OLIMPIADA NAȚIONALĂ DE INFORMATICĂ, ETAPA JUDEȚEANĂ
CLASA A VII-A
DESCRIEREA SOLUȚIILOR

COMISIA ȘTIINȚIFICĂ

Problema 1: Parking

Propusă de: prof. Florentina Ungureanu, Colegiul Național de Informatică, Piatra Neamț

Vom reprezenta o mașină ca o structură (struct) cu 4 câmpuri: linia și coloana pe care se află mașina, orientarea mașinii (0 pentru orizontal, 1 pentru vertical) și numărul seriei în care mașina a ieșit, respectiv 0 dacă mașina este încă în parcare. Vom reține mașinile într-un vector vm, astfel vom putea identifica în continuare fiecare mașină prin poziția sa în vector (un număr între 1 și q). O ieșire din parcare o vom identifica prin poziția sa (linia și coloana pe care se află). Vom reține ieșirile într-un vector vi.

Pentru a identifica mașinile care pot ieși din parcare putem parcurge vectorul de mașini și pentru fiecare mașină verificăm dacă poate ieși din parcare în seria curentă (ajunge la o ieșire deplasându-se în una dintre cele două direcții de mișcare posibile) și dacă da, o reținem în seria curentă. O astfel de abordare obține un punctaj substanțial, dar pe testele mari va obține depășire de timp, datorită numărului mare de mașini, complexitatea fiind $\mathcal{O}(q \cdot nrserii)$.

O abordare mai eficientă ar fi să parcurgem ieșirile și să verificăm, pentru fiecare ieșire, dacă există o mașină care poate ieși în seria curentă utilizând ieșirea respectivă. Apar 4 cazuri:

(1) dacă ieșirea se află pe coloana 0, această ieșire poate fi utilizată doar de prima mașină de pe linia pe care se află ieșirea (dacă aceasta este poziționată orizontal);
(2) dacă ieșirea se află pe coloana $m+1$, această ieșire poate fi utilizată doar de ultima mașină de pe linia pe care se află ieșirea (dacă aceasta este poziționată orizontal);
(3) dacă ieșirea se află pe linia 0, această ieșire poate fi utilizată doar de prima mașină de pe coloana pe care se află ieșirea (dacă aceasta este poziționată vertical);
(4) dacă ieșirea se află pe linia $n+1$, această ieșire poate fi utilizată doar de ultima mașină de pe coloana pe care se află ieșirea (dacă aceasta este poziționată vertical).

Când analizăm cazurile (1) și (2), respectiv (3) și (4) trebuie să avem grijă la situația în care există o singură mașină pe linie, respectiv pe coloană, situație în care această mașină este atât prima, cât și ultima (deci trebuie să evităm să scoatem din parcare aceeași mașină de două ori și să contorizăm eronat mașinile care ies din parcare). Pentru a trata simplu și eficient cele 4 cazuri vom construi două matrici auxiliare ml și mc, având n linii și m coloane, respectiv m linii și n coloane. Pe linia i ($1 \leq i \leq n$) în matricea ml se află lista mașinilor de pe linia i a parcării, sortate crescător după coloană. Pe linia j ($1 \leq j \leq m$) a matricii mc se află lista mașinilor de pe coloana j a parcării, sortate crescător după linie. Ca urmare pentru ieșirile de pe coloana 0 verificăm doar prima mașină de pe linia corespunzătoare din ml, pentru ieșirile de pe coloana $m+1$ doar ultima mașină de pe linia corespunzătoare din ml. Similar, pentru ieșirile de pe linia 0 verificăm doar prima mașină de pe coloana corespunzătoare din mc, iar pentru ieșirile de pe linia $n+1$, doar ultima mașină de pe coloana corespunzătoare din mc.

Pentru cerința 1 efectuăm ieșirea mașinilor din prima serie. Pentru cerința 2 vom efectua ieșirea mașinilor în serii succesive, până când în seria curentă nu mai iese nicio mașină din parcare. Pentru ieșirea mașinilor dintr-o serie vom proceda în două etape:

(1) Parcurgem ieșirile și identificăm mașinile care pot ieși în seria curentă, reținând aceste mașini într-un vector și memorând în structura care reprezintă mașina numărul seriei curente.
(2) Parcurgem vectorul care conține mașinile care pot ieși în seria curentă și eliminăm aceste mașini din matricile ml și mc.

Este obligatorie ieșirea mașinilor dintr-o serie în două etape, în alt mod nu putem simula simultaneitatea ieșirii mașinilor din parcare. Dacă vom elimina mașinile succesiv pe măsură ce identificăm că pot ieși, este posibil să iasă din parcare în seria curentă mașini care nu ar fi trebuit să iasă (de exemplu mașina a poate ieși din parcare în seria curentă; mașina a blochează mașina b; dacă eliminăm imediat mașina a, atunci când ajungem la mașina b aceasta nu va mai fi blocată și va ieși și ea în aceeași serie cu a, ceea ce este evident incorect).

