Algoritmi şi structuri de date (I). Seminar 13: Aplicaţii ale programării dinamice (II).

Problema 1 (*Problema colectării obiectelor*) Se consideră o grilă de dimensiune $m \times n$ ale cărei celule pot fi de unul dintre următoarele trei tipuri: (i) celulă accesibilă goală; (ii) celulă accesibilă care conţine un obiect; (iii) celulă inaccesibilă. Se pune problema determinării unui traseu pentru un robot care pleacă din celula din colţul din stânga sus cu scopul de a ajunge în celula din colţul din dreapta jos trecând prin celule accesibile şi colectând cât mai multe obiecte.

Indicaţie. Se consideră că informaţiile privind starea fiecărei celule sunt stocate într-o matrice $B[1..m, 1..n]$ cu elemente din $\{-1, 0, 1\}$: $B[i, j] = -1$ dacă celula este inaccesibilă, $B[i, j] = 0$ dacă celula este accesibilă dar nu conţine obiect, $B[i, j] = 1$ dacă celula este accesibilă şi conţine un obiect. Un exemplu de matrice este:

$$B = \begin{bmatrix} 0 & -1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & -1 & 1 & 0 \\ 0 & 1 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ -1 & -1 & -1 & 0 & 1 & 0 \end{bmatrix}$$

Considerând că $F[i, j]$ conţine numărul maxim de obiecte colectate de-a lungul unui traseu care se termină în celula (i, j) relaţia de recurenţă pentru calculul lui $F[i, j]$ este:

$$F[i,j] = \begin{cases} B[0,0] & i = 0, j = 0 \\ F[i-1,0] + B[i,0] & j = 0, i \geq 0, B[i,0]! = -1 \\ -\infty & j = 0, i > 0, B[i,0] = -1 \\ F[0,j-1] + B[0,j] & i = 0, j > 0, B[0,j]! = -1 \\ -\infty & i = 0, j > 0, B[0,j] = -1 \\ \max\{F[i-1,j], F[i,j-1]\} + B[i,j] & \text{altfel} \end{cases}$$

Problema 2 (*Secţionarea optimală a unei tije*) Se consideră o tijă de lungime n care se doreşte a fi secţionată în fragmente de lungimi întregi. Stiind că pentru fiecare lungime de fragment este un alt preţ se pune problemă determinării unei modalităţi de secţionare care să conducă la un cost cât mai mic.

De exemplu pentru o tijă de lungime 8 pentru care preţurile secţionărilor sunt $p = [2, 3, 1, 4, 6, 5, 6, 5]$ (obţinerea unui segment de lungime 1 are preţul 2, obţinerea unui segment de lungimea 2 are preţul 3, pentru un segment de lungime 3 preţul este 1 etc) o secţionare optimală ar conduce la două segmente de lungime 3 şi un segment de lungime 2.

Indicaţie. Problema secţionării este similară cu cea a înmulţirii optimale a unui şir de matrici în sensul că pentru a se obţine o secţionare optimală, fiecare dintre cele două fragmente obţinute în urma unei tăieturi trebuie secţionat optimal. Tija poate fi considerată ca un şir de elemente de lungime 1. Considerăm $C[i, j]$ costul secţionării optimale a porţiunii de tijă delimitată de elementele i şi j. Relaţia de recurenţă corespunzătoare calculului lui $C[i, j]$ este:

$$C[i,j] = \begin{cases} p[0] & i = j \\ \min(p[j-i], \min_{i \leq l < j} C[i,l] + C[l+1,j]) & i < j \\ 0 & i > j \end{cases}$$

Dacă $i = j$ înseamnă că fragmentul are deja lungimea 1 şi nu mai trebuie secţionat, iar preţul lui corespunde lui $p[1]$. Dacă $i < j$ atunci se alege varianta de cost minim: fie nu se aplică tăietură (în acest caz preţul este corespunzător lungimii fragmentului, $p[j - i]$) sau se aplică la poziţia l ($i \leq l < j$) care conduce la costul cel mai mic (în condiţiile în care fiecare dintre fragmente este la rândul său fragmentat optimal). Pentru a determina mai uşor poziţiile de tăiere este util ca în paralel cu construirea tabloului C să se completeze şi un tablou cu poziţiile de tăiere: $t[i, j]$ conţine poziţia tăieturii aplicate tijei delimitate de elementele i şi j.

$$t[i,j] = \begin{cases} i & i = j \\ j & \text{dacă } i < j \text{ şi } C[i,j] = p[j-i] \\ lmin & \text{dacă } i < j \text{ şi } C[i,lmin] + C[lmin+1,j]) \leq C[i,l] + C[l+1,j]), l = \overline{i,j-1} \\ 0 & i > j \end{cases}$$

Pentru exemplul de mai sus se obţine următoarea matrice de costuri:

```
[[2 3 1 3 4 2 4 5]
 [0 2 3 1 3 4 2 4]
 [0 0 2 3 1 3 4 2]
 [0 0 0 2 3 1 3 4]
 [0 0 0 0 2 3 1 3]
 [0 0 0 0 0 2 3 1]
 [0 0 0 0 0 0 2 3]
 [0 0 0 0 0 0 0 2]]
```

