

drum - Descrierea soluției

Autor Carmen Mincă

O soluție se poate obține prin aplicarea metodei programării dinamice, metoda înainte. Se utilizează două masive tridimensionale A și T . În masivul A , $A[1][1][1]$ va memora valoarea asociată punctului de coordonate $(1, 1, 1)$, ..., $A[k][i][j]$ va memora valoarea asociată punctului de coordonate (k, i, j) , pentru $1 \leq k \leq n$, $1 \leq i \leq k$, $1 \leq j \leq k$. Masivul T se va inițializa cu valorile din masivul A și va fi utilizat pentru a reține valorile sumelor maxime parțiale calculate pentru drumurile construite.

Pentru un punct de coordonate (k, i, j) , suma maximă care se poate calcula, pentru un drum care pornește din acest punct, se obține pe baza relațiilor:

$$T[n][i][j] = A[n][i][j], \text{ pentru } 1 \leq i \leq n, 1 \leq j \leq n. (k=n)$$

$$T[k][i][j] = A[k][i][j] + \max\{T[k+1][i][j+1], T[k+1][i+1][j], T[k+1][i+1][j+1]\} \\ \text{pentru } k=n-1, n-2, \dots, 1, 1 \leq i \leq k, 1 \leq j \leq k.$$

Suma maximă va fi reținută în $T[1][1][1]$. Pentru a afișa numerelor punctelor de pe un drum de sumă maximă, refacem "traseul" prin care a fost obținută suma maximă, pornind de la $T[1][1][1]$ sau se memorează coordonatele punctelor în timpul construirii sumelor parțiale maxime.

Pentru exemplul din enunț, se pot reprezenta punctele separat, pe plane:

Planul de ecuație: $k=1$

$P_1(1, 1, 1)$

Planul de ecuație: $k=2$

$P_2(2, 1, 1)$	$P_3(2, 1, 2)$
$P_4(2, 2, 1)$	$P_5(2, 2, 2)$

Planul de ecuație: $k=3$

$P_6(3, 1, 1)$	$P_7(3, 1, 2)$	$P_8(3, 1, 3)$
$P_9(3, 2, 1)$	$P_{10}(3, 2, 2)$	$P_{11}(3, 2, 3)$
$P_{12}(3, 3, 1)$	$P_{13}(3, 3, 2)$	$P_{14}(3, 3, 3)$

Valorile asociate punctelor:

Planul de ecuație: $k=1$

3

Planul de ecuație: $k=2$

6	5
7	2

Planul de ecuație: $k=3$

4	5	8
7	6	1
7	8	13

Se obțin două drumuri: $D_1 = (P_1, P_4, P_{13})$ și $D_2 = (P_1, P_5, P_{14})$ și $D_1 < D_2$. Astfel drumul căutat este $D_1 = (P_1, P_4, P_{13})$ iar suma maximă este $S = 3 + 7 + 8 = 3 + 2 + 13 = 18$.

atac - Descrierea soluției

Asociem hartii un graf neorientat cu n vârfuri și m muchii, fiecărui oraș corespunzându-i un vârf și fiecărui drum direct corespunzându-i o muchie. Notăm cu k numărul de noduri care sunt legate printr-o muchie de nodul X . Fie aceste noduri $X_1, X_2, \dots, X_k$.

Din fiecare nod U_i care conține o unitate, vom face o cautare în lățime pentru a afla lungimile minime ale drumurilor de la acest nod la toate nodurile X_j , $1 \leq j \leq k$.

Soluția 1. (100 puncte) - Tiberiu Daneț

Problema se reduce la a realiza un cuplaj maxim de cost minim în graful bipartit în care una dintre mulțimi este formată din nodurile ce conțin unități, iar cealaltă mulțime de nodurile $X_1, X_2, \dots, X_k$ (fiecare unitate trebuie dusă într-un nod X_j și nu are rost să ducem mai multe unități în același nod X_j). Costul de pe muchia ce leagă nodul U_i de nodul X_j este lungimea minimă a drumului de la U_i la X_j găsită anterior.

Complexitatea cuplajului maxim de cost minim în graful bipartit complet cu $k + U$ noduri se face în complexitate $O(k * k * k * U)$ (un drum alternant de creștere se poate obține cu algoritmul Bellman-Ford și sunt k drumuri alternante de creștere).

Soluția 2. (60 puncte) - Alin Burța

Dacă numărul unităților disponibile este suficient de mic (pentru 60% din teste avem $u \leq 8$), determinarea modului în care cuplăm cele u unități cu cele k noduri vecine cu X se poate realiza cu un algoritm de tip backtracking.

virus - Descrierea soluției

Autor Pățcaș Csaba

Problema se poate rezolva prin mai multe metode, având complexități diferite.

Fie S suma lungimilor virușilor.

O abordare bazată pe algoritmul naiv de căutare al subșirului are complexitatea $O(S \cdot m)$, și poate obține 10 de puncte.

Construirea unui arbore de sufixe se poate face în $O(m^2)$, după care numărarea apariției fiecărui șir se poate face în $O(S)$. Deci aceasă soluție are în total complexitatea $O(m^2 + S)$ și obține 30 de puncte.

Aplicarea de n ori a unui algoritm liniar de potrivire a șirurilor (de exemplu Knuth-Morris-Pratt, sau Boyer-Moore) are complexitatea $O(n \cdot m + S)$ și obține între 30 și 60 de puncte.

Construirea șirului de sufixe al șirului de biți și se poate face prin diferiți algoritmi cu complexități de $O(m \cdot \log^2 m)$, $O(m \cdot \log m)$ sau $O(m)$. După obținerea șirului de sufixe, numărul de apariții al fiecărui virus se poate determina prin două căutări binare.

Această abordare are în total complexitatea $O(m \cdot \log^2 m + S \cdot \log m)$, $O(m \cdot \log m + S \cdot \log m)$ sau $O(m + S \cdot \log m)$ și poate obține 70–80 de puncte.

Pentru obținerea punctajului maxim este nevoie de o implementare eficientă a algoritmului Aho-Corasick.