Olimpiada Națională de Informatică, Clasa a VIII-a Descrierea soluțiilor

Comisia științifică

April 15, 2025

Problema 1. Mușuroi

Propusă de: stud. Andrei Boacă, Facultatea de Informatică, Universitatea "Alexandru Ioan Cuza" Iași

Soluție O(Q*N*M*K)

Se poate simula drumul pe care îl parcurge furnica, oprindu-ne în momentul în care am ajuns a K-a dată la celula de final. Această soluție se încadrează în timp pentru N, M, $Q \leq 100$ și $K \leq 5$.

Soluție O(Q*N*M)

Observația principală este că, având în vedere că furnica nu poate ieși din matrice, va putea parcurge o infinitate de pași prin matrice. Prin urmare, există o celulă prin care va trece de cel puțin două ori. Să notăm prima celulă pe care o va întâlni furnica a doua oară cu P_1. Este evident că, dacă pornind din celula $P1$ am putut ajunge încă o dată în celula $P1$ avem de fapt o secvență periodică de celule prin care trece furnica. Vom nota lungimea acestei secvențe periodice cu L. În consecință, orice drum al unei furnici care pornește din celula A va fi de forma A, A_2, A_3,..., P_1, P_2, P_3,... P_L, P_1, P_2, P_3, ... Pentru soluția O($Q * N * M$) putem simula drumul furnicii până când întâlnește a doua oară o celulă, moment în care știm că am găsit celula P_1. Astfel, celulele care au apărut între cele două apariții ale lui P_1 vor face parte din perioadă. Deci, pentru orice celulă din perioadă, dacă aceasta apare prima dată la momentul T, atunci va apărea următoarea dată la momentul $T + L$, a treia oară la momentul $T + 2 * L$, ș.a.m.d. Deci, putem calcula ușor timpul după care celula va apărea a K-a oară.

Soluție O(Q + N*M)

Pornind de la soluția anterioară, putem observa că nu este necesar ca de fiecare dată să parcurgem tot drumul până la celula P_1 pentru fiecare întrebare. Mai mult, putem precalcula celula P_1, distanța până la aceasta și lungimea perioadei pentru fiecare celulă din matrice. În plus, vom reține și distanța fiecărei celule ce face parte dintr-o perioadă către o celulă pe care o vom

desemna celulă de start pentru acea perioadă (acea celulă poate fi aleasă arbitrar). Vom proceda în felul următor pentru a afla toate aceste informații.

Considerăm inițial toate celulele ca fiind nevizitate. Iterăm prin celule într-o ordine arbitrară, iar dacă aceasta nu a fost vizitată, vom începe să parcurgem drumul pornind din ea. Dacă la un moment dat înâlnim pe drum o celulă vizitată într-o iterație anterioară, atunci știm că pentru acea celulă am calculat deja informațiile necesare, prin urmare putem distribui acele informații corespunzător și către celulele vizitate în iterația curentă. Altfel, dacă întâlnim o celulă deja vizitată în iterația curentă, înseamnă că am găsit celula P_1 și lungimea perioadei pentru toate celulele parcurse în iterația curentă, deci le putem din nou actualiza corespunzător. Acum, putem răspunde la fiecare întrebare în O(1), deoarece știm lungimea prefixului până la celula P_1, precum și lungimea L.

Problema 2. Notwen

Propusă de: stud. Dumitru Ilie, Facultatea de Matematică-Informatică, Universitatea București

Soluții parțiale

Se poate simula operația descrisă în enunț utilizând operații cu numere mari.

Soluția oficială

Observație: Reprezentăm numerele în baza 2 și observăm cum se modifică distanța.

