Proiectarea algoritmilor

Paradigma programării dinamice

Lucrare de laborator nr. 12

Cuprins

Problema rucsacului, varianta discretă - descrierea problemei

- Se consideră un rucsac de capacitate $M \in \mathbb{Z}_+$ și n obiecte $1, \ldots, n$ de dimensiuni (greutăți) $w_1, \ldots, w_n \in \mathbb{Z}_+$.
- Un obiect i este introdus în totalitate în rucsac, $x_i = 1$, sau nu este introdus deloc, $x_i = 0$, astfel că o umplere a rucsacului constă dintr-o secvență $x_1, \ldots, x_n$ cu $x_i \in \{0, 1\}$ și $\sum\limits_{i=1}^{n} x_i \cdot w_i \leq M$.
- Introducerea obiectului i în rucsac aduce profitul $p_i \in \mathbb{Z}$, iar profitul total este $\sum\limits_{i=1}^{n} x_i p_i$.
- *Problema constă în a determina o alegere* $(x_1, \ldots, x_n)$ *care să aducă un profit maxim.*
- Singura deosebire față de varianta continuă studiată la metoda greedy constă în condiția $x_i \in \{0, 1\}$, în loc de $x_i \in [0, 1]$.

Problema rucsacului, varianta discretă - model matematic

Problema inițială (starea $\mathrm{RUCSAC}(n, M)$):

- Funcția obiectiv:

$$\max \sum_{i=1}^{n} x_i \cdot p_i$$

- Restricții:

$$\sum_{i=1}^{n} x_i \cdot w_i \leq M$$
$$x_i \in \{0, 1\}, i = 1, \dots, n$$
$$w_i \in \mathbb{Z}_+, p_i \in \mathbb{Z}, i = 1, \dots, n$$
$$M \in \mathbb{Z}_+$$

Problema rucsacului, varianta discretă - model matematic (continuare)

Generalizarea problemei inițiale (starea $\mathrm{RUCSAC}(j, X)$):

- Funcția obiectiv:

$$\max \sum_{i=1}^{j} x_i \cdot p_i$$

- Restricții:

$$\sum_{i=1}^{j} x_i \cdot w_i \leq X$$
$$x_i \in \{0, 1\}, i = 1, \ldots, j$$
$$w_i \in \mathbb{Z}_+, p_i \in \mathbb{Z}, i = 1, \ldots, j$$
$$X \in \mathbb{Z}_+$$

Problema rucsacului, varianta discretă - model matematic (continuare)

- Notăm cu $f_j(X)$ valoarea optimă pentru instanța $\mathrm{RUCSAC}(j, X)$.
- Dacă $j = 0$ și $X \geq 0$, atunci $f_j(X) = 0$.
- Presupunem $j > 0$. Notăm cu $(x_1, \ldots, x_j)$ alegerea care dă valoarea optimă $f_j(X)$.
  - Dacă $x_j = 0$ (obiectul j nu este pus în rucsac), atunci, conform principiului de optim, $f_j(X)$ este valoarea optimă pentru starea $\mathrm{RUCSAC}(j-1, X)$ și de aici $f_j(X) = f_{j-1}(X)$.
  - Dacă $x_j = 1$ (obiectul j este pus în rucsac), atunci, din nou conform principiului de optim, $f_j(X)$ este valoarea optimă pentru starea $\mathrm{RUCSAC}(j-1, X - w_j)$ plus p_j și, de aici, $f_j(X) = f_{j-1}(X - w_j) + p_j$.
- Combinând relațiile de mai sus obținem:

$$f_j(X) = \begin{cases} -\infty, & \text{dacă } X < 0 \\ 0, & \text{dacă } j = 0 \text{ şi } X \geq 0 \\ \max\{f_{j-1}(X), f_{j-1}(X - w_j) + p_j\}, & \text{dacă } j > 0 \text{ şi } X \geq 0 \end{cases} \quad (1)$$

- Am considerat $f_j(X) = -\infty$, dacă $X < 0$.

