97-30-R, 1997.

[8] P. Zimmermann: *Gaïa: package for the random generation of combinatorial structures. MapleTech 1*, 1 (1994), 38-46.