Concursul Județean de Informatică „Future for Future”
Ediția a II-a, 2025
Clasa a IX-a
Descrierea soluțiilor

1. Problema Tăbliță

Propusă de: Andrei-Sebastian Drăgulescu, CNMB

Subtask 1 (15 puncte)

Pentru cerința 1, trebuie să calculăm numărul de secvențe valide care se pot forma. Începem prin a număra într-o variabilă X câte cifre au în total numerele citite. Întrucât toate cifrele de pe tăbliță sunt nenule, toate secvențele sunt valide. În total există $X - K + 1$ secvențe.

Subtask 2 (15 puncte)

Observăm că pentru a determina numărul de secvențe valide putem scădea din numărul total de secvențe calculat anterior cel de secvențe care nu sunt valide. Cum toate secvențele care nu sunt valide au cifra 0 pe prima poziție, este suficient să numărăm câte cifre de 0 se găsesc pe inscripție între pozițiile 1 și $X - K + 1$.

Subtask 3 (20 puncte)

Pentru cerința 2, trebuie să construim secvențele de pe inscripție. Observăm că, similar ca la subtaskul 1, toate secvențele sunt valide. Începem prin a parcurge cifrele fiecărui număr citit de la prima la ultima. Putem să stocăm secvența validă curentă într-o variabilă T de tip `long long` și vom adăuga fiecare cifră începând de la poziția K la finalul secvenței de până acum. Cum știm sigur că secvența va fi una validă, vom actualiza maximul și suma cu valoarea T. După actualizare, tot ce ne rămâne de făcut este să eliminăm prima cifră din numărul T.

Subtask 4 (50 puncte)

Pentru acest subtask putem folosi același algoritm ca pentru subtaskul 3, dar trebuie să verificăm pentru fiecare dintre sevențele T validitatea lor. Pentru asta, înainte de a actualiza maximul și suma, vom verifica dacă T conține exact K cifre.

Complexitate timp: $\mathcal{O}(N)$.

Exemplu de implementare: https://kilonova.ro/submissions/504221

2. Problema Curier

Propusă de: Alexandru Thury-Burileanu, CNMB

Subtask 1 (10 puncte)

N și Q sunt foarte mici, deci putem simula, pentru fiecare întrebare, fiecare tură al lui Dorel în parte, scăzând de la fiecare casă cuprinsă în interval. Vom parcurge casele crescător după poziția lor. Vom repeta acest lucru pe intervalul $[i, i + K - 1]$, cât timp $a_i > 0$, pentru $i \leq N - K + 1$. La final, vom verifica dacă șirul conține doar valori de 0. Ordinea turelor nu contează.

Complexitate timp: $\mathcal{O}(T \cdot Q \cdot N^2 \cdot a_{max})$.

Exemplu de implementare: https://kilonova.ro/pastes/srRO7HVwg3iR

Subtask 2 (20 puncte)

Putem optimiza ultima soluție, comprimând turele lui Dorel. De exemplu, pentru șirul 2, 4, 4, 2 și $K = 3$, în loc să facem de două ori intervalul $[1, 3]$ și de două ori intervalul $[2, 4]$, putem să scădem 2 în intervalul $[1, 3]$ și 2 în intervalul $[2, 4]$. Așadar, pentru fiecare $i \leq N - K + 1$, vom scădea a_i din elementele intervalului $[i, i + K - 1]$. La final, vom verifica dacă șirul conține doar valori de 0, ca în soluția precedentă.

Complexitate timp: $\mathcal{O}(T \cdot Q \cdot N^2)$.

Exemplu de implementare: https://kilonova.ro/pastes/3TECMRlYtqP3

Subtask 3 (30 puncte)

În loc să iterăm prin tot intervalul cuprins de o tură comprimată în $\mathcal{O}(K)$, putem să facem același lucru în $\mathcal{O}(1)$. Ne vom folosi de **Șmenul lui Mars (Difference Arrays)** și vom ține un vector auxiliar b. În loc să iterăm prin intervalul $[i, i + K - 1]$ și să scădem din toate elementele acestuia, vom scădea a_i din b_i și vom aduna a_i la b_{i+K} pentru $i \leq N - K + 1$. La fiecare pas, $b_i \leftarrow b_i + b_{i-1}$ pentru a propaga scăderile anterioare și $a_i \leftarrow a_i + b_i$, pentru a lua în calcul scăderile făcute. La final, vom verifica dacă șirul conține doar valori de 0, ca în soluțiile anterioare.

Complexitate timp: $\mathcal{O}(T \cdot Q \cdot N)$.

Exemplu de implementare: https://kilonova.ro/pastes/hW4QTxrUb6ZX

Subtask 4 (20 puncte)

O observație care reduce drastic timpul este că doar K care divid suma numerelor din șir pot fi valide, deoarece turele făcute de Dorel sunt doar scăderi repetate.

Complexitate timp teoretică: $\mathcal{O}(T \cdot Q \cdot N)$, dar, în realitate, se comportă foarte bine, fiind de minim 10 ori mai rapidă ca soluția anterioară.

Exemplu de implementare: https://kilonova.ro/pastes/ecBYEnQz27gg

Subtask 5 (20 puncte) – Soluția finală

O altă observație importantă este $K \leq N$. Așadar, query-urile se vor repeta cu siguranță. Vom precalcula răspunsurile pentru $1 \leq K \leq N$, iar răspunsul la fiecare query poate fi găsit în $\mathcal{O}(1)$.

Complexitate timp teoretică: $\mathcal{O}(T \cdot N^2)$, dar, în realitate, se comportă mult mai bine.

Exemplu de implementare: https://kilonova.ro/pastes/71XT5HjJIgrn

3. Problema Flotă

Propusă de: George-Mihai Tega, CNMB

Subtaskurile 1 și 2 (10 + 20 puncte)

Pentru a număra divizorii unui număr, putem folosi algoritmul de scoatere a divizorilor în $\mathcal{O}(\sqrt{N})$ sau formula bazată pe descompunerea în factori primi: dacă $a = p_1^{e_1} p_2^{e_2} \dots p_k^{e_k}$, numărul divizorilor săi este $\sigma_0(a) = (e_1 + 1)(e_2 + 1) \dots (e_k + 1)$.

Subtaskurile 3 și 4 (10 + 40 puncte)

Calculăm suma divizorilor fie cu ajutorul algoritmului de scoatere a divizorilor, fie prin formula $\sigma_1(a) = (1 + p_1 + p_1^2 + \dots + p_1^{e_1}) \dots (1 + p_k + p_k^2 + \dots + p_k^{e_k})$.

Subtaskul 5 (20 puncte) – Soluția finală

Ce fel de numere au suma divizorilor impară? Pentru a răspunde la această întrebare, observăm în formula lui $\sigma_1(a)$ că toate sumele din paranteze trebuie să fie impare. Dacă p_i este impar, e_i trebuie să fie par, iar dacă p_i este par (adică, fiind prim, $p_i = 2$), nu avem restricții asupra lui e_i.

Așadar, toți exponenții factorilor primi diferiți de 2 sunt pari. Cu alte cuvinte, numărul este pătrat perfect sau dublul unui pătrat perfect.

Pentru a verifica această proprietate, putem folosi funcția `sqrt()` din biblioteca `cmath` sau un vector caracteristic/de frecvență în care marcăm toate numerele valide.

Exemple de implementare: https://kilonova.ro/pastes/PXtKW66hgNtd, https://kilonova.ro/pastes/jAWT72royopz