Proiectarea algoritmilor
Lucrare de laborator nr. 13
Paradigma *backtracking*
Problema submulţimilor de sumă dată

Cuprins

1 Descriere **1**

2 Modelul matematic **1**

3 Algoritm **3**

4 Sarcini de lucru şi barem de notare **3**

1 Descriere

Se consideră o mulţime A cu n elemente, fiecare element $a \in A$ având o dimensiune $s(a) \in \mathbb{Z}_+$ şi un număr întreg pozitiv M.

Problema constă în determinarea tuturor submulţimilor $A' \subseteq A$ cu proprietatea $\sum\limits_{a \in A'} s(a) = M$.

2 Modelul matematic

Presupunem $A = \{1, \ldots, n\}$ şi $s(i) = w_i, 1 \leq i \leq n$. Pentru reprezentarea soluţiilor avem două posibilităţi.

1. Prin vectori care să conţină elementele care compun soluţia.
   - Această reprezentare are dezavantajul că trebuie utilizat un algoritm de enumerare a vectorilor de lungime variabilă.
   - De asemenea, testarea condiţiei $a \in A \setminus A'$? nu mai poate fi realizată în timpul $O(1)$ dacă nu se utilizează spaţiu de memorie suplimentar.
2. Prin vectori de lungime n, $(x_1, \ldots, x_n)$ cu $x_i \in \{0, 1\}$ având semnificaţia: $x_i = 1$ dacă şi numai dacă w_i aparţine soluţiei (vectorii caracteristici).

Exemplu: Fie $n = 4$, $(w_1, w_2, w_3, w_4) = (4, 7, 11, 14)$ şi $M = 25$. Soluţiile sunt

- $(4, 7, 14)$ care mai poate fi reprezentată prin $(1, 2, 4)$ sau $(1, 1, 0, 1)$ şi
- $(11, 14)$ care mai poate fi reprezentată prin $(3, 4)$ sau $(0, 0, 1, 1)$.

Vom opta pentru ultima variantă, deoarece vectorii au lungime fixă.

Remarcăm faptul că spaţiul soluţiilor conţine 2^n posibilităţi (elementele mulţimii $\{0,1\}^n$) şi poate fi reprezentat printr-un arbore binar.

În procesul de generare a soluţiilor potenţiale, mulţimea A este partiţionată astfel:

- o parte $\{1,\ldots,k\}$ care a fost luată în considerare pentru a stabili candidaţii la soluţie şi
- a doua parte $\{k+1,\ldots,n\}$ ce urmează a fi luată în considerare.

Cele două părţi trebuie să satisfacă următoarele două inegalităţi:

- suma parţială dată de prima parte (adică de candidaţii aleşi) să nu depăşească 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 inegalităţi pot constitui criteriul de mărginire. Cu acest criteriu de tăiere, arborele parţial rezultat pentru exemplul anterior este cel reprezentat în figura 1.

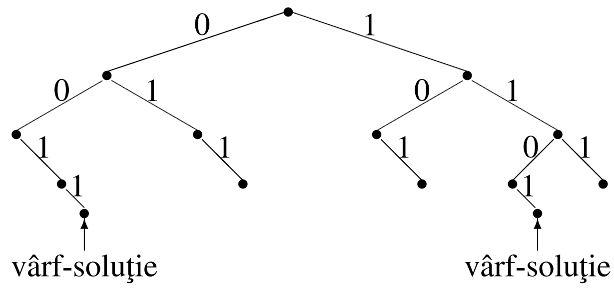

Figura 1: Arbore parţial pentru submulţimi de sumă dată

Criteriul de mărginire nu elimină toţi subarborii care nu conţin vârfuri-soluţie, dar elimină foarte mulţi, restrângând astfel spaţiul de căutare.

Atingerea unui vârf pe frontieră presupune imediat determinarea unei soluţii: suma $\sum_{i=k+1}^{n} w_i$ este zero (deoarece $k = n$) şi dubla inegalitate dată de relaţiile 1 şi 2 implică $\sum_{i=1}^{n} w_i = M$.

Observaţie: Dacă se utilizează drept criteriu de mărginire numai inegalitatea 1, atunci atingerea unui vârf pe frontieră în arborele parţial nu presupune neapărat şi obţinerea unei soluţii. Mai trebuie verificat dacă suma submulţimii alese este exact M.

Se consideră $w_1, w_2, ..., w_n$ în ordine crescătoare (fără a restrânge generalitatea).

Pentru cazul în care suma parţială dată de candidaţii aleşi 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 parţială corespunzătoare valorilor $x_1, ..., x_{k-1}$ ($s = \sum_{i=1}^{k-1} x_i w_i$) şi cu r suma $\sum_{i=k}^{n} w_i$. Presupunem $w_1 \leq M$ şi $\sum_{i=1}^{n} w_i \geq M$.

3 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
```

Precondiţii: $w_1 \leq M$ şi $\sum_{i=1}^{n} w_i \geq M$. Astfel, înainte de apelul iniţial sunt asigurate condiţiile $s+w_k \leq M$ şi $s+r \geq M$. Apelul iniţial este `submultimiOpt` $\left(0, 1, \sum_{i=1}^{n} w_i\right)$.

Condiţiile $s+w_k \leq M$ şi $s+r \geq M$ sunt asigurate şi 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 $s+r > M$ şi $x_k = 1$.

Nu se verifică explicit nici $k > n$.

- Iniţial, $s = 0 < M$ şi $s+r \geq M$ şi $k = 1$.
- De asemenea, în linia „`if` $(s+w_k+w_{k+1} \leq M)$”, deoarece $s+w_k < M$, rezultă $r \neq w_k$, deci $k+1 \leq n$.

4 Sarcini de lucru şi barem de notare

Sarcini de lucru:

1. Scrieţi o funcţie C/C++ care implementează algoritmul `submultimiOpt`.
2. Se consideră o mulţimea $A = \{1, \ldots, n\}$ şi un număr întreg pozitiv M. Fiecare element $i \in A$ are o dimensiune $w_i \in \mathbb{Z}_+$. Scrieţi un program care să afişeze submulţimile $A' \subseteq A$ cu proprietatea $\sum_{i \in A'} w_i = M$.

Barem de notare:

1. Implementarea algoritmului `submultimiOpt`: 7p
2. Afişarea soluţiei: 2p
3. Baza: 1p

Bibliografie

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