Proiectarea algoritmilor

Paradigma Backtracking

Lucrare de laborator nr. 13

Cuprins

Problema submulțimilor de sumă dată - descriere

- Se consideră o mulțime A cu n elemente, fiecare element $a \in A$ având o dimensiune $s(a) \in \mathbb{Z}_+$ și un număr întreg pozitiv M.
- Problema constă în determinarea tuturor submulțimilor $A' \subseteq A$ cu proprietatea $\sum\limits_{a \in A'} s(a) = M$.

Problema submulțimilor de sumă dată - modelul matematic

- Presupunem $A = \{1, \ldots, n\}$ și $s(i) = w_i, 1 \leq i \leq n$.
- Pentru reprezentarea soluțiilor avem două posibilități.
  1. Prin vectori care să conțină elementele care compun soluția.
     - Această reprezentare are dezavantajul că trebuie utilizat un algoritm de enumerare a vectorilor de lungime variabilă.
     - De asemenea, testarea condiției $a \in A \setminus A'$? nu mai poate fi realizată în timpul $O(1)$ dacă nu se utilizează spațiu de memorie suplimentar.
  2. Prin vectori de lungime n, $(x_1, \ldots, x_n)$ cu $x_i \in \{0, 1\}$ având semnificația: $x_i = 1$ dacă și numai dacă w_i aparține soluției (vectorii caracteristici).
- *Exemplu:* Fie $n = 4$, $(w_1, w_2, w_3, w_4) = (4, 7, 11, 14)$ și $M = 25$. Soluțiile sunt
  - $(4, 7, 14)$ care mai poate fi reprezentată prin $(1, 2, 4)$ sau $(1, 1, 0, 1)$ și
  - $(11, 14)$ care mai poate fi reprezentată prin $(3, 4)$ sau $(0, 0, 1, 1)$.
- Vom opta pentru ultima variantă, deoarece vectorii au lungime fixă.

Problema submulțimilor de sumă dată - modelul matematic (continuare)

- Remarcăm faptul că spațiul soluțiilor conține 2^n posibilități (elementele mulțimii $\{0,1\}^n$) și poate fi reprezentat printr-un arbore binar.
- În procesul de generare a soluțiilor potențiale, mulțimea A este partiționată astfel:
  - o parte $\{1,\ldots,k\}$ care a fost luată în considerare pentru a stabili candidații la soluție și
  - a doua parte $\{k+1,\ldots,n\}$ ce urmează a fi luată în considerare.
- Cele două părți trebuie să satisfacă următoarele două inegalități:
  - suma parțială dată de prima parte (adică de candidații aleși) să nu depășească M:

$$\sum_{i=1}^{k} x_i \cdot w_i \leq M \tag{1}$$

  - ceea ce rămâne să fie suficient pentru a forma suma M:

$$\sum_{i=1}^{k} x_i \cdot w_i + \sum_{i=k+1}^{n} w_i \geq M \tag{2}$$

- Cele două inegalități pot constitui criteriul de mărginire.
- Cu acest criteriu de tăiere, arborele parțial rezultat pentru exemplul anterior este cel reprezentat în figura 1.

Problema submulțimilor de sumă dată - modelul matematic (continuare)

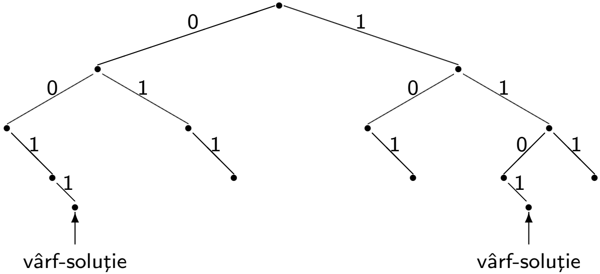

Figura 1 : Arbore parțial pentru submulțimi de sumă dată

- Criteriul de mărginire nu elimină toți subarborii care nu conțin vârfuri-soluție, dar elimină foarte mulți, restrângând astfel spațiul de căutare.
- Atingerea unui vârf pe frontieră presupune imediat determinarea unei soluții: suma $\sum_{i=k+1}^{n} w_i$ este zero (deoarece $k = n$) și dubla inegalitate dată de relațiile 1 și 2 implică $\sum_{i=1}^{n} w_i = M$.
- *Observație*: Dacă se utilizează drept criteriu de mărginire numai inegalitatea 1, atunci atingerea unui vârf pe frontieră în arborele parțial nu presupune neapărat și obținerea unei soluții. Mai trebuie verificat dacă suma submulțimii alese este exact M.

