DESCRIEREA SOLUȚIILOR, OLIMPIADA JUDEȚEANĂ DE INFORMATICĂ CLASA A VIII-A

COMISIA ȘTIINȚIFICĂ

Problema 1: Pelican

Propusă de: prof. Nistor Moț, Școala Gimnazială „Dr. Luca", Brăila

Problema admite mai multe abordări. O posibilă abordare este de a citi succesiv comenzile pelicanului şi de a executa fiecare comandă pentru fiecare dintre cele P raţe. Este evident o abordare corectă, dar care va depăşi timpul de execuţie. Totuşi această abordare poate fi îmbunătăţită bazându-ne pe următoarele observaţii:

(1) Dacă în fişierul de ieşire există o comandă Z, atunci vom executa această comandă pentru toate raţele. Dacă există mai multe comenzi Z, doar ultima contează, deci doar aceasta va fi executată. Toate comenzile A până la comanda Z executată pot fi ignorate, doar comenzile R contează. În plus, observăm că nu trebuie să executăm fiecare comandă R separat, putem cumula numărul de grade şi putem executa o singură rotaţie finală pentru toate raţele.

(2) Ideea de cumulare poate fi utilizată mai departe. Pentru a reduce numărul de comenzi executate de toate raţele, putem lucra pe secvenţe de comenzi A, respectiv secvenţe de comenzi R. Pentru fiecare secvenţă de comenzi identice vom cumula nr şi la finalul secvenţei executăm o deplasare/rotaţie cumulată pentru toate raţele. Desigur, pentru deplasări lucrăm modulo N, iar pentru rotaţii lucrăm modulo 4.

Aceste observaţii pot conduce la un punctaj bun, dar pentru 100 de puncte trebuie observat că două raţe care au iniţial aceeaşi orientare se vor deplasa „solidar", adică păstrează aceeaşi poziţie relativă. Dacă avem două raţe cu aceeaşi orientare în poziţiile iniţiale (i_1, j_1) şi (i_2, j_2), comanda A nr va duce ambele raţe în poziţii care păstrează aceeaşi distanţă între ele. De exemplu, pentru orientarea Sud cele două raţe vor ajunge în poziţiile $(i_1 + nr, j_1)$ şi respectiv $(i_2 + nr, j_2)$. Această observaţie ne permite să efectuăm comenzile doar cu 4 raţe „virtuale" ce pornesc din poziţia (0,0) având orientarile N, E, S, respectiv V.

La final, pentru fiecare raţă vom determina poziţia sa finală în funcţie de poziţia raţei virtuale care avea iniţial aceeaşi orientare cu raţa respectivă. Dacă raţa virtuală ajunge în poziţia finală (x, y), atunci o raţă care iniţial avea aceeaşi orientare cu cea virtuală şi era plasată în poziţia (i_1, j_1) va ajunge în poziţia finală $(i_1 + x, j_1 + y)$. Evident, toate deplasările se fac modulo N. La fel, rotaţiile se execută modulo 4.

Mai mult, dacă dorim, putem renunţa la a calcula poziţia finală pentru toate cele 4 direcţii şi să efectuăm mutările doar pentru o singură direcţie, celelalte putând fi deduse prin rotaţii cu câte 90 de grade. De exemplu, considerând că o raţă ce porneşte din poziţia 0,0 având orientarea iniţială N ajunge în poziţia finală (x, y), atunci raţa din pozitia (i_1, j_1) ajunge:

- dacă are orientarea iniţială N, în poziţia finală $(i_1 + x, j_1 + y)$
- dacă are orientarea iniţială S, în poziţia finală $(i_1 - x, j_1 - y)$
- dacă are orientarea iniţială E, în poziţia finală $(i_1 + y, j_1 - x)$
- dacă are orientarea iniţială V, în poziţia finală $(i_1 - y, j_1 + x)$.

Problema 2: Strips

Propusă de: prof. Emanuela Cerchez, Colegiul Naţional „Emil Racoviţă" Iaşi

Soluţia 1 — 40 de puncte. Pentru $N \leq 10^6$, putem utiliza un vector pentru a reprezenta culorile benzilor plasate pe tabla de joc (0 pentru poziţie neocupată, 1 pentru o poziţie ocupată de o bandă roşie, respectiv 2 pentru o poziţie ocupată de o bandă verde). Citim succesiv mutările. Pentru fiecare mutare verificăm dacă este validă. Dacă da, plasăm pe tabla de joc o bandă în poziţia respectivă. Dacă nu, contorizăm un punct penalizare pentru jucătorul care este la rând. Pentru cerinţa 2 este suficient să parcurgem tabla de joc şi să determinăm lungimea maximă a unei secvenţe formată numai din 1, respectiv lungimea maximă a unei secvenţe formată numai din 2.

