Proiectarea algoritmilor
Lucrare de laborator nr. 5
Paradigma *greedy*
Arbori binari ponderaţi pe frontieră

Cuprins

1 Arbori binari ponderaţi pe frontieră **1**

1.1 Descriere . . . 1

1.2 Algoritm pentru construirea unui arbore binar cu lungimea externă ponderată minimă 2

1.3 Implementarea algoritmului pentru construirea unui arbore binar cu lungimea externă ponderată minimă . . . 2

2 Sarcini de lucru **2**

1 Arbori binari ponderaţi pe frontieră

1.1 Descriere

Considerăm arbori binari cu proprietatea că orice vârf are 0 sau 2 succesori şi vârfurile de pe frontieră au ca informaţii (etichete, ponderi) numere, notate cu $\mathit{info}(v)$. Convenim să numim aceşti arbori ca fiind *ponderaţi pe frontieră*. Pentru un vârf v din arborele t notăm cu d_v lungimea drumului de la rădăcina lui t la vârful v. *Lungimea externă ponderată* a arborelui t este:

$$\mathrm{LEP}(t) = \sum_{v \text{ pe frontiera lui } t} d_v \cdot \mathit{info}(v)$$

Modificăm aceşti arbori etichetând vârfurile interne cu numere ce reprezintă suma etichetelor din cele două vârfuri fii. Pentru orice vârf intern v avem $\mathit{info}(v) = \mathit{info}(v_1) + \mathit{info}(v_2)$, unde v_1, v_2 sunt fiii lui v (Figura 1).

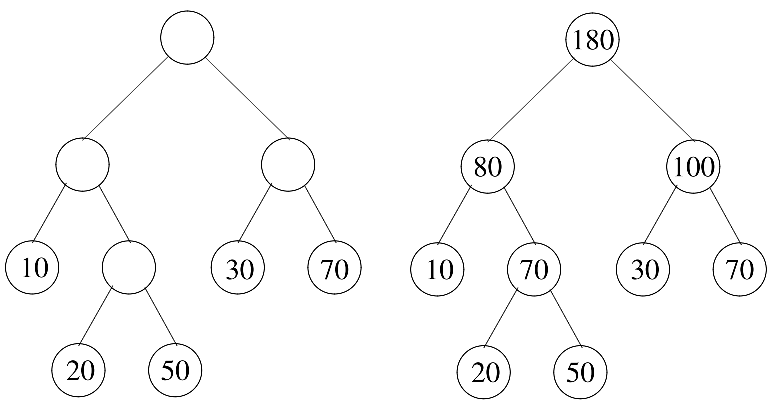

Figura 1: Arbore binar ponderat pe frontieră, înainte şi după modificare

Lema 1.1 *Fie t un arbore binar ponderat pe frontieră. Atunci:*

$$\mathrm{LEP}(t) = \sum_{v \text{ intern în } t} info(v)$$

Lema 1.2 *Fie t un arbore din $\mathscr{T}(x)$ cu LEP minimă şi v_1, v_2 două vârfuri pe frontiera lui t. Dacă $info(v_1) < info(v_2)$ atunci $d_{v_1} \geq d_{v_2}$.*

Lema 1.3 *Presupunem $x_0 \leq x_1 \leq \cdots \leq x_{n-1}$. Există un arbore în $\mathscr{T}(x)$ cu LEP minimă şi în care vârfurile etichetate cu x_0 şi x_1 (vârfurile sunt situate pe frontieră) sunt fraţi.*

1.2 Algoritm pentru construirea unui arbore binar cu lungimea externă ponderată minimă

Ideea algoritmului rezultă direct din Lema 1.3. Presupunem $x_0 \leq x_1 \leq \cdots \leq x_{n-1}$. Ştim că există un arbore optim t în care x_0 şi x_1 sunt memorate în vârfuri frate. Tatăl celor două vârfuri va memora $x_0 + x_1$. Prin ştergerea celor două vârfuri ce memorează x_0 şi x_1 se obţine un arbore t'. Fie $t1'$ un arbore optim pentru secvenţa $y = (x_0 + x_1, x_2, \ldots, x_{n-1})$ şi $t1$ arborele obţinut din $t1'$ prin ”agăţarea” a două vârfuri cu informaţiile x_0 şi x_1 de vârful ce memorează $x_0 + x_1$. Avem $\mathrm{LEP}(t1') \leq \mathrm{LEP}(t')$ ce implică $\mathrm{LEP}(t1) = \mathrm{LEP}(t1') + x_0 + x_1 \leq \mathrm{LEP}(t') + x_0 + x_1 = \mathrm{LEP}(t)$. Cum t este optim, rezultă $\mathrm{LEP}(t1) = \mathrm{LEP}(t)$ şi de aici t' este optim pentru secvenţa y. Considerăm în loc de secvenţe de numere secvenţe de arbori.
Notaţii: $t(x_i)$ = arborele format dintr-un singur vârf etichetat cu x_i si $rad(t)$ = rădăcina arborelui t.
Premise: Iniţial se consideră n arbori cu un singur vârf, care memorează numerele $x_i, i = 0, \ldots, n-1$.

```
procedure lep(x, n)
   1:  B ← {t(x_0),...,t(x_{n-1})}
   2:  while (#B > 1) do
   3:      alege t1, t2 din B cu info(rad(t1)),info(rad(t2)) minime
   4:      construieşte arborele t în care subarborii rădăcinii
   5:        sunt t1, t2 şi info(rad(t))=info(rad(t1))+info(rad(t2))
   6:      B ← (B \ {t1,t2}) ∪ {t}
end
```

1.3 Implementarea algoritmului pentru construirea unui arbore binar cu lungimea externă ponderată minimă

a) Dacă mulţimea B este implementată printr-o listă liniară, atunci în cazul cel mai nefavorabil operaţia 3 este are timpul de execuţie $O(n)$, iar operaţia 6 are timpul de execuţie $O(1)$.

b) Dacă mulţimea B este implementată printr-o listă liniară ordonată, atunci în cazul cel mai nefavorabil operaţia 3 are timpul de execuţie $O(1)$, iar operaţia 6 are timpul de execuţie $O(n)$.

c) Dacă mulţimea B este implementată printr-un *heap*, atunci în cazul cel mai nefavorabil operaţia 3 are timpul de execuţie $O(\log n)$, iar operaţia 6 are timpul de execuţie $O(\log n)$.

Concluzie: *heapul* este alegerea cea mai bună pentru implementarea mulţimii B.

2 Sarcini de lucru

Scrieţi un program C/C++ care implementează algoritmul de construcţie a unui arbore binar cu lungimea externă ponderată minimă.

Bibliografie

[1] Lucanu, D. şi Craus, M., *Proiectarea algoritmilor*, Editura Polirom, 2008.

[2] Moret, B.M.E.şi Shapiro, H.D. , *Algorithms from P to NP: Design and Efficiency*, The Benjamin/Cummings Publishing Company, Inc., 1991.