

Seminar 11

Programare dinamică.

Ștefan Ciobâcă, Dorel Lucanu
Universitatea “Alexandru Ioan Cuza”, Iași

Săptămâna 9 Mai - 13 Mai 2016

1. Algoritmi pseudopolinomiali pentru probleme *knapsack-like*:

- (a) Găsiți un algoritm pseudopolinomial pentru problema discretă a rucsacului (care folosește metoda programării dinamice). Care sunt subproblemele rezolvate de algoritm?
- (b) Găsiți un algoritm pseudo-polinomial pentru următoarea problemă: dându-se o mulțime de numere întregi, să se determine dacă poate fi partiționată în două submulțimi de sumă egală.

2. Algoritmi de tip *edit-distance*:

- (a) Găsiți un algoritm pentru problema distanței de editare când singurele operații permise sunt inserarea și ștergerea.
- (b) Găsiți un algoritm polinomial pentru problema celui mai lung subsir crescător. Care sunt subproblemele rezolvate de algoritm? Arătați cum este aplicat principiul de optim în relația de recurență.
- (c) Dându-se două șiruri de numere, găsiți cel mai lung subsir crescător care apare în ambele șiruri.
- (d) Găsiți un algoritm polinomial pentru problema subsecvenței de sumă maximă (dându-se un vector $A[0..n-1]$, să se găsească doi indecși i, j astfel încât suma elementelor $A[i..j]$ să fie maximă).

3. Algoritmi în $O(n^3)$:

- (a) O matrice $A_{(n,m)}$ poate fi înmulțită cu o matrice $B_{(m,k)}$ făcând $n \times m \times k$ înmulțiri și obținem o matrice $C_{(n,k)}$. Înmulțirea matricilor este asociativă. Dacă avem trei matrici $A_{(10000,10000)}, B_{(10000,10)}, C_{(10,100)}$, facem mai multe înmulțiri dacă calculăm $A \times (B \times C)$ (câte?) decât dacă calculăm $(A \times B) \times C$ (câte înmulțiri?).

Dându-se dimensiunile $(n_0, m_0), (n_1, m_1), \dots, (n_{l-1}, m_{l-1})$ unei secvențe de matrici A_0, A_1, \dots, A_{l-1} (atenție: se dau doar dimensiunile, nu și matricile), să se determine numărul minim de înmulțiri de scalari necesare pentru calculul produsului matricilor.

Hint. Subproblemele sunt: $x[i..j]$ = numărul minim de înmulțiri necesare pentru a calcula $A_i \times A_{i+1} \times \dots \times A_j$.

- (b) Fie a_0, \dots, a_{n-1} o secvență de chei ordonate crescător care sunt folosite pentru a construi un arbore binar de căutare. Dându-se:
 - p_i ($0 \leq i \leq n-1$), probabilitatea de căuta elementul a_i ;
 - q_i ($0 \leq i \leq n$), probabilitatea de a căuta un element mai mic decât a_i (dacă există a_i) și mai mare decât a_{i-1} (dacă există a_{i-1}),

să se calculeze arborele binar care asigură timpul mediu de căutare minim.

Hint. Subproblemele sunt: $x[i..j]$ = arborele binar de căutare optim pentru numerele a_i, a_{i+1}, \dots, a_j . Atenție: datele de intrare nu sunt a_0, \dots, a_{n-1} , ci p_i ($0 \leq i \leq n-1$) și q_i ($0 \leq j \leq n$).