Câteva exemple:

$$\begin{aligned} 14 &= 1110_2 \to 1101_2 \to 1011_2 \to 0111_2 \to -1 \to 1 \\ &= 0001_2 \to 0000_2 \\ &= 0 \end{aligned}$$

$$\begin{aligned} 8 &= 1000_2 \to 0111_2 \to 0101_2 \to 0001_2 \to -7 \to 7 \\ &= 0111_2 \to 0110_2 \to 0100_2 \to 0000_2 \\ &= 0 \end{aligned}$$

$$\begin{aligned} 21 &= 10101_2 \to 10100_2 \to 10010_2 \to 01110_2 \to 00110_2 \to -10 \to 10 \\ &= 01010_2 \to 01001_2 \to 00111_2 \to 00011_2 \to -5 \to 5 \\ &= 00101_2 \to 00100_2 \to 00010_2 \to -2 \to 2 \\ &= 00010_2 \to 00001_2 \to -1 \to 1 \\ &= 00001_2 \to 00000_2 \\ &= 0 \end{aligned}$$

Dacă analizăm câteva numere putem observa o regulă generală. Pornind de la un număr în binar, prima coordonată după ciocnirea cu bara verticală se obține inversând biții numărului (un bit 0 se transformă în 1 și un bit 1 în 0).

Demonstrația acestui fapt este următoarea:

Numerele scăzute din distanță sunt $1, 2, 4, 8, 16, \ldots$. Dar de fapt se scade un prefix din acest șir, până când numărul devine negativ. Șirul de prefixe este $1, 3, 7, 15, 31, \ldots$. Acestea sunt numere de forma $2^k - 1$ unde k este un număr natural nenul. În binar acestea au forma $1_2, 11_2, 111_2, 1111_2, 11111_2, \ldots$.

Primul moment în care super-mărul lăsat să cadă de la distanța x se lovește de bara verticală este dat de k-ul minim pentru care $2^k - 1 \geq x$. Acesta corespunde exact celui mai semnificativ bit din x. Poziția super-mărului după ciocnirea de bara verticală este $-\left(x - \left(2^k - 1\right)\right) = 2^k - 1 - x$.

Deoarece $2^k - 1 = 1111\ldots1_2$ iar $x \leq 2^k - 1$ formula $2^k - 1 - x$ se reduce doar la inversarea biților lui x.

Obținem următoarea soluție:

Convertim numărul în baza 2. Pentru fiecare bit i care diferă de bitul $i + 1$ vom adăuga ceva la răspuns, fie 1 pentru cerința 1, fie $i + 1$ (sau i dacă se indexează de la 1) pentru cerința 2.

Conversia în baza 2

Algoritmul clasic de conversie al unui număr N din baza 10 în baza 2 folosește împărțiri repetate la 2. Resturile obținute sunt scrise în ordine inversă pentru a obține reprezentarea binară a lui N. Acest algoritm, cel puțin în forma aceasta este prea lent.

Pentru a optimiza algoritmul avem 2 posibilități.

1. Stocăm numărul într-o bază mai mare, ușor de calculat din baza 10, mai exact $10^{B_{10}}$. Pentru aceasta vom stoca B_{10} cifre într-un singur număr.

2. Convertim numărul într-o bază mai mare, din care putem ajunge ușor la baza 2, mai exact 2^{B_2}. După ce am făcut această conversie, fiecare "cifră" din această bază va reprezenta B_2 biți.

Soluția oficială folosește ambele optimizări, cu $B_{10} = 9$ și $B_2 = 30$. Astfel convertim un număr din baza 10^9 în baza 2^{30}. Aceste numere sunt apropiate și se poate observa că algoritmul va avea complexitatea $O\left(C^2\right)$ unde C este numărul de "cifre" din reprezentarea în baza 10^9 a lui N.

Bonus

Aceste două optimizări îmbunătățesc algoritmul de conversie. Există totuși o limită impusă de reprezentarea numerelor pe calculator, depinzând de dimensiunea maximă a unui număr. Mai exact, conversia poate genera o eroare dacă $2^{B_2} \cdot 10^{B_{10}} - 1$ nu poate fi reprezentat fără probleme pe calculator.