Problema submulțimilor de sumă dată - model matematic îmbunătățit

- Se consideră $w_1, w_2, ..., w_n$ în ordine crescătoare (fără a restrânge generalitatea);
- Pentru cazul în care suma parțială dată de candidații aleși este strict mai mică decât M ($\sum_{i=1}^{k} x_i w_i < M$), se introduce un criteriu de mărginire suplimentar:

$$\sum_{i=1}^{k} x_i w_i + w_{k+1} \leq M.$$

- Presupunem valorile $x_1, ..., x_{k-1}$ calculate.
- Notăm cu s suma parțială corespunzătoare valorilor $x_1, ..., x_{k-1}$ ($s = \sum\limits_{i=1}^{k-1} x_i w_i$) și cu r suma $\sum\limits_{i=k}^{n} w_i$.
- Presupunem $w_1 \leq M$ și $\sum\limits_{i=1}^{n} w_i \geq M$.

Problema submulțimilor de sumă dată - algoritm

```
procedure submultimiOpt(s,k,r)
    x_k ← 1
    if (s+w_k=M)
      then scrie(x^k) /* x^k=(x_1,x_2,...,x_k)) */
      else if (s+w_k+w_{k+1} ≤ M)
              then submultimiOpt(s+w_k,k+1,r-w_k)
           if ((s+r-w_k ≥ M) and (s+w_{k+1} ≤ M))
              then x_k ← 0
              submultimiOpt(s,k+1,r-w_k)
end
```

- *Precondiții*: $w_1 \leq M$ și $\sum_{i=1}^{n} w_i \geq M$. Astfel, înainte de apelul inițial sunt asigurate condițiile $s + w_k \leq M$ și $s + r \geq M$. Apelul inițial este submultimiOpt$\left(0, 1, \sum_{i=1}^{n} w_i\right)$.
- Condițiile $s + w_k \leq M$ și $s + r \geq M$ sunt asigurate și la apelul recursiv.
- Înainte de apelul recursiv submultimiOpt(s+w_k,k+1,r-w_k) nu este nevoie să se mai verifice dacă $\sum_{i=1}^{k} x_i w_i + \sum_{i=k+1}^{n} w_i \geq M$, deoarece $s + r > M$ și $x_k = 1$.
- Nu se verifică explicit nici $k > n$.
  - Inițial, $s = 0 < M$ și $s + r \geq M$ și $k = 1$.
  - De asemenea, în linia „if $(s + w_k + w_{k+1} \leq M)$”, deoarece $s + w_k < M$, rezultă $r \neq w_k$, deci $k + 1 \leq n$.

Sarcini de lucru și barem de notare

Sarcini de lucru:

1. Scrieți o funcție C/C++ care implementează algoritmul `submultimiOpt`.
2. Se consideră o mulțimea $A = \{1, \ldots, n\}$ și un număr întreg pozitiv M. Fiecare element $i \in A$ are o dimensiune $w_i \in \mathbb{Z}_+$. Scrieți un program care să afișeze submulțimile $A' \subseteq A$ cu proprietatea $\sum\limits_{i \in A'} w_i = M$.

Barem de notare:

1. Implementarea algoritmului `submultimiOpt`: 7p
2. Afișarea soluției: 2p
3. Baza: 1p

Bibliografie

Lucanu, D. și Craus, M., *Proiectarea algoritmilor*, Editura Polirom, 2008.