Complexitatea algoritmului: pentru fiecare serie, în etapa 1 se parcurg toate ieșirile pentru a identifica mașinile ($\mathcal{O}(K)$), iar în etapa 2 trebuie să eliminăm mașinile care ies în seria curentă din ml și mc (din una dintre matrici se poate elimina în $\mathcal{O}(1)$, dar pentru cealaltă matrice este necesară o căutare secvențială a mașinii urmată de eliminarea acesteia, deci complexitatea va fi $\mathcal{O}(lg)$, unde cu lg notăm numărul de mașini existente pe linia/coloana respectivă (deci putem spune că, în cazul cel mai defavorabil, eliminarea unei mașini din ml, respectiv mc se realizează în timp liniar în funcție de n, respectiv de m).

Există algoritmi mai eficienți de rezolvare a acestei probleme, dar necesită cunoștințe care depășesc programa clasei a VII-a.

Problema 2: Ron

Propusă de: prof. Filonela Bălașa, Colegiul Național "Grigore Moisil", București

Observație inițială: O creangă de lungime lg așezată pe riglă cu capătul din stânga în dreptul numărului poz reprezintă, din punct de vedere matematic, intervalul $[poz, poz + lg - 1]$.

Cerința 1: Pentru rezolvarea cerinței 1, este necesar să determinăm numărul maxim de numere fermecate aflate în unul dintre intervalele corespunzătoare crengilor de soc. Matematic, numărul pătratelor numerelor prime dintr-un interval $[x, y]$ este egal cu numărul numerelor prime din intervalul $[\sqrt{x}, \sqrt{y}]$.

Vom utiliza Ciurul lui Eratostene pentru a determina numerele prime mai mici sau egale cu $\sqrt{2000000000}$, apoi vom construi un vector auxiliar a în care vom memora pentru fiecare număr natural x, numărul numerelor prime mai mici sau egale cu x. Numărul pătratelor numerelor prime din intervalul $[x, y]$ este $a[\sqrt{y}] - a[\sqrt{x-1}]$.

Vom citi succesiv descrierea fiecărei crengi de soc, vom determina numărul numerelor fermecate acoperite de fiecare creangă și vom reține maximul.

Cerința 2: Pentru rezolvarea cerinței 2 este necesar să determinăm numărul de intervale obținute în urma reuniunii celor n intervale corespunzătoare crengilor de soc.

Vom ordona intervalele crescător după extremitatea inițială (capătul din stânga). Vom parcurge intervalele în ordine și vom reuni toate intervalele care pot fi reunite în unul singur (să numim rezultatul reuniunii interval-reuniune). La fiecare pas comparăm capătul din stânga al intervalului curent (st_i) cu capătul din dreapta al intervalului-reuniune curent (să-l notăm drR). Dacă $st_i \leq 1 + drR$ (cele două crengi se suprapun sau se ating) atunci intervalul curent poate fi adăugat la intervalul-reuniune curent (adică $drR = dr_i$, dacă $dr_i > drR$). În caz contrar, intervalul-reuniune curent nu mai poate fi extins, ca urmare creăm un nou interval-reuniune, pe care îl inițializăm cu intervalul curent.

Problema admite numeroase abordări pentru punctaje parțiale. De exemplu, pentru valorile mici ale lui poz și lg se poate utiliza un vector caracteristic în care vor fi marcate cu 1 pozițiile ocupate de crengile de soc, problema reducându-se astfel la determinarea numărului de secvențe formate numai din 1.

Echipa

Problemele pentru această etapă au fost pregătite de:

- Prof. Emanuela Cerchez, Colegiul Național "Emil Racoviță" Iași
- Prof. Filonela Bălașa, Colegiul Național "Grigore Moisil" București
- Prof. Florentina Ungureanu, Colegiul Național de Informatică Piatra Neamț
- Prof. Claudiu-Cristian Gorea-Zamfir, Liceul de Informatică "Grigore Moisil" Iași
- Stud. Dumitru Ilie, Facultatea de Matematică-Informatică, Universitatea București
- Stud. Ioan-Cristian Pop, Universitatea Politehnica București
- Prof. Nistor Moț, Școala "Dr. Luca" Brăila
- Lector dr. Paul Diac, Facultatea de Informatică, Universitatea "Al. I. Cuza" Iași
- Prof. Radu Vișinescu, Colegiul Național "Ion Luca Caragiale" Ploiești
- Stud. Theodor-Gabriel Tulba-Lecu, Universitatea Politehnica București