Problema rucsacului, varianta discretă - model matematic (continuare)

- Din relația (1) rezultă că proprietatea de substructură optimă se caracterizează astfel:
  - Soluția optimă $(x_1, \ldots, x_j)$ a problemei $\mathrm{RUCSAC}(j, X)$ include soluția optimă $(x_1, \ldots, x_{j-1})$ a subproblemei $\mathrm{RUCSAC}(j-1, X - x_j w_j)$.
- Soluția optimă pentru $\mathrm{RUCSAC}(j, X)$ se poate obține utilizând soluțiile optime pentru subproblemele $\mathrm{RUCSAC}(i, Y)$ cu $1 \leq i < j, 0 \leq Y \leq X$.
- Relația (1) implică o recursie în cascadă și deci numărul de subprobleme de rezolvat este $O(2^n)$, fapt pentru care calculul și memorarea eficientă a valorilor optime pentru subprobleme devine un task foarte important.

Problema rucsacului, varianta discretă - exemplu

- Fie $M = 10, n = 3$ și greutățile și profiturile date de următorul tabel:

i	1	2	3
w_i	3	5	6
p_i	10	30	20

- Valorile optime pentru subprobleme sunt calculate cu ajutorul relației (1)≡(2)

$$f_j(X) = \begin{cases} -\infty, & \text{dacă } X < 0 \\ 0, & \text{dacă } j = 0 \text{ şi } X \geq 0 \\ \max\{f_{j-1}(X), f_{j-1}(X - w_j) + p_j\}, & \text{dacă } j > 0 \text{ şi } X \geq 0 \end{cases} \tag{2}$$

- Valorile optime pot fi memorate într-un tablou bidimensional astfel:

X	0	1	2	3	4	5	6	7	8	9	10
f_0	0	0	0	0	0	0	0	0	0	0	0
f_1	0	0	0	10	10	10	10	10	10	10	10
f_2	0	0	0	10	10	30	30	30	40	40	40
f_3	0	0	0	10	10	30	30	30	40	40	40

- Tabloul de mai sus este calculat linie cu linie.
  - Pentru a calcula valorile de pe o linie sunt consultate numai valorile de pe linia precedentă.
  - Exemplu: $f_2(8) = \max\{f_1(8), f_1(8-5) + 30\} = \max\{10, 40\} = 40$.

Problema rucsacului, varianta discretă - exemplu

- Tabloul valorilor optime are dimensiunea $n \cdot M$ (au fost ignorate prima linie și prima coloană).
- Dacă $M = O(2^n)$ rezultă că atât complexitatea spațiu, cât și cea timp sunt exponențiale.
- Privind tabloul de mai sus observăm că există multe valori care se repetă.
- *Cum putem memora mai compact tabloul valorilor optime*?
- *Soluție*: Construim graficele funcțiilor $f_0, f_1, f_2 \cdots$

Problema rucsacului, varianta discretă - exemplu

$$f_0(X) = \begin{cases} -\infty & ,X < 0 \\ 0 & ,X \geq 0 \end{cases}$$

$$g_0(X) = f_0(X - w_1) + p_1 = \begin{cases} -\infty & ,X < 3 \\ 10 & ,3 \leq X \end{cases}$$

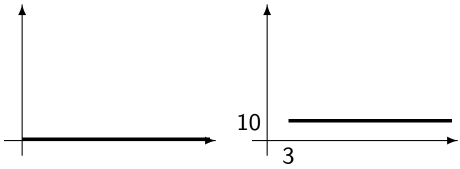

Figura 1: Funcțiile f_0 și g_0

Problema rucsacului, varianta discretă - exemplu

$$f_1(X) = \max\{f_0(X), g_0(X)\} = \begin{cases} -\infty & , X < 0 \\ 0 & , 0 \leq X < 3 \\ 10 & , 3 \leq X \end{cases}$$

$$g_1(X) = f_1(X - w_2) + p_2 = \begin{cases} -\infty & , X < 5 \\ 30 & , 5 \leq X < 8 \\ 40 & , 8 \leq X \end{cases}$$