Astfel, dacă ne dorim să creștem B_{10}, s-ar putea ca B_2 să trebuiască să scadă pentru a nu avea probleme.

Problema 3. Program

Propusă de: stud. Răzvan Rotaru, Facultatea de Informatică, Universitatea "Alexandru Ioan Cuza" Iași

Subtaskul 1: $1 \leq N \leq 5000$, $P = 1$, $C = 1$

Se determină ziua maximă de începere a studiului astfel încât toate cursurile să se finalizeze la timp, fără a include excursiile. Deoarece cursurile se desfășoară secvențial, timpul minim de finalizare al fiecărui curs se calculează prin însumarea duratelor cursurilor anterioare și a celui curent. Folosind tehnica sumelor parțiale, se obține pentru fiecare curs timpul minim necesar finalizării acestuia. Acest timp se compară cu termenul limită al cursului, iar diferența (termenul limită minus timpul minim de finalizare) indică cu cât poate fi amânată începerea studiului fără a încălca restricțiile de timp. Pentru determinarea minimului, se compară toate perechile de diferențe. Ziua maximă de start se stabilește prin identificarea valorii minime a acestor diferențe, aceasta reprezentând numărul maxim de zile în care Mihăiță poate să nu învețe, la care se adaugă 1.

Complexitate: O(N^2).

Observație: o soluție care încearcă să fixeze inițial, în mod consecutiv, ziua în care Mihăiță începe parcurgerea cursurilor și verifică pentru fiecare curs dacă poate fi finalizat la timp, căutând valoarea maximă a acestei zile de start pentru care condiția este satisfăcută, obține jumătate din punctajul pentru acest subtask. Complexitate: O($Vmax * N$).

Subtaskul 2: Fără restricții suplimentare, $C = 1$, $P = 1$

Pentru a găsi minimul dintre diferențe este suficientă o singură parcurgere, în care reținem minimul de zile necesare, într-o variabilă auxiliară. O altă abordare posibilă pentru $c = 1$ este să încercăm să poziționăm cursurile cât mai aproape de termenul limită, aflând astfel ziua maximă în care Mihăiță se poate apuca de studierea cursurilor. Pentru această abordare trebuie să parcurgem cursurile de la ultimul la primul și să ne asigurăm că perioadele în care fixăm parcurgerea cursurilor nu se intersectează. Complexitate: O(N).

Subtaskul 3: $1 \leq P, Vmax \leq 5000$, $N = 1$, $C = 2$

O excursie va fi asimilată cu un interval închis. Pentru a determina numărul maxim de excursii desfășurate simultan într-o anumită zi, se aplică algoritmul „Șmenul lui Mars”. Metoda utilizează un vector de frecvență în care, pentru fiecare excursie, se marchează extremitatea de început cu +1 și cu -1 poziția imediat următoare extremității de final. După parcurgerea vectorului de frecvență în ordine și calcularea sumei parțiale, se obține numărul de excursii active pentru fiecare zi. Numărul maxim obținut la un moment dat indică numărul maxim de excursii care se suprapun în aceeași zi. Complexitate: O($P * Vmax$).

Subtaskul 4: Fără restricții suplimentare, $C = 2$, $N = 1$

În acest subtask, extremitățile intervalelor excursiilor pot avea valori foarte mari, ceea ce face imposibilă utilizarea unui vector de frecvență. Pentru a rezolva această problemă se

utilizează o tehnică de „liniarizare” a extremităţilor. Se extrag toate extremităţile intervalelor și se etichetează conform tipului lor: extremităţile de început sunt marcate cu +1, iar cele de final cu -1. Aceste extremităţi se sortează în ordine cronologică, iar, în caz de egalitate, se acordă prioritate extremităţilor de început (pentru a reflecta corect începerea unui interval înainte ca altul care are un timp de final identic să se încheie). Se parcurge lista sortată:

- dacă extremitatea curentă este una început al unui interval, creştem numărul de excursii care se desfăşoară la momentul curent;
- dacă extremitatea curentă este una de final al unui interval, comparăm numărul de excursii care se desfăşoară simultan la momentul curent (numărul de intervale deschise care nu au fost închise) cu maximul, apoi scădem cu 1 numărul de excursii care se desfăşoară la momentul curent.

