

Descrierea Soluțiilor  
Concursul Național “InfoPro”, Runda 3  
Grupa C1

## 1 Problema Potter

*Propunători: prof. Dana Lica - Centrul Județean de Excelență Prahova, Ploiești;  
stud. Florian Ușurelu - Universitatea din București/Universitatea Constantin  
Brâncuși*

Pentru a calcula numărul de modalități pe care Harry Potter le are la dispoziție pentru a așeza cei  $K$  pioni pe tablă, vom determina inițial numărul de celule sigure. Având în vedere limita de memorie vom folosi 2 vectori caracteristici, pentru a marca de la citire, liniile și coloanele nesigure.

Ne aflăm în fața unei probleme de forma: *În câte moduri pot așeza  $K$  pioni în  $X$  celule sigure* Ne vom folosi de metoda *stars and bars* și ne rămâne de determinat valoarea numărului de combinări  $\binom{K+X-1}{X-1}$

Pentru soluția de 100 de puncte pentru calculul combinărilor se va folosi inversul modular. Considerând  $Mod$  un număr prim, numărul de *combinări de  $N$  luate câte  $K$  modulo  $Mod$*  se poate calcula astfel:

$$\binom{N}{K} \pmod{Mod} \equiv N! \cdot (K!)^{Mod-2} \cdot (N-K)!^{Mod-2} \pmod{Mod}$$

Ridicările la putere se vor efectua în timp logaritmic.

## 2 Problema Rectangles

*Propunător: stud. Stelian Chichirim - Universitatea din București*

### Subtask 1 $O(N^2)$ :

Vom parcurge fiecare subsecventa in parte si vom tine primele 2 maxime. Pentru fiecare subsecventa adunam la raspuns produsul celor 2 maxime. Pentru a parcurge toate subsecventele ne fixam pozitia de inceput a subsecventei si ne extendem pana la sfarsitul sirului. Astfel, complexitatea de timp este  $O(N^2)$ .

### Subtask 2 $O(N * (\text{numarul de elemente distincte}))$ :

Notam cu  $K$  numarul de elemente distincte din sir.

Ne vom fixa pozitia de inceput a unei subsecvente. Putem observa ca, cu cat extendem subsecventa, cele 2 maxime se pot schimba de maxim  $2 * K$  ori, deoarece de fiecare data cand apare un numaru nou se poate schimba primul sau al doilea maxim si cand mai apare inca o data acelasi numar se poate schimba al doilea maxim, deci de la a treia aparitie a unui numar, acesta nu mai afecteaza valoarea celor 2 maxime.

Astfel, daca ne fixam inceputul subsecventei de la pozitia 1 la pozitia  $N$ , putem mentine mereu pentru fiecare numar distinct din sir primele 2 aparitii ale acestuia care sunt dupa pozitia fixata (inclusiv pozitia fixata). Vom mentine toate aceste pozitii sortate crescator (cand ne mutam cu o pozitie mai departe putem mentine aceste pozitii sorate crescator in  $O(K)$ ) si vom calcula in  $O(K)$  parcurgand aceste pozitii suma dreptunghiurilor tuturor subsecventelor care incep in pozitia respectiva. In final, complexitatea de timp este  $O(N * K)$ .

### Subtask 3 $O(N * \log(N))$ :

Ideea pentru solutia de  $O(N * \log(N))$  si de  $O(N)$  este sa ne fixam fiecare pozitie din sir ca fiind al doilea maxim din subsecventele care il contin.

Vom calcula urmatoarele 4 informatii pentru fiecare pozitie din sir:

- $st1_i$  = prima pozitie din stanga lui  $i$  care are valoarea strict mai mare ca valoarea  $V_i$
- $st2_i$  = a doua pozitie din stanga lui  $i$  care are valoarea strict mai mare ca valoarea  $V_i$
- $dr1_i$  = prima pozitie din dreapta lui  $i$  care are valoarea mai mare sau egal ca valoarea  $V_i$
- $dr2_i$  = a doua pozitie din dreapta lui  $i$  care are valoarea mai mare sau egal ca valoarea  $V_i$

Dupa ce am calculat aceste 4 siruri. Pentru o pozitie  $poz$  fixata vom aduna la raspuns valorile tuturor subsecventelor care au al doilea maxim egal cu valoarea de pe aceasta pozitie. Astfel, daca:

