SOLUȚII OLIMPIADA NAȚIONALĂ DE INFORMATICĂ, ETAPA JUDEȚEANĂ
CLASELE 11-12

COMISIA ȘTIINȚIFICĂ

Problema 1: BIROCRAŢIE

Propusă de: prof. Zoltan Szabó

Subtask 1 (12 puncte): În cazul în care B are toate elementele pozitive, câștigul maxim se obține prin însumararea tuturor elementelor.

Subtask 2 (12 puncte): În cazul în care B are toate elementele egale și negative, trebuie să minimizăm pierderea, ceea ce reușim folosind doar deplăsări orizontale și verticale, renunțând complet la deplasările oblice, trecând astfel prin $2 * N - 1$ birouri. Adică, ținând cont, că toate elementele sunt egale, răspunsul la această cerință este $(2 * N - 1) * B[1][1]$

Subtask 3 (15 puncte): În cazul în care pe fiecare diagonală paralelă cu diagonala secundară elementele din B sunt egale, vom proceda astfel: Fiecare diagonală cu elemente egale pozitive se ia în întregime, iar pentru fiecare diagonală cu elemente negative se ia un singur element, astfel obținem suma maximă posibilă.

Subtask 4 (13 puncte): În cazul în care elementele de pe chenarul lui B sunt negative, iar celelalte elemente sunt pozitive, se va calcula suma tuturor elementelor pozitive din matrice (matricea fără chenar) la care se adaugă primul și ultimul element: $B[1][1]$, $B[N][N]$, respectiv în cazul diagonalelor de lungime 2, se va alege elementul cu valoarea mai mică.

Subtask 5 (13 puncte): În cazul în care toate elementele din B sunt egale în valoare absolută, elementele de pe chenar sunt pozitive, iar celelalte elemente sunt negative, vom trece peste cât mai multe elemente pozitive și vom evita toate elementele negative. Astfel se obține formula

$$(2 * N + 1) * B[1][1]$$

Subtask 6 (16 puncte): Problema se rezolvă cu programare dinamică în complexitate $O(N^3)$, Pentru fiecare element, valoarea maximă poate fi obținută doar de pe linia anterioară, adăugând elementul curent, de pe coloana anterioară adăugând elementul curent, sau de pe diagonala secundară, adăugând toate elementele, care formează secvența până la elementul curent, ceea ce necesită o complexitate liniară relativ la lungimea diagonalei, deci în ansamblu, N^2 elemente vor necesita $O(N^3)$ timp.

Subtask 7 (19 puncte): Problema se rezolvă cu programare dinamică în complexitate $O(N^2)$. Pentru fiecare element, valoarea maximă poate fi obținută doar de pe linia anterioară, adăugând elementul curent, de pe coloana anterioară adăugând elementul curent, sau de pe diagonala secundară, adăugând toate elementele care formează secvența până la elementul curent. Calculul elementelor de pe diagonale, prin parcurgerea ordonată și completă atât de la stânga la dreapta, cât și de la dreapta la stânga, va permite ca cele K elemente ale unei diagonale să le putem rezolva cumulativ în timp $O(K)$, adică pentru fiecare element vom avea un cost de timp $O(1)$, deci, în ansamblu problema va avea o complexitate de $O(N^2)$.

Problema 2: NESTEMATE

Propusă de: Stud. Mihaela Cismaru

Subtask 1 (11 puncte). Se verifică dacă cele două nestemate au cel puțin o proprietate ce coincide, caz în care declarăm că este necesară o singură transformare, în caz contrar se va afișa -1.

Subtask 2 (13 puncte). Se încearcă toate nestematele pentru a vedea dacă pot fi folosite ca stare intermediară sau dacă nestemata A poate fi transformată direct in nestemata B. Dacă nu este posibila transformarea in niciunul din aceste moduri se va afișa -1.

Subtask 3 (16 puncte). Se încearcă toate combinațiile de 2 nestemate intermediare, se încearcă doar cu o piatră intermediară și se încearcă și dacă este posibilă transformarea directă. Se afișează numărul minim de transformări iar în caz că nu este posibilă transformarea în niciunul din aceste moduri se va afișa -1.

Subtask 4 (10 puncte). Folosind o abordare de tip backtracking, încercăm să luăm toate posibilitățile de transformări succesive. Reținem lungimea minimă a unui lanț de transformări valid cât și dacă s-a reușit găsirea a cel puțin un astfel de lanț.

Subtask 5 (10 puncte). Modelăm relațiile dintre nestemate ca un graf. Pentru fiecare două nestemate se stabilește dacă au o proprietate în comun și prin urmare fiecare din aceste pietre poate fi transformată în cealaltă (transformarea este mereu bidirecțională). Se va crea o matrice de adiacență și se va căuta drumul minim de la nestemata A la nestemata B. Drumul minim poate fi găsit printr-un algoritm precum parcurgerea în lățime (BFS). Pentru această cerință este necesară o rezolvare în complexitatea $O(N^2)$.

Subtask 6 (13 puncte). Se vor genera muchiile într-o complexitate mai bună folosindu-ne de valorile prezente în nestemate. Pentru fiecare valoare se va genera o listă de nestemate ce conțin valoarea respectivă în configurație. Pentru fiecare valoare, toate nestematele din cadrul unei liste vor fi unite fiecare cu fiecare printr-o muchie. Se va avea grijă să nu se genereze de mai multe ori o muchie între aceleași două nestemate. Se va căuta drumul minim de la nestemata A la nestemata B printr-un algoritm precum parcurgerea în lățime (BFS). Deoarece la această cerință se garantează că o valoare apare la maxim 3 nestemate distincte, la un pas nu vor fi generate mai mult de 3 muchii. Dimensiunea grafului va fi rezonabilă pentru o complexitate de $O(N + MaxVal)$ unde MaxVal este cea mai mare valoare dintre proprietățile nestematelor.

Subtask 7 (27 puncte). Pentru soluția finală nu se va genera deloc graful deoarece numărul de muchii poate deveni foarte mare. Pentru fiecare valoare vom ține liste cu toate nestematele ce conțin valoarea respectivă. Vom folosi o abordare de parcurgere în lățime (BFS) pentru a găsi distanța minimă. La fiecare pas pornind de la o nestemată luăm toate proprietățile acesteia și explorăm doar listele proprietăților ce nu au mai fost întâlnite până în acel moment. Fiecare listă a unei proprietăți va fi parcursă exact o dată pe parcursul algoritmului. Parcurgem listele si adăugăm în coadă doar nestematele ce nu au mai fost parcurse până în acel moment. Această abordare se încadrează în complexitatea optimă $O(N + MaxVal)$ unde MaxVal este cea mai mare valoare dintre proprietățile nestematelor.

Problema 3: ACOPERIRE

Propusă de: Dl. Mihai Ciucu

Observația de bază a problemei este că dacă un segment nou acoperă un segment inițial, atunci trebuie să acopere mijlocul segmentului inițial. Dacă intervalul de acoperiri are lungimea minim jumătate din intervalul initial, este și garantat suficient. Echivalent ar fi să spunem că orice interval de lungime minim $L/2$ acoperă un interval inițial de lungime L dacă și numai dacă conține mijlocul intervalului inițial.

Ca să ajungem la concluzia de mai sus, considerăm un interval oarecare $[A, B]$. Dacă ne gândim care este segmentul minim cel mai din stânga care îl acoperă, care ar fi intervalul $[A, (A+B)/2]$, și cel mai din dreapta, adică $[(A+B)/2, B]$, vedem că doar elementul din mijloc este intersecția lor.

De aici, putem vedea că un interval de acoperire dintr-o soluție validă poate fi translatat fără a-i afecta corectitudinea până când unul din capete coincide cu unul din mijloacele intervalelor acoperite inițial. Facem asta pentru a putea genera toate intervalele de acoperire pornind de la puncte definite de segmentele inițiale.

Pentru soluția finală căutăm binar lungimea minimă a celui mai lung interval, folosind întrebări de forma: "Dacă toate intervalele de acoperire au lungima L, am putea acoperi cu cel mult K intervale mulțimea celor initiale?"

Întrebarea nouă este mai simplă, și o răspundem printr-o parcurgere a intervalelor inițiale, ordonate după mijloacele lor: Începem un interval de lungime L în primul mijloc și considerăm acoperite toate segmentele cu mijlocul conținut în acesta. Când întâlnim un segment nou care nu este acoperit de ultimul interval adăugat, înseamnă că trebuie să începem un nou interval în mijlocul intervalului inițial. Complexitatea pentru pasul acesta este $O(N)$ odată ce intervalele sunt sortate la început după mijloc.

Pentru soluția minimă lexicografic tot ce trebuie făcut e să vedem că dacă intervalele trebuie să înceapă cât mai din stânga, este echivalent cu a vrea ca intervalele să se termine cât mai în stânga, și deci doar facem algoritmul în sens invers după mijloace (pornim ultimul interval în ultimul mijloc, etc.).

Pentru a nu fi erori de precizie, se dublează în implementare capetele intervalelor, astfel încât mijloacele să fie întregi. La afișare se poate face cu: cout « setprecision(2) « (x / 2.0) (fără fixed) sau tratând explicit numere pare sau impare, cu caz particular pentru afișarea lui -1.

Soluții partiale:

Subtask 1 (10 puncte). Pentru $N = 1$, doar trebuie afișat intervalul de la capătul stânga la mijlocul intervalului.

Subtask 2 (10 puncte). Când sunt două intervale disjuncte, pentru K=2 doar se face $[start, mijloc]$ pentru fiecare, iar pentru $K = 1$ sunt două cazuri, ca la subtaskul 3.

Subtask 3 (20 puncte). Când $K = 1$ soluția tot timpul este ori $[primulMijloc, ultimulMijloc]$, ori se termină în ultimul mijloc și trebuie văzut pentru fiecare interval în parte unde poate începe cel mai târziu.

Subtask 4 (20 puncte). Când toate intervalele nu se intersectează, ele pot fi sortate oricum, căci bănuim că unii concurenți instinctiv le sortează după unul din capete și apoi încearcă un greedy. Problemele de genul de obicei implică o ordonare și parcurgere a evenimentelor, doar aici trebuia văzut că evenimentele interesante sunt mijloacele.

Echipa

Problemele pentru această etapă au fost pregătite de:

- Prof. Zoltan Szabó
- Prog. Mihai Ciucu
- Stud. Mihaela Cismaru
- Prof. Mihai Bunget
- Asist. drd. Andrei Constantinescu
- Prog. Adrian Budău
- Prog. Daniel Posdărăscu
- Stud. Vlad-Mihai Bogdan
- Stud. Matei Tinca
- Stud. Bogdan Sitaru
- Stud. Luca Metehău
- Stud. Alexandru Lorinz
- Stud. Liviu Silion