Olimpiada Națională de Informatică, Etapa județeană, Clasa a IX-a Descrierea soluțiilor

Comisia științifică

March 15, 2025

Problema 1. SUMMAT

Propusă de: prof. Mihai Bunget

Rezolvarea problemei Summat presupune folosirea unor tehnici de programare precum: generarea unei matrice cu elemente date, sume parţiale pe vector, sume parţiale pe matrice, observaţii referitoare la un şir de numere în care secvenţele de elemente consecutive egale devin din ce în ce mai lungi. Pentru obţinerea a 100 de puncte nu este necesară căutarea binară, timpul de execuţie fiind setat pentru căutare secvenţială. Observaţia esenţială pentru a putea obţine 100 de puncte este că secvenţele de valori egale din matrice sunt foarte lungi şi astfel vom avea multe linii consecutive cu aceleaşi elemente. De asemenea de menţionat şi faptul că valoarea elementelor matricei A nu depăşeşte 55, deci suma elementelor oricărei submatrice se va încadra pe 64 de biţi.

Subtask 1.(24 puncte)

Se generează matricea A formată cu elementele şirului 1, 2, 2, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, ..., apoi pentru fiecare submatrice se calculează suma elementelor, prin parcurgerea submatricei respective. Complexitatea soluţiei este $O(Q \cdot M \cdot N)$.

Subtask 2.(14 puncte)

Deoarece $M = 1$ matricea A are o singura linie, deci este un şir. Se generează şirul A, se calculează şirul S al sumelor parţiale ale lui A şi, pentru fiecare submatrice dată, suma elementelor sale este $S_{c2} - S_{c1-1}$. Complexitatea soluţiei este $O(N + Q)$.

Subtask 3.(19 puncte)

Se generează matricea A, se calculează matricea S a sumelor parţiale şi, pentru fiecare submatrice dată, suma elementelor sale este $S_{l2,c2} - S_{l2,c1-1} - S_{l1-1,c2} + S_{l1-1,c1-1}$. Complexitatea soluţiei este $O(M \cdot N + Q)$.

Subtask 4.(11 puncte)

Deoarece $M = 1$, matricea A are o singură linie. Vom nota cu $Sum(k)$ suma elementelor de pe această linie pană la elementul de indice k. Pentru a calcula $Sum(k)$ se determină i maxim astfel încât $T = 1+2+2^2+2^3+...+2^i <= k$, în acelasi timp calculând suma $S = 1+2\cdot 2+2^2\cdot 3+2^3\cdot 4+...+2^i\cdot i$, şi vom obţine $Sum(k) = S + (k - T) \cdot (i + 1)$. Suma elementelor unei submatrice date va fi $Sum(c2) - Sum(c1 - 1)$. Complexitatea soluţiei este $O(C \cdot Q)$, unde constanta C este aproximativ 50 pentru căutarea lui i secvenţial, respectiv aproximativ 6 pentru căutarea lui i binar.

Subtask 5.(15 puncte)

Pentru fiecare linie i de la $l1$ la $l2$ se calculează $Sum((i - 1) \cdot N + c2) - Sum((i - 1) \cdot N + c1 - 1)$ iar suma acestor numere reprezintă suma elementelor submatricei date. Complexitate soluţiei este $O(Q \cdot 100 \cdot C)$, unde constanta C are aceleaşi valori ca mai sus.

Subtask 6.(17 puncte)

Se determină coordonatele elementelor din matricea A în care se schimbă valoarea elementului din şirul 1, 2, 2, 3, 3, 3, 3, ... dat. Fie $(x[i], y[i])$ coordonatele acestor elemente, unde $1 \leq i \leq 55$ (se poate aproxima cu uşurinţă că numărul maxim din matricea A nu poate depăşi valoarea 55). Se determină k minim cu $l1 \leq x[k]$ şi p maxim cu $x[p] \leq l2$ (acest lucru se poate face secvenţial sau prin căutare binară). Pentru fiecare i de la k la p se calculează $Sum((x[i]-1)\cdot N+c2)-Sum((x[i]\cdot N+c1-1)$ (adică suma elementelor de pe linia $x[i]$ din submatricea dată), numar care se adaugă la suma elementelor submatricei date. De asemenea, pe liniile cuprinse între linia $x[i]$ şi linia $x[i+1]$, exclusiv acestea, se află acelaşi număr în submatricea dată şi anume i, deci suma elementelor de pe aceste linii va fi $(x[i+1]-x[i]-1)\cdot(c2-c1+1)\cdot i$, număr care se adaugă la suma elementelor submatricei date. Complexitatea soluţiei este $O(Q \cdot C)$,unde constanta C are aceleaşi valori ca mai sus.

Problema 2. TELEPORTOR

Propusă de: stud. Bogdan Ioan Popa

Observăm că dacă avem două camere vecine pe linie sau coloană numerotate cu x și $x+1$, atunci nu trebuie să folosim teleportorul pentru a traversa de la camerele numerotate cu x la camerele numerotate cu $x+1$. Fie P numărul de valori x distincte pentru care există cel puțin două camere numerotate cu x și $x+1$ vecine pe linie sau coloană. Numărul minim de teleportări necesare vizitării tuturor camerelor este egal cu $K - P - 1$.

Subtask 1.

Pentru fiecare valoare x de la 1 la $K-1$, parcurgem matricea pentru a depista dacă există cel puțin două celule vecine care sunt numerotate cu x respectiv $x+1$. Complexitatea acestei soluții este $O(N^2 \cdot Q \cdot K)$.

Subtask 2.

Parcurgem matricea și pentru fiecare celulă vom vizita cei 4 vecini ai ei. Dacă găsim pe celula curentă valoarea x, iar într-unul dintre vecini valoarea $x+1$, vom marca acest lucru într-un vector de apariții pe poziția x. Numărul de poziții marcate în vectorul de apariții va fi egal valoarea cu P. Complexitatea acestei soluții este $O(N^2 \cdot Q)$.

Subtask 3.

Observăm ca fiecare valoare apare de la 1 la K exact o dată, mai puțin una singură, care apare de două ori. Fie b valoarea care apare de exact două ori la un moment dat. Orice transformare (i, j, c) poate afecta doar o celulă cu proprietatea că $A_{i,j} = b$ înainte de transformare. Vom optimiza calculul lui P folosind această observație. La început vom parcurge matricea și vom calcula P considerând doar perechi de celule vecine numerotate cu valori x și $x+1$ diferite de b, după care vom considera și contribuțiile celor două celule care au valoarea egală cu b. Menținând această strategie, putem actualiza valoarea lui P în $O(1)$ de la o transformare la alta, astfel se obține o soluție de complexitate $O(N^2 + Q)$.

Subtask 4.

Pe parcursul celor Q transformări vom menține un vector de frecvență F_x = câte perechi de celule vecine există pe care se află valoarea x respectiv $x+1$. Valoarea lui P este dată de numărul de valori nenule din F. Actualizarea unei celule presupune scăderea (dacă este cazul) unor valori F_x, apoi creșterea (dacă este cazul) unor valori F_x. De fiecare dată când un F_x devine 0, vom scădea P cu 1, de fiecare dată când F_x devine nenul vom crește P cu 1. Complexitatea acestei soluții este $O(N^2 + Q)$ și obține 100 de puncte.

Problema 3. NATAŢIE

Propusă de: stud. Alex-Matei Ignat

Observație: pentru soluțiile ce folosesc sortări, complexitatea pentru fiecare subtask în parte depinde de sortarea folosită, însă orice sortare cu complexitate cel mult pătratică $O(N^2)$ obține punctaj maxim pentru restricțiile date. Soluțiile prezentate nu iau în calcul și această complexitate, deși aceasta poate fi mai semnificativă decât complexitatea soluției propriu-zise în care se află timpul minim al cursei.

Subtask 1. (17 puncte)

Dacă toate vitezele sunt egale, putem alege oricare M rațe din cele N. Toate configurațiile în care se respectă ordinea crescătoare a rezistențelor sunt valide și dau același timp pentru cursă. Răspunsul va fi $(2 \cdot d_M)/v$, unde v reprezintă o viteză a oricărei rațe. Complexitate: $O(1)$.

Subtask 2. (16 puncte)

Dacă toate rezistențele sunt egale, cea mai optimă variantă este să luăm cele M rațe cu cele mai mari viteze. Putem sorta rațele crescător după viteză punându-le în ordinea crescătoare a vitezelor pe culoarele de la 1 la M. Apoi, simulăm cursa, răspunsul fiind cel mai mare timp în care o rață se întoarce. Complexitate: $O(M)$.

Subtask 3. (15 puncte)

Din moment ce avem cel mult $3 \cdot 10^3$ rațe și trebuie să folosim de fiecare dată M dintre cele N și $M = N - 1$, înseamnă că doar o rață nu va fi folosită. Deci, putem să eliminăm fiecare rață în parte și să simulăm cursa cu cele rămase. Observația cheie este că pentru o configurație aleasă cu M rațe, cea mai optimă așezare a lor pe cele M culoare este cea în care le sortăm crescător după rezistență, iar în caz de egalitate crescător după viteză. Se sortează inițial toate rațele după acest criteriu, eliminăm fiecare în parte și simulăm cursa cu cele rămase, determinând cel mai bun timp dintre curse. Complexitate: $O(M \cdot N)$.

Subtask 4. (18 puncte)

O primă observație este faptul că răspunsul aparține mulțimii $\{t | t = (2 \cdot d_j)/v_i, 1 \leq i \leq N, 1 \leq j \leq M\}$. Pe scurt, timpul cursei optime este cu siguranță unul dintre timpii unei perechi (rață, baliză). Putem calcula toți timpii posibili, îl luăm pe fiecare în parte, iar apoi verificăm dacă există vreo configurație prin care să alegem M rațe și să obținem un timp cel mult egal cu cel ales. În continuare, trebuie să găsim o strategie prin care putem determina dacă M dintre cele N rațe pot fi alese încât să obținem un timp mai mic sau egal decât un timp fixat. În primă fază, toate rațele se sortează după rezistență, iar în caz de egalitate după viteză. Apoi, parcurgem în ordine toate rațele, încercând la fiecare pas să punem rața curentă pe cel mai mic culoar neocupat. Acest lucru este corect deoarece dacă o rață cu rezistența r este folosită pe un culoar, rațele cu rezistențe mai mici nu mai pot fi folosite, deci este optim să o luăm pe cea cu rezistență

minimă dacă se întoarce în timp util. Legat de viteză (la egalitate de rezistențe), este clar că dacă o rață cu viteza v se întoarce înapoi în timp util, este optim să păstrăm rațele cu viteze mai mari pentru culoarele cu balize mai îndepărtate, deci sortarea și simularea cursei sunt corecte. Dacă găsim M rațe care ne obțin un timp mai mic sau egal decât cel fixat, reținem acel timp, la final răspunsul fiind cel mai mic astfel de timp. Complexitate: $O(M \cdot N^2)$.

Subtask 5. (11 puncte)

Din moment ce distanțele balizelor sunt cel mult $3 \cdot 10^3$ și vitezele sunt numere naturale, înseamnă că timpul maxim al unei curse este $6 \cdot 10^3$. Știind că rezultatul este un timp întreg și folosind ideea anterioară, putem lua toți timpii naturali în ordine crescătoare și să verificăm pentru fiecare în parte dacă există o cursă posibilă. Primul timp găsit este răspunsul căutat. Complexitate: $O(N \cdot d_M)$.

Subtask 6. (13 puncte)

O observație cheie este că dacă pentru un timp x nu reușim (folosind ideile anterioare) să alegem M rațe astfel încât să obținem un timp mai mic sau egal pentru cursa noastră, înseamnă că nu vom reuși nici pentru alți timpi mai mici. Deci, putem face o căutare binară pe rezultat, în care luăm la fiecare pas mijlocul ca fiind timpul maxim pe care îl poate avea cursa noastră. Încercăm să căutăm o configurație de M rațe prin care timpul cursei este mai mic sau egal decât cel fixat. Dacă găsim, atunci vom căuta un timp mai mic sau egal ($dr \leftarrow mij$). Dacă nu, atunci avem nevoie de un timp strict mai mare ($st \leftarrow mij + 1$). La finalul căutării binare, timpul găsit este chiar răspunsul. Complexitate: $O(N \cdot log\ d_M)$.

Subtask 7. (10 puncte)

Dacă rezultatul nu este natural, ideea anterioară este validă în continuare, însă trebuie modificată căutarea binară. Din moment ce trebuie să afișăm rezultatul cu o eroare maximă de 10^{-3}, vom căuta binar răspunsul până când diferența dintre capete este mai mică decât această eroare. Deoarece căutăm binar pe numere reale, se modifică atribuirile pentru capetele din stânga și dreapta. Dacă găsim o configurație, atunci vom căuta un timp mai mic sau egal ($dr \leftarrow mij$). Dacă nu, atunci avem nevoie de un timp strict mai mare, însă căutarea fiind pe numere reale, atribuirea e asemănătoare cu cea din cazul anterior ($st \leftarrow mij$). Pentru un punctaj maxim este necesară și afișarea cu precizie, astfel încât să se afișeze cel puțin 3 cifre zecimale. Complexitate: $O(N \cdot log\ d_M)$.

Echipa

Problemele pentru această etapă au fost pregătite de:

- Prof. Cristina Elena Anton, Colegiul Naţional ”Gheorghe Munteanu Murgoci”, Brăila
- Prof. Cristian Toncea, Colegiul Naţional ”Nicolae Bălcescu”, Brăila
- Prog. Mihai Ciucu, C.S. Academy, Bucureşti
- Prog. Vlad-Alexandru Gavrilă-Ionescu, Google, Zurich
- Stud. Bogdan Ioan Popa, Facultatea de Matematică şi Informatică, Universitatea Bucureşti
- Stud. Alex-Matei Ignat, Universitatea ”Babeş-Bolyai”, Cluj-Napoca
- Prog. Mihaela Cismaru, Google, Paris
- Stud. Andrei Robert Ion, ICHB, Bucureşti
- Stud. Victor-Ştefan Arseniu, Facultatea de Automatică şi Calculatoare, Universitatea Politehnică Bucureşti
- Stud. David-Andrei Lăuran, Universitatea ”Babeş-Bolyai”, Cluj-Napoca
- Stud. Sebastian-Ion Predescu, Facultatea de Automatică şi Calculatoare, Universitatea Politehnică Bucureşti
- Stud. Alexandru Lorintz, Universitatea ”Babeş-Bolyai”, Cluj-Napoca
- Stud. Răzvan Viorel Uzum, Universitatea ”Babeş-Bolyai”, Cluj-Napoca
- Prof. Mihai Bunget, Colegiul Naţional ”Tudor Vladimirescu”, Târgu Jiu