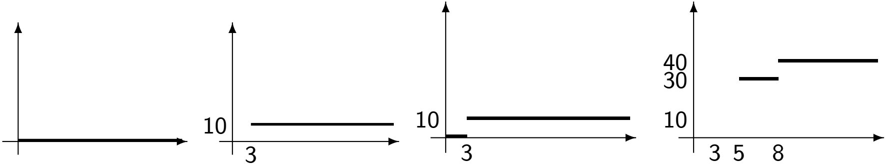

Figura 2: Funcțiile f_0 și g_0; Funcțiile f_1 și g_1

Problema rucsacului, varianta discretă - exemplu

$$f_2(X) = \max\{f_1(X), g_1(X)\} = \begin{cases} -\infty & , X < 0 \\ 0 & , 0 \leq X < 3 \\ 10 & , 3 \leq X < 5 \\ 30 & , 5 \leq X < 8 \\ 40 & , 8 \leq X \end{cases}$$

$$g_2(X) = f_2(X - w_3) + p_3 = \begin{cases} -\infty & , X < 6 \\ 20 & , 6 \leq X < 9 \\ 30 & , 9 \leq X < 11 \\ 50 & , 11 \leq X < 14 \\ 60 & , 14 \leq X \end{cases}$$

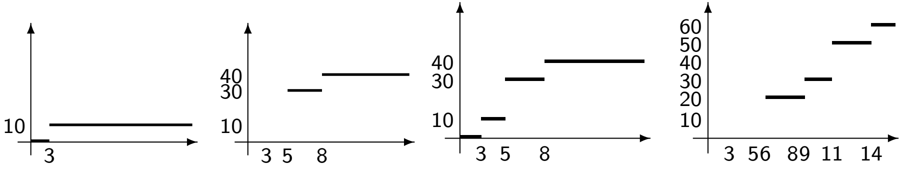

Figura 3: Funcțiile f_1 și g_1; Funcțiile f_2 și g_2

Problema rucsacului, varianta discretă - exemplu

$$f_3(X) = \max\{f_2(X), g_2(X)\} = \begin{cases} -\infty & , X < 0 \\ 0 & , 0 \leq X < 3 \\ 10 & , 3 \leq X < 5 \\ 30 & , 5 \leq X < 8 \\ 40 & , 8 < X \leq 11 \\ 50 & , 11 \leq X < 14 \\ 60 & , 14 \leq X \end{cases}$$

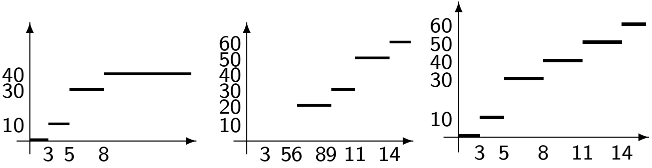

Figura 4: Funcțiile f_2 și g_2; Funcția f_3

Problema rucsacului, varianta discretă - exemplu (comentarii)

- Se remarcă faptul că funcțiile f_i și g_i sunt funcții în scară. Graficele acestor funcții pot fi reprezentate prin mulțimi finite din puncte din plan.
  - De exemplu, graficul funcției f_2 este reprezentat prin mulțimea $\{(0,0),(3,10),(5,30),(8,40)\}$.
- O mulțime care reprezintă o funcție în scară conține acele puncte în care funcția face salturi.
- Graficul funcției g_i se obține din graficul funcției f_i printr-o translație.
- Graficul funcției f_{i+1} se obține prin interclasarea graficelor funcțiilor f_i și g_i.

