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

COMISIA ȘTIINȚIFICĂ

Problema 1: Hibrid

Propusă de: stud. Andrei Onuț, Universitatea Yale, S.U.A.

Observație inițială: Pentru că cele P porțiuni taxabile sunt disjuncte două câte două, iar nicio bornă dintre cele N nu se poate afla la capetele sau în interiorul segmentelor ce descriu porțiunile taxabile (mai formal, nu există nicio pereche de indici (i, j): $1 \leq i \leq P$ și $1 \leq j \leq N$ pentru care $st_i \leq x_j \leq dr_i$), înseamnă că dr_i **ar putea fi *ignorat* pentru fiecare i:** $1 \leq i \leq P$. Astfel, dacă ar fi nevoie să verificăm, de exemplu, de câte ori s-a trecut peste o porțiune taxabilă, ar trebui doar verificat de câte ori punctul de coordonată st_i a fost *vizitat* în timp ce se efectua deplasarea între două borne consecutive de pe traseu; a număra de câte ori s-a trecut peste o porțiune taxabilă i ($1 \leq i \leq P$) este echivalent cu a număra numărul de indici j ($1 \leq j < N$) pentru care: $min(x_j, x_{j+1}) < st_i < max(x_j, x_{j+1})$; dr_i, observăm, nu mai influențează această abordare.

Notații:

$$min(a, b) = \begin{cases} a & \text{dacă } a < b; \\ b & \text{altfel.} \end{cases}$$

$$max(a, b) = \begin{cases} a & \text{dacă } a > b; \\ b & \text{altfel.} \end{cases}$$

Subtask 1: Întrucât se cunoaște faptul că pentru efectuarea traseului nu se va trece peste niciuna dintre cele P porțiuni taxabile (unde este impusă folosirea motorului termic), în fișierul de ieșire se va afișa -1 (în cazul în care $C = 1$) sau 0 (în cazul în care $C = 2$).

Subtask 2: Întrucât $0 \leq x_i \leq 70$, pentru fiecare i: $1 \leq i \leq N$, înseamnă că lungimea fiecărui segment de dreaptă între două borne consecutive de pe traseu va fi de cel mult 70 ($|x_{i+1} - x_i| \leq 70$, pentru fiecare i: $1 \leq i < N$).

Să considerăm introducerea următorului tablou unidimensional $fr[0, \ldots, 70]$ ce conține exact 71 de elemente, cu următoarea semnificație: $fr[x]$ ($0 \leq x \leq 70$) = de câte ori s-a trecut, în timpul efectuării traseului, prin coordonata x (de pe axa numerelor). Observăm că $fr[x] = 0$ în cazul în care nu s-a trecut niciodată prin dreptul coordonatei x.

Pentru a calcula într-o complexitate de $O(71 \times N)$ care vor fi valorile elementelor din vectorul $fr[0, \ldots, 70]$, putem utiliza următoarea metodă, de tip *brute-force*:

```
for(int i = 1; i < N; ++i) /// iterăm prin lista coordonatelor de borne
{
    int p1 = min(x[i], x[i + 1]); /// capăt "stânga" pe axa numerelor întregi
    int p2 = max(x[i], x[i + 1]); /// capăt "dreapta" pe axa numerelor întregi

   for(int j = p1; j <= p2; ++j) /// iterăm prin coordonatele de pe axa numerelor
        ++fr[j]; /// incrementăm numărul de "vizite" prin dreptul coordonatei j
}
```

Apoi, pentru cerința $C = 1$ se determină care este indicele minim i ($1 \leq i \leq P$) pentru care valoarea $fr[st_i]$ este maximă, iar pentru cerința $C = 2$ se calculează suma:

$$\sum_{i=1}^{P} c_i \times fr[st_i].$$

Complexitate temporală totală: $O(P + N \times 71)$.

De remarcat! Pentru acest subtask se mai știe, în plus, și că $0 \leq st_i < dr_i \leq 70$, pentru fiecare i: $1 \leq i \leq P$. De asemenea, de vreme ce porțiunile taxabile sunt disjuncte două câte două, înseamnă că numărul P nu poate fi prea mare. Mai exact, acesta este cel mult egal cu 35, și se poate obține, spre exemplu, pentru: $st_i = (i-1) \times 2$ și $dr_i = st_i + 1$, pentru fiecare i: $1 \leq i \leq P$, rezultând în intervalele: $[0,1], [2,3], \ldots, [68,69]$. Astfel, înseamnă că și o soluție cu o complexitate de $O(P \times N) = O(35 \times N)$ va trece toate testele aferente acestui subtask.

Subtask 3: Ca și în cazul precedent, avem în continuare că $|x_{i+1} - x_i| \leq 70$, pentru fiecare i: $1 \leq i < N$, ceea ce înseamnă că am putea *simula* traseul descris, iterând prin fiecare coordonată prin care mașina hibrid se deplasează. De asemenea, valoarea absolută a coordonatelor bornelor este cel mult egală cu 300000; prin urmare, pentru fiecare deplasare între două borne consecutive am putea aplica același algoritm ca și mai sus. Trebuie avut grijă la faptul că indicii ale căror valori dorim să le modificăm într-un tablou unidimensional de frecvență/numărare ar putea deveni și negativi. Așadar, vom aplica indicilor o *translație spre dreapta* cu 300000 poziții: mai exact, poziția -300000 va fi translatată în poziția 0, poziția -299999 va fi translatată în poziția 1, ..., poziția 0 va fi translatată în poziția 300000, ..., poziția 300000 va fi translatată în poziția 600000.

Secvența de cod ar putea deveni:

```
for(int i = 1; i < N; ++i) /// iterăm prin lista coordonatelor de borne
{
    int p1 = min(x[i], x[i + 1]); /// capăt "stânga" pe axa numerelor întregi
    int p2 = max(x[i], x[i + 1]); /// capăt "dreapta" pe axa numerelor întregi

   for(int j = p1; j <= p2; ++j) /// iterăm prin coordonatele de pe axa numerelor
    ++fr[j + 300000]; /// incrementăm numărul de "vizite" prin dreptul coordonatei j
}
```

→ unde: $fr[0, \ldots, 600000]$ este tabloul unidimensional cu ajutorul căruia numărăm de câte ori s-a trecut prin dreptul fiecărei coordonate. Așadar, dacă dorim să știm, de exemplu, de câte ori