- $st2_i$  exista adunam la raspuns  $V_{st1_{poz}} * V_{poz} * (st1_{poz} - st2_{poz}) * (dr1_{poz} - poz)$
- $dr2_i$  exista adunam la raspuns  $V_{dr1_{poz}} * V_{poz} * (dr2_{poz} - dr1_{poz}) * (poz - st1_{poz})$

Se pot calcula in multe moduri aceste 4 siruri in  $O(N * \log N)$ . Un mod ar fi sa calculam mai intai  $st1$  si  $dr1$  folosind o stiva pentru ambele parti. Pentru  $st2$  ne folosim de pozitia data de  $st1$  si cautam binar prima pozitie in care maximul de pe intervalul care incepe in aceasta pozitie si se termina in  $st1 - 1$  este mai mare ca valoarea de pe pozitia fixata. Similar se face si pentru  $dr2$ . Daca folosim *RMQ* pentru a afla maximul de pe un interval in  $O(1)$ , complexitatea finala va fi  $O(N \log N)$ .

#### Subtask 4 $O(N)$ :

Solutia de  $O(N)$  este exact la fel ca solutia de  $O(N * \log(N))$ , doar ca vom calcula sirurile  $st1, st2, dr1, dr2$  in  $O(N)$ . Vom folosi o stiva pentru a calcula  $st1$ , practic, parcurgem numerele de la sfarsit la inceput si tinem intr-o stiva numerele, sortate descrescator, pentru care inca nu le-am gasit prima pozitie din stanga cu valoare mai mare. Pentru o pozitie  $i$  vom elimina toate elementele din stiva care au valoarea mai mica decat  $V_i$  ( $st1$  pentru aceste numere este  $i$ ), apoi vom adauga acest numar in varful stivei. Pentru a calcula si  $st2$  vom mai tine o stiva in care vom adauga numerele eliminate din prima stiva in ordine descrescatoare (in ordinea inversa a eliminarii). Acum, pentru o pozitie  $i$ , mai intai eliminam toate elementele cu valoarea mai mica din a doua stiva ( $st2$  pentru aceste numere este  $i$ ), apoi eliminam din prima stiva si le adaugam pe cele eliminate in a doua stiva. Pentru  $dr1$  si  $dr2$  se face similar. Cum fiecare element va fi bagat si eliminat din fiecare stiva maxim o data, complexitatea finala va fi  $O(N)$ .

### 3 Problema Radar

*Propunător: stud. Bogdan Iordache - Universitatea din București*

Înainte de a trece la rezolvarea propriu-zisă a problemei, vom face câteva observații menite să ușureze implementarea:

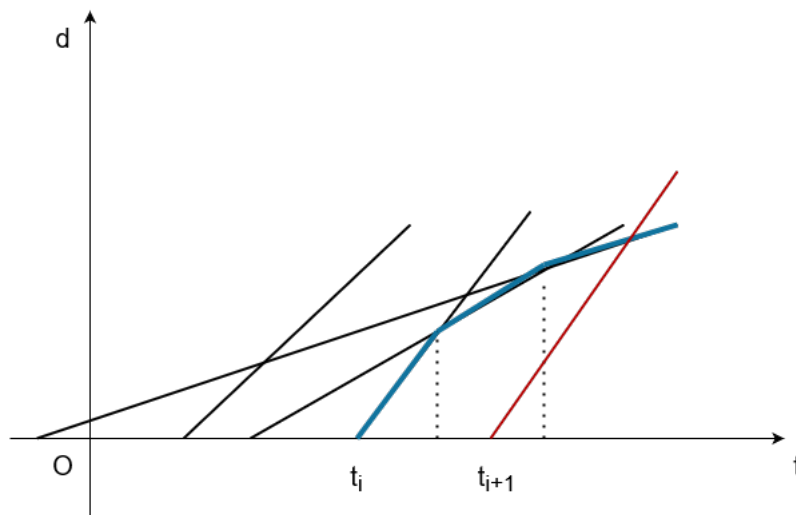
- vitezele pot fi considerate de la început în modul (în problema noastră direcția de deplasare a unei mașini este irelevantă)
- vom reindexa toate mașinile astfel încât acestea să fie sortate crescător după timpul la care au fost detectate
- după reindexare, putem observa că pentru a răspunde la o interogare  $t$ , cu  $t$  cuprins între timpii de detectare a două mașini consecutive ( $t_i$  și  $t_{i+1}$ ), ne interesează cum sunt aranjate la acel moment de timp doar primele  $i$  mașini. Astfel, vom răspunde la interogări "offline": sortăm de la început toate interogările, iar după ce am procesat mașina  $i$  răspundem la toate interogările cu  $t_i \leq t < t_{i+1}$ .