şi matrice de tăieturi (cu indicii elementelor din tijă pornind de la 0 - o valoare egală cu 0 înseamnă că se aplică tăietură după primul element, o valoare egală cu 1 înseamnă că se aplică tăietură după al doilea element etc):

```
[[0 1 2 0 4 2 0 2]
 [0 1 2 3 1 5 3 1]
 [0 0 2 3 4 2 6 4]
 [0 0 0 3 4 5 3 7]
 [0 0 0 0 4 5 6 4]
 [0 0 0 0 0 5 6 7]
 [0 0 0 0 0 0 6 7]
 [0 0 0 0 0 0 0 7]]
```

Problema 3 (*Problema închiderii tranzitive*) Considerăm o relaţie binară $R \subset \{1,\ldots,n\} \times \{1,\ldots,n\}$. Închiderea sa tranzitivă este o relaţie binară $R^* \subset \{1,\ldots,n\} \times \{1,\ldots,n\}$ având proprietatea: dacă pentru $i,j \in \{1,\ldots,n\}$ există $i_1,\ldots,i_m \in \{1,\ldots,n\}$ cu proprietăţile: $(i_1,i_2) \in R$, $(i_2,i_3) \in R$, ... ,$(i_{m-1},i_m) \in R$ iar $i = i_1$ şi $j = i_m$ atunci $(i,j) \in R^*$.

Pentru a construi R^* se consideră următorul set de relaţii binare $R^0 = R$, R^1, ..., $R^n = R^*$, definite astfel: dacă pentru $i,j \in \{1,\ldots,n\}$ există $i_1,\ldots,i_m \in \{1,\ldots,k\}$ cu proprietăţile: $(i_1,i_2) \in R$, $(i_2,i_3) \in R$, ..., $(i_{m-1},i_m) \in R$ iar $i = i_1$ şi $j = i_k$ atunci $(i,j) \in R^k$. Relaţiile pot fi descrise prin recurenţe astfel: $(i,j) \in R^k$ dacă $(i,j) \in R^{k-1}$ sau $(i,k) \in R^{k-1}$ şi $(k,j) \in R^{k-1}$.

Indicaţie. Pentru a elabora algoritmul de construire a lui R^* considerăm relaţiile binare pe $\{1,2,\ldots,n\}$ reprezentate prin matrici ale căror elemente sunt definite astfel:

$$r_{ij} = \begin{cases} 1 & \text{dacă } (i,j) \in R \\ 0 & \text{dacă } (i,j) \notin R \end{cases}$$

Relaţiile de recurenţă pentru construirea matricilor asociate relaţiilor binare $R^0, R^1, \ldots, R^n$ sunt:

$$r_{ij}^k = \begin{cases} 1 & \text{dacă } r_{ij}^{k-1} = 1 \text{ sau } (r_{ik}^{k-1} = 1 \text{ şi } r_{kj}^{k-1} = 1) \\ 0 & \text{altfel} \end{cases} \qquad k = \overline{1,n}$$

cu $r_{ij}^0 = r_{ij}$. Folosind aceste relaţii de recurenţă se poate construi matricea asociată relaţiei binare R^n, utilizând doar două matrici auxiliare în modul descris în Algoritmul 1.

Algoritmul pentru determinarea închiderii tranzitive este cunoscut şi sub numele de algoritmul lui Warshall.

Algorithm 1 Determinarea închiderii tranzitive a unei relaţii binare

```
închidere_tranzitivă(R[1..n, 1..n])
R2[1..n, 1..n] ← R[1..n, 1..n]
for k ← 1, n do
  R1[1..n, 1..n] ← R2[1..n, 1..n]
  for i ← 1, n do
    for j ← 1, n do
      if (R1[i, j] = 1) or (R1[i, k] = 1 and R1[k, j] = 1) then
        R2[i, j] ← 1
      else
        R2[i, j] ← 0
      end if
    end for
  end for
end for
return R2[1..n, 1..n]
```

Probleme suplimentare

1. (*Problema subşirului strict crescător de sumă maximă.*) Fie $a_1, a_2, \ldots, a_n$ un şir de valori reale pozitive. Să se determine un subşir strict crescător pentru care suma elementelor este maximă. *Indicaţie.* Se aplică ideea de la determinarea celui mai lung subşir strict crescător doar că în loc să se contorizeze numărul de elemente din subşir se adună numărul lor.

2. Se consideră un şir de valori reale (pozitive şi negative). Să se determine subşir de elemente cu semne alternate (un element pozitiv este urmat de un element negativ iar un element negativ este urmat de un element pozitiv) pentru care suma valorilor modulelor este maximă.

   Indicaţie. Se aplică ideea de la determinarea unui subşir strict crescător de sumă maximă doar că în loc ca elementele să fie ordonate crescător acestea trebuie să fie cu semne alternate (şi se cumulează modulele elementelor).

3. Aplicaţi tehnica memoizării în algoritmul de determinare a distanţei de editare.