Valoarea maximă obținută indică numărul maxim de excursii care se suprapun într-o anumită zi. Complexitate: O($S * logS$).

Subtaskul 5: $1 \leq N, P \leq 5000$, $C = 3$

Soluția se bazează pe un algoritm de tip interclasare: se parcurg simultan lista capitolelor și lista excursiilor, ambele sortate crescător. Folosim o variabilă care reține prima zi disponibilă pentru studiu. Pentru fiecare excursie se analizează perioada liberă de studiu dinaintea datei de start a excursiei, pentru a verifica dacă aceasta permite finalizarea completă a capitolului curent. Dacă intervalul liber este suficient pentru a termina cursul, se programează acel capitol și se trece la capitolul următor, continuând evaluarea în cadrul aceluiași interval liber. Dacă intervalul nu este suficient de mare, se actualizează data disponibilă pentru studiu la ziua imediat următoare zilei în care se termină excursia, iar încercarea de a finaliza același capitol se reia cu noua perioadă liberă. Propunerea este acceptată doar dacă, pentru fiecare curs, ziua de finaliza nu depășește termenul limită. Complexitate: O($P * N$), deoarece pentru fiecare propunere iterăm prin fiecare capitol.

Subtaskul 6 – Fără restricții suplimentare, $C = 3$

Se extinde metoda de la Subtaskul 5 și se optimizează determinarea numărului maxim de cursuri ce pot fi parcurse într-un interval liber prin utilizarea căutării binare pe vectorul de sume parțiale al duratelor cursurilor (calculat și la Subtaskul 1). Se precalculează, de asemenea, un vector auxiliar care, pentru fiecare curs, reține minimul diferenței dintre termenul limită și numărul de zile necesare studiului cursului curent și al celor ce urmează, facilitând astfel verificarea rapidă a constrângerilor. Pentru fiecare propunere de excursii, intervalele se sortează crescător după extremitatea inițială. Parcurgem intervalele libere dinaintea fiecărei excursii şi, pentru fiecare interval liber, căutăm binar pe vectorul de sume parțiale pentru a identifica rapid numărul maxim de cursuri finalizabile înainte de începerea excursiei curente. Dacă nu se pot programa toate cursurile disponibile în intervalul respectiv, se trece la următoarea excursie, se actualizează perioada de studiu disponibilă și se repetă verificarea pentru cursurile rămase, asigurându-se totodată că fiecare termen limită este respectat. Complexitate: O($SlogN$).

Echipa

Problemele pentru această etapă au fost pregătite de:

- Prof. Emanuela Cerchez, Colegiul Național "Emil Racoviță" Iași
- Stud. Dumitru Ilie, Facultatea de Matematică-Informatică, Universitatea București
- Stud. Andrei Boacă, Facultatea de Informatică, Universitatea "Alexandru Ioan Cuza", Iași
- Prof. Isabela Patricia Coman, Colegiul Național de Informatică "Tudor Vianu" București
- Stud. Victor Botnaru, Facultatea de Automatică și Calculatoare, Universitatea Națională de Știință și Tehnologie POLITEHNICA București
- Stud. Răzvan Alexandru Rotaru, Facultatea de Informatică, Universitatea "Alexandru Ioan Cuza", Iași
- Stud. Giulian Buzatu, Facultatea de Matematică-Informatică, Universitatea București
- Prof. Nistor Moț, Liceul Teoretic "Dr. Luca" Brăila
- Prof. Alin Burța, Colegiul Național "B. P. Hașdeu" Buzău
- Prof. Florentina Ungureanu, Colegiul Național de Informatică Piatra Neamț