Pentru a rezolva problema, cel mai ușor ne este să o analizăm grafic. Dacă reprezentăm distanța fiecărei mașini față de radar ca funcție de timp, obținem câte o semidreaptă cu pantă pozitivă ce are originea în punctul  $t_i$  de pe axa OX.

Presupunem acum că am procesat deja primele  $i$  semidrepte, iar acum dorim să observăm ce informații ne sunt necesare pentru a răspunde la interogările cuprinse în intervalul  $[t_i, t_{i+1})$ . Vedem că mașina  $i$  va fi cea mai apropiată de radar începând de la momentul  $t_i$  până când va depăși pentru prima oară o altă mașină, apoi acesta din urmă va fi cea mai apropiată pentru un interval de timp, până când și ea va depăși mai departe altă mașină. În final va mai rămâne doar cea mai lentă mașină care va fi cea mai apropiată de radar până la momentul  $+\infty$ . Spunem că fiecare mașină dintre cele menționate anterior "domină" un interval de timp. Dacă cunoaștem aceste intervale de timp este suficient ca pentru o interogare  $t$  să căutăm binar care este intervalul dominat care conține momentul  $t$  și astfel găsim cea mai apropiată mașină de radar la acel moment.

Mai rămâne de văzut cum obținem aceste intervale dominate pe măsură ce procesăm semidreptele. La prima semidreaptă observăm că avem un singur interval dominat:  $[t_1, +\infty)$ . Acum să presupunem că avem determinate aceste intervale pentru primele  $i$  semidrepte (evident putem avea mai puțin de  $i$  intervale întrucât unele semidrepte au devenit irelevante - nu vor mai fi niciodată cele mai apropiate de radar). Atunci când vrem să procesăm semidreapta  $i + 1$  aceasta va elimina toate intervalele dominate care se terminau înainte de poziția  $t_{i+1}$ , dar și pe toate pentru care în capătul drept, semidreapta nouă corespunde unei distanțe mai mici decât semidreapta ce domina acel interval (cu alte cuvinte, se elimină toate intervalele care au devenit irelevante sau pentru care noua semidreaptă este mai bună decât cele vechi). Astfel se elimină un prefix al intervalelor dominate descoperite la pasul  $i$ . Rămâne acum să adăugăm intervalul

dominat de semidreapta  $i + 1$ , iar pentru aceasta avem nevoie să o intersectăm cu prima semidreaptă dintre cele rămase după eliminare. Dacă acestea se intersectează la momentul de timp  $x$ , atunci intervalul dominat de semidreapta  $i + 1$  este  $[t_{i+1}, x)$ . Întrucât ne interesează doar momente întregi de timp putem păstra doar partea întreagă a lui  $x$ .



Pentru a gestiona optim aceste intervale vom folosi o stivă. Pentru a calcula complexitatea trebuie să avem în vedere: sortarea semidreptelor și a interogărilor de la început, procesarea intervalelor dominate care se va face liniar datorită stivei, rezolvarea interogărilor făcând căutări binare. Obținem astfel complexitatea  $O(N \log N + Q(\log Q + \log N))$ .

**Echipa** care a pregătit setul de probleme pentru această rundă a fost formată din:

- prof. Daniela Lica, Centrul Județean de Excelență Prahova, Ploiești
- prof. Flavius Boian, Colegiul Național "Spiru Haret" Târgu Jiu
- prof. Florentina Ungureanu, Inspectoratul Școlar Județean Neamț/ Colegiul Național de Informatică Piatra-Neamț
- prof. Marius Nicoli, Colegiul Național "Frații Buzești" Craiova
- prof. Octavian Dumitrașcu, Colegiul Național "Dinicu Golescu" Câmpulung Muscel
- stud. Andrei Arhire - Universitatea Alexandru Ioan Cuza
- stud. Bogdan Iordache - Universitatea din București
- stud. Florian Ușurelu - Universitatea din București/  
Universitatea Constantin Brancusi
- stud. Stelian Chichirim - Universitatea din București