Problema rucsacului, varianta discretă - algoritm (descriere)

- În general, fiecare f_i este complet specificat de o mulțime $S_i = \{(X_j, Y_j) \mid j = 0, \ldots, r\}$, unde $Y_j = f_i(X_j)$.
    - Presupunem $X_1 < \cdots < X_r$.
- Analog, funcțiile g_i sunt reprezentate prin mulțimile $T_i = \{(X + w_{i+1}, Y + p_{i+1}) \mid (X, Y) \in S_i\}$.
    - Notăm $T_i = \tau(S_i)$ și $S_{i+1} = \mu(S_i, T_i)$.
- Mulțimea S_{i+1} se obține din S_i și T_i prin interclasare.
    - Operația de interclasare se realizează într-un mod asemănător cu cel de la interclasarea a două linii ale orizontului.
- Se consideră o variabilă L care ia valoarea 1 dacă graficul lui f_{i+1} coincide cu cel al lui f_i și cu 2 dacă el coincide cu cel al lui g_i.
    - Deoarece $(0,0)$ aparține graficului rezultat, considerăm $L = 1, j = 1$ și $k = 1$.

Problema rucsacului, varianta discretă - algoritm de interclasare grafice (descriere)

Presupunând că la un pas al interclasării se compară $(X_j, Y_j) \in S_i$ cu $(X_k, Y_k) \in T_i$, atunci:

- dacă $L = 1$:
  - dacă $X_j < X_k$, atunci se adaugă (X_j, Y_j) în S_{i+1} și se incrementează j;
  - dacă $X_j = X_k$:
    - dacă $Y_j \geq Y_k$, atunci se adaugă (X_j, Y_j) în S_{i+1} și se incrementează j și k;
    - dacă $Y_j < Y_k$, atunci se adaugă (X_k, Y_k) în S_{i+1}, $L = 2$ și se incrementează j și k;
  - dacă $X_j > X_k$ sau $j > |S_i|$:
    - dacă $Y_{j-1} \geq Y_k$, atunci se incrementează k;
    - dacă $Y_{j-1} < Y_k$, atunci $L = 2$;
- dacă $L = 2$:
  - dacă $X_k < X_j$, atunci se adaugă (X_k, Y_k) în S_{i+1} și se incrementează k;
  - dacă $X_k = X_j$:
    - dacă $Y_k \geq Y_j$, atunci se adaugă (X_k, Y_k) în S_{i+1} și se incrementează j și k;
    - dacă $Y_k < Y_j$, atunci se adaugă (X_j, Y_j) în S_{i+1}, $L = 1$ și se incrementează j și k;
  - dacă $X_k > X_j$ sau $k > |T_i|$:
    - dacă $Y_{k-1} \geq Y_j$, atunci se incrementează j;
    - dacă $Y_{k-1} < Y_j$, atunci $L = 1$;

Dacă se termină mulțimea S_i, atunci se adauga la S_{i+1} restul din T_i.
Dacă se termină mulțimea T_i, atunci se adauga la S_{i+1} restul din S_i.
Notăm cu `interclGrafice(S_i,T_i)` funcția care determină S_{i+1} conform algoritmului de mai sus.

Problema rucsacului, varianta discretă - algoritm de extragere a solutiei (exemplu)

- $S_3 = \{(0,0),(3,10),(5,30),(8,40),(11,50),(14,60)\}$.
- $S_2 = \{(0,0),(3,10),(5,30),(8,40)\}$.
- $S_1 = \{(0,0),(3,10)\}$.
- $S_0 = \{(0,0)\}$.
- Se caută în $S_n = S_3$ perechea (X_j, Y_j) cu cel mai mare X_j pentru care $X_j \leq M$. Obținem $(X_j, Y_j) = (8,40)$. Deoarece $(8,40) \in S_3$ și $(8,40) \in S_2$ rezultă $f_{optim}(M) = f_{optim}(8) = f_3(8) = f_2(8)$ și deci $x_3 = 0$. Perechea (X_j, Y_j) rămâne neschimbată.
- Pentru că $(X_j, Y_j) = (8,40)$ este în S_2 și nu este în S_1, rezultă că $f_{optim}(8) = f_1(8 - w_2) + p_2$ și deci $x_2 = 1$. În continuare se ia $(X_j, Y_j) = (X_j - w_2, Y_j - p_2) = (8-5, 40-30) = (3,10)$.
- Pentru că $(X_j, Y_j) = (3,10)$ este în S_1 și nu este în S_0, rezultă că $f_{optim}(3) = f_1(3 - w_1) + p_1$ și deci $x_1 = 1$.