Soluţia 2 — 100 de puncte. Pe măsură ce plasăm benzi pe tabla de joc formăm practic zone continue de aceeaşi culoare, care nu se suprapun. În loc să reţinem tabla de joc, vom reţine pentru fiecare jucător zonele obţinute din benzile plasate de acesta pe tabla de joc. O zonă va fi reprezentată prin cele două extremităţi (poziția de început şi poziția de sfârșit a zonei).

```
struct zona { int inc, sf; };
```

Considerăm că jucătorul 0 este Ana şi jucătorul 1 este Bogdan.

Structurile de date utilizate sunt:

```
zona Z[2][NRMAX]; //zonele celor doi jucatori
int nrz[2];       //numarul de zone pentru fiecare jucator
int p[2];         //penalizarea fiecarui jucator
int lgmax[2];     //lungimea maxima a unei zone pentru fiecare jucator
```

$NRMAX$ este numărul maxim de zone ale unui jucător (se garantează că la finalul jocului $NRMAX$ nu depășește 5000, dar pe parcursul jocului poate fi mai mare).
Vom reţine zonele în ordinea crescătoare a pozițiilor de început.

Vom citi succesiv mutările celor doi jucători. La o mutare verificăm mai întâi validitatea acesteia. Să notăm *poz* poziţia corespunzătoare mutării curente şi cu *cine* jucătorul care este la mutare. Evident, adversarul va fi $1 - cine$.

Pentru a verifica validitatea vom căuta pe tabla adversarului cea mai mică poziţie (*unde*) pentru care $poz \leq Z[adversar][unde].inc$. Zonele fiind sortate, vom face o căutare binară.

Dacă o bandă plasată în poziţia *poz* se intersectează cu sau de lipește de cea de pe poziţia *unde* (la dreapta) sau cu cea de pe poziţia $unde - 1$ (la stânga) atunci mutarea nu este validă.

În cazul în care mutarea este validă, plasăm o bandă pe tabla jucătorului care este la mutare. Pentru aceasta facem din nou o căutare binară, de data aceasta în zonele jucătorului care este la mutare şi determinăm cea mai mică poziţie (*unde*) pentru care $poz \leq Z[cine][unde].inc$.

Pot apărea următoarele situaţii:

(1) banda poate fi alipită zonei $Z[cine][unde]$ (dacă această zonă există şi $poz + L - 1 \geq Z[cine][unde].inc - 1$);
(2) banda poate fi alipită zonei $Z[cine][unde-1]$ (dacă această zonă există şi $poz \leq Z[cine][unde-1].sf + 1$);
(3) în urma alipirii zonele $Z[cine][unde-1]$ şi $Z[cine][unde]$ pot fi şi ele alipite, formând astfel o singură zonă (le alipim şi eliminăm una dintre ele);
(4) dacă nicio alipire nu este posibilă, inserăm o nouă zonă cu începutul *poz* şi sfârşitul $poz + L - 1$ în poziţia *unde* (astfel zonele rămânând sortate).

Eficienţa timp a acestui algoritm este afectată de situaţiile 3 sau 4 (eliminarea, respectiv inserarea fiind liniară).

Utilizarea unor structuri de date avansate (de exemplu, structura de date set din STL) poate optimiza acest pas, dar o astfel de implementare depăşeşte nivelul clasei a VIII-a și nu este necesară pentru a obține 100 de puncte.

Este posibilă o implementare bazată pe aceeaşi idee, asociind fiecărei zone culoarea acesteia şi reţinând zonele sortate după extremitatea iniţială într-un singur vector. Dar în acest caz dimensiunea vectorului este mai mare şi timpul necesar pentru eliminare/inserare creşte.

Echipa

Problemele pentru această etapă au fost pregătite de:

- prof. Emanuela Cerchez, Colegiul Național ,,Emil Racoviță" Iaşi
- prof. Nistor Moţ, Școala Gimnazială ,,Dr. Luca" Brăila
- prof. Ionel-Vasile Piţ-Rada, Colegiul Naţional ,,Traian", Drobeta-Turnu Severin
- lector dr. Paul Diac, Facultatea de Informatică, Universitatea ,,Alexandru Ioan Cuza" Iaşi
- prof. Filonela Bălaşa, Colegiul Național ,,Grigore Moisil" Bucureşti
- prof. Lucia Miron, Colegiul Național ,,Costache Negruzzi" Iaşi
- stud. Ioan Cristian Pop, Universitatea Politehnica București
- stud. Stelian Chichirim, Universitatea București
- prof. Daniel Popa, Liceul Teoretic ,,Aurel Vlaicu" Orăştie
- stud. Radu Muntean, ETH Zurich