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

Comisia științifică

April 14, 2025

Problema 1. DASHGAME

Propusă de: stud. Uzum Răzvan Viorel

Subtask 1.(10 puncte)

Pentru $a_i = b_i = 0$, răspunsul este *DA* întotdeauna, cu excepţia cazului când $m = 1$ și $n \neq 1$, când jucătorul pierde la a doua coloană. Pentru numărul minim de apăsări, strategia este să apăsăm doar atunci când jucătorul se află pe linia 1 sau linia m. Aplicăm această strategie parcurgând coloanele, ținând minte direcția caracterului și linia pe care se află. Numărul maxim de apăsări este $n - 1$, deoarece jucătorul poate apăsa la fiecare coloană în parte, în afară de ultima.

Subtask 2.(10 puncte)

Putem proceda similar cu subtask-ul anterior, folosind metoda greedy. Pentru numărul minim de apăsări, parcurgem coloanele și schimbăm direcția doar atunci când este abolut necesar, adică atunci când, dacă nu s-ar apăsa ecranul, jucătorul ar pierde pe următoarea coloană. Pentru numărul maxim de pași, acesta schimbă direcția ori de câte ori poate, atât timp cât nu ar pierde pe următoarea coloană dacă schimbă direcția.

Subtask 3.(10 puncte)

Se generează toate traseele posibile cu metoda backtracking și se selectează cel cu număr minim și maxim de apăsări în caz de câștig. În caz că jucătorul nu poate câștiga, se alege poziția cea mai din dreapta pe care jucătorul pierde într-unul dintre traseele generate.

Subtask 4.(10 puncte)

Răspunsul este întotdeauna *DA*. Numărul minim de apăsări este 1, iar numărul maxim de apăsări este $n - 1$.

Subtask 5.(20 puncte)

Putem folosi metoda programării dinamice pentru a calcula, pentru fiecare punct în care jucătorul se poate afla, care este numărul minim și numărul maxim de apăsări pe ecran care au fost necesare pentru ca acesta să ajungă acolo. Pentru numărul minim de apăsări, folosim un tablou tridimensional a, unde $a[i][j][p]$ = numărul minim de apăsări pentru ca jucătorul să ajungă pe linia i, coloana j, având direcția dreapta-sus, dacă $p = 0$ și dreapta-jos dacă $p = 1$. Formulele de recurență sunt:
$a[i][j][1] = min(a[i-1][j-1][0] + 1, a[i-1][j-1][1])$
$a[i][j][0] = min(a[i+1][j-1][0], a[i+1][j-1][1] + 1)$
Similar, pentru numărul maxim de apăsări, formăm tabloul b, unde $b[i][j][p]$ = numărul maxim de apăsări pentru ca jucătorul să ajungă pe linia i, coloana j, având direcția dreapta-sus, dacă $p = 0$ și dreapta-jos dacă $p = 1$. Formulele de recurență sunt:
$b[i][j][1] = max(b[i-1][j-1][0] + 1, b[i-1][j-1][1])$
$b[i][j][0] = max(b[i+1][j-1][0], b[i+1][j-1][1] + 1)$

Subtask 6.(40 puncte)

Pentru un stâlp cu înălțimea x, ne putem imagina că există pe coloana anterioară un stâlp cu înălțimea $x - 1$, cu două coloane mai la stânga un stâlp cu înălțimea $x - 2$, și tot așa. Dacă jucătorul se lovește de un stâlp imaginar, este garantat că acesta se va lovi și de un stâlp real. Așadar, putem actualiza vectorii a și b, parcurgându-i de la dreapta la stânga și aplicând: $a[i] = max(a[i], a[i+1] - 1)$, respectiv $b[i] = max(b[i], b[i+1] - 1)$. Apoi, putem folosi metoda greedy de la subtask-ul 2. În caz că jucătorul nu poate câștiga, putem aplica o căutare binară pentru a găsi coloana cea mai din dreapta pe care jucătorul ar fi câștigat, dacă nu ar mai fi existat coloanele de după. Astfel, răspunsul va fi coloana imediat din dreapta celei găsite.

Problema 2. ECHILIBRARE

Propusă de: Mihai Ciucu

Toate soluțiile se bazează pe câteva observaţii simple. În primul rând, nu are rost să mutăm bile decât din jumatatea cu cele mai multe bile.

Intâi să calculăm câte bile trebuie mutate în total, pentru a ne simplifica logica. Dacă diferenţa maximă e deja $\leq K$, nu trebuie sa mutăm nimic si afişăm 0 0. Dacă în jumătatea cea mai mică avem A bile, şi în cea mai mare avem B bile, şi trebuie să avem diferenţa maxim K, rezolvând o inecuaţie obţinem:

$(DiferentaCurent - K)/2$ unde $DiferentaCurenta = B - A$. Pentru că lucrăm cu întregi, considerăm partea întreagă superioară (adăugăm 1 înainte de diviziunea pe întregi la 2).

Comisia vă recomandă tot timpul să verificaţi formule de acest tip pe câteva cazuri la limită, pare şi impare.

Subtask 1. Soluţie O(N) pentru prima subcerinţă şi O(N) per query (20 de puncte)

Pentru prima cerinţă putem să luăm una câte una urnele cele mai mari, şi să le mutăm tot conţinutul până când am obţinut numărul necesar de bile mutate. Pentru a face asta eficient, sortăm mai întâi aceste urne după numărul de bile.

Pentru a afla deranjul maxim incercăm toate deranjurile maxim incremental cât timp se poate obţine o soluţie cu acelaşi număr de mutări.

Subtask 2. Soluţii O(N + Max) / O(N) per query (între 40 şi 50 de puncte)

Soluţia de mai sus se poate rafina dacă ne dăm seama că oricare ar fi deranjul minim, acesta poate fi obţinut din cele mai mari X urne, unde X este numărul minim de urne din care putem muta bile.

Observaţie: Orice soluţie cu deranj minim corect este echivalentă cu una în care din urne ori se ia tot conţinutul, ori se ia fix deranjul (exceptând cel mult o urnă). Daca nu ar fi aşa, ar însemna că există o soluţie cu mai puţine urne modificate.

Putem să încercăm să mărim deranjul minim cu câte o unitate, menţinând o sumă parţiale cu toate urnele din care luăm complet, şi incercând să vedem în O(1) dacă se pot muta restul de bile necesar din urnele neprocesate.

Această soluţie are complexitate O(Max) şi poate fi redusă la O(N) dacă atunci când mărim cu 1 numărul de urne luate complet verificăm explicit prin formulă dacă avem deranj posibil pentru urnele rămase.

Subtask 3. O(log Max log N) sau O(log N) per query

Pentru că am lucrat pe un şir cu valori crescătoare, folosim căutarea binară pentru a afla valorile dorite.

În primul rând, pentru a afla numărul minim de urne căutăm binar câte din cele mai mari urne trebuie mutate. Verificarea în timp constant dacă x urne sunt suficiente necesită să le

sortăm descrescător întâi şi să calculăm o sumă parţială pe primele i urne. Odată ce am aflat numărul minim de urne, putem să căutăm binar care este deranjul minim.

Verificarea dacă un deranj minim este posibil poate fi facută implementată relativ simplu prin incă o căutare binară, unde verificam câte bile din cele mai mari X conţin mai puţine bile decât deranjul minim, caz în care acestea vor fi luate complet, şi suma lor se poate calcula cu vectorul de sume parţiale. Celelalte trebuie limitate la deranjul căutat, şi doar verificăm dacă au minim valoarea necesară.

La fel ca mai sus, soluţia de O(log Max log N) se poate reduce la una de complexitate O(log N) dacă în loc să căutăm binar deranjul, căutăm binar poziţia în cele mai mari X urne până la care toate urnele se iau complet. Dacă calculăm suma parţială a urnelor care se iau complet, asta implica un deranj minim necesar care se poate calcula din suma rămasă neacoperită.

Problema 3. Antidivizor

Propusă de: stud. Predescu Sebastian-Ion

Subtask 1. (35 puncte)

Datorita restricțiilor mici a lui N și T, soluția care obține punctajul pentru acest subtask este cea prin calcul direct. Pentru fiecare număr N, se calculeză în mod direct care este antidivizorul numerelor sub acesta și se adună la rezultat. Complexitate: $O(N \times T)$.

Subtask 2. (26 puncte)

În cadrul acestui subtask, avem restricția ca $1 \leq N \leq 10^6$, deci putem precalculara toate toate soluțiile pentru orice N mai mic decât 10^6, iar după ă răspundem în $O(1)$ pentru orice întrebare. Complexitate: $O(T + N)$.

Subtask 3. (19 puncte)

O primă observație este faptul că, pentru ca $F(k) = x$, atunci 1 divide k, 2 divide k, ..., $x - 1$ divide k, dar x nu divide k. Deci asta înseamnă că $cmmmc(1, 2, ..., x - 1)$ divide k, unde cmmmc-ul reprezintă cel mai mic multiplu comun al numerelor respective. x nu divide pe k, deci $cmmmc(1, 2, ..., x - 1, x)$ nu divide k. Așadar, numărul de k-uri mai mici decât N, care au $F(k) = x$, este $n/cmmmc(1, 2, ..., x - 1) + n/cmmmc(1, 2, ..., x - 1, x)$. Astfel, pentru fiecare N, calculăm în timp real valorile cmmmc-urilor si aflăm pentru ficare valoarea care este răspunsul. Se poate observa cu usurință faptul că $F(k) \leq 50$. Complexitate $O(50 \times 50 \times T)$.

Subtask 4. (20 puncte)

Punctajul maxim îl obținem prin precalcularea valorilor cmmmc-urilor. Complexitate: $O(T \times 50)$.

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. Răzvan Viorel Uzum, Universitatea ”Babeş-Bolyai”, Cluj-Napoca
- 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
- Prof. Mihai Bunget, Colegiul Naţional ”Tudor Vladimirescu”, Târgu Jiu