empowersoft

Concurs online de informatică
Categoria PROGRAMARE

Clasa a 10-a

DESCRIERE SOLUŢII

PROBLEMA 1 EXPOZIŢII

Autor: prof. Iulia Lincan
Colegiul Național "Mihai Viteazul", Ploieşti

1. Se reţin intervalele într-un vector v cu elemente de tip struct şi apoi se sortează crescător acest vector după ziua de început.

2. Se foloseşte metoda programării dinamice

Se utilizează un vector D cu N elemente, unde:

D[i] = suma maximă obţinută la pasul i (pictorul i), cu i de la 1 la N

Formule de recurenţă:

Iniţial, **D[i] = max(c, D[i - 1])**, unde **c = (v[i].y - v[i].x + 1) * v[i].z**, adica suma obţinută dacă pictorul i îşi prezintă lucrările în tot intervalul dorit.

Apoi,
D[i] = max(D[i], D[j] + c), daca v[j].y < v[i].x (pictorul i poate începe expoziţia dupa pictorul j)
= max(D[i], D[j] + (v[i].y - v[j].y) * v[i].z), altfel
pentru orice j de la 1 la i-1

unde, **D[j] + (v[i].y - v[j].y) * v[i].z**, înseamnă că intervalul pentru pictorul i ramane v[i].y - v[j].y, ca să intre pictorul j până la sfârşit.

Apoi, tot pentru valoarea i curentă, se calculează:

$$s = \sum(v[k].uy - \max(v[k].ux,\ v[i].x) + 1) * v[k].z \text{, cu k de la i-1 la 1}$$

şi se acualizează **D[i] = max(D[i], D[i-1] - s + c)**, ceea ce înseamnă că se scade din suma maximă obţinută la pasul **i-1** sumele corespunzătoare intervalelor anterioare, ca să poata intra pictorul i de la început.
Pe parcursul algoritmului se actualizează corespunzător în câmpurile *ux* şi *uy* capetele intervalelor care contribuie la suma maximă.

Deoarece intervalele sunt sortate după capătul din stânga şi nu există două intervale egale sau incluse unul în altul, vectorul D este strict crescător, deci soluţia problemei va fi D[N].

Sortarea are complexitatea O(NlogN), prelucrarea $O(N^3)$, algoritmul are din punct de vedere al timpului de executare complexitatea $O(N^3)$.

PROBLEMA 2 INAMICI

Autor: student Alexandru Ilaşi
Computer Science University of Oxford, anul II

Soluţia 100 puncte
Observăm că toți inamicii dintr-un interval [left, right) pot fi eliminați în right − left mutări, eliminând câte o coloană pe rând.

Ploieşti, 28 mai 2022

Cum se poate optimiza? Dacă aplicăm un număr de eliminări orizontale, se elimină mai mulți inamici decât mutări făcute. Astfel, la fiecare pas putem fie să eliminăm fiecare coloană, fie să eliminăm coloana cu număr minim de inamici eliminând liniile rămase în viață din aceasta, urmând să împărțim jocul în 2 jocuri -> ce rămâne la stânga și ce rămâne la dreapta.
Algoritmul aplică tehnica divide et impera, la fiecare pas alegând mutarea optimă.
În funcție de modul în care găsim coloana cu număr minim de inamici, punctajul poate diferi. Pentru optimizare se poate folosi structura rmq pentru a răspunde la query-uri în complexitate O(log (lungime interval)). Acesta va conduce la o complexitate finală de O(n log n). (O(n log n) pentru a construi structura + O(n log n) pentru a afla raspunsul)

PROBLEMA 3 ARBO

Autori: prof. Luminița-Gabriela Năstase
elev Valentina-Georgiana Veleat
Colegiul Național " Nichita Stănescu", Ploieşti

- Prima cerință se rezolvă cu ajutorul algoritmului cunoscut sub numele de *Înfășurătoarea convexă*. Se introduce în stivă punctul având cea mai mică abscisă, iar dacă sunt mai multe puncte cu aceeași abscisă, punctul ales va avea cea mai mică ordonată. Se încearcă pe rând restul de puncte aflate în exteriorul poligonului curent, până când ajungem din nou la punctul de pornire.
- A doua cerință presupune utilizarea algoritmului *Welzl (Minimum Enclosing Circle)*. Se alege un punct p aleatoriu și un set P de puncte. Se află cercul minim care conține cele P-{p} puncte. Se verifică apoi dacă p se află în interiorul cercului. Dacă nu, p se află pe circumferința cercului determinat.
- Ultima cerință necesită calculul ariei poligonului convex

PROBLEMA 4 RESTAURANT

Autor: prof. Marius Nicoli
Colegiul Național "Frații Buzești", Syncro Soft - Craiova

Putem considera că avem de împărțit numerele de la 1 la N în K grupuri, fiecare cu număr cunoscut de elemente, Mi.

Vom nota mai departe:

- C[n][k] ca fiind "combinări de n luate câte k" (numărul de submulțimi cu k elemente ale mulțimii {1, 2, …, n}).
- F[n] ca fiind n factorial (produsul primelor n numere naturale nenule).

Pentru a stabili componența primului grup avem C[n][M1] variante de alegere.

În cadrul acestui grup, pentru a determina numărul de variante de a permuta circular elementele sale, avem rezultatul: F[M1- 1] / 2.
Justificăm acest rezultat astfel: fiind vorba de permutări circulare ale mulțimii 1, 2, … M1, pentru a nu număra soluții de două ori, fixăm în notație o valoare pe prima poziție și apoi le permutăm pe

celelalte (avem (M1-1)! Variante pentru asta). Împărțim acest număr la 2 pentru că o aranjare a celor M1-1 valori nefixate este identică cu aceea în care le scriem pe acestea în ordine inversă.

Așadar, pentru a selecta elementele primului grup și a le aranja în toate modurile avem:
C[n][M1]*F[M1-1]/2
Acum reluăm procedeul pentru a stabili componența celui de-al doilea grup din elementele rămase. Avem C[n-M1][M2]*F[M2-1]/2 (după ce am ales M1 elemente în primul grup ne mai rămân n-M1 din care să le selectăm pe cele din al doilea grup).
Continuăm astfel și obținem rezultatul:

(C[n][M1]*F[M1-1]/2) * (C[n-M1][M2]*F[M2-1]/2) * (C[n-M1-M2][M3]*F[M3-1]/2) * …

Pentru determinarea valorii C[n][k] avem variantele:

(1) Formula n! / (k! * (n-k)!)
(2) Pentru mai multe puncte C[n][k] = C[n-1][k-1]+C[n-1][k] (triunghiul lui Pascal simulat pe o matrice sau pe un vector)
(3) Pentru punctaj maxim putem folosi metoda descrisă în materialul de mai jos:

https://cppi.sync.ro/materia/lectii/combinari.pdf
Este suficientă modalitatea prezentată pe pagina a doua a acestui material, cea bazată pe descompunerea în factori primi a factorialelor.