Problema rucsacului, varianta discretă - algoritm de extragere a solutiei (descriere)

- Inițial se determină perechea $(X_j, Y_j) \in S_n$ cu cel mai mare X_j pentru care $X_j \leq M$. Valoarea Y_j constituie încărcarea optimă a rucsacului, *i.e.*, valoarea funcției obiectiv din problema inițială.
- Pentru $i = n-1, \ldots, 0$:
    - dacă (X_j, Y_j) este în S_i, atunci $f_{i+1}(X_j) = f_i(X_j) = Y_j$ și se consideră $x_{i+1} = 0$ (obiectul $i+1$ nu este ales);
    - dacă (X_j, Y_j) nu este în S_i, atunci $f_{i+1}(X_j) = f_i(X_j - w_{i+1}) + p_{i+1} = Y_j$ și se consideră $x_{i+1} = 1$ (obiectul $i+1$ este ales), $X_j = X_j - w_{i+1}$ și $Y_j = Y_j - p_{i+1}$.

Problema rucsacului, varianta discretă - algoritm (pseudocod)

procedure $\mathtt{rucsac\_II(M,\ n,\ w,\ p,\ x)}$

$\quad S_0 \leftarrow \{(0,0)\}$

$\quad T_0 \leftarrow \{(w_1,p_1)\}$

$\quad \mathtt{for}\ i \leftarrow 1\ \mathtt{to}\ n$

$\qquad S_i \leftarrow \mathtt{interclGrafice}(S_{i-1},T_{i-1})$

$\qquad T_i \leftarrow \{(X+w_{i+1},Y+p_{i+1}) \mid (X,Y) \in S_i\}$

$\quad$ determină (X_j,Y_j) cu $X_j = \max\{X_i \mid (X_i,Y_i) \in S_n, X_i \leq M\}$

$\quad \mathtt{for}\ i \leftarrow n\text{-}1\ \mathtt{downto}\ 0\ \mathtt{do}$

$\qquad \mathtt{if}\ (X_j,Y_j) \in S_i$

$\qquad\quad \mathtt{then}\ x_{i+1} \leftarrow 0$

$\qquad\quad \mathtt{else}\ x_{i+1} \leftarrow 1$

$\qquad\qquad X_j \leftarrow X_j - w_{i+1}$

$\qquad\qquad Y_j \leftarrow Y_j - p_{i+1}$

end

Sarcini de lucru și barem de notare

Sarcini de lucru:

1. Scrieți o funcție C/C++ care implementează algoritmul `rucsac`.
2. Se consideră un rucsac de capacitate $M \in \mathbb{Z}_+$ și n obiecte $1, \ldots, n$ de dimensiuni (greutăți) $w_1, \ldots, w_n \in \mathbb{Z}_+$.. Scrieți un program care să afișeze soluția optimă.

Barem de notare:

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

Bibliografie

- Lucanu, D. și Craus, M., *Proiectarea algoritmilor*, Editura Polirom, 2008.
- R.E. Bellman și S.E. Dreyfus, *Applied Dynamic Programming*, Princeton University Press, 1962.
- Moret, B.M.E.și Shapiro, H.D. , *Algorithms from P to NP: Design and Efficiency*, The Benjamin/Cummings Publishing Company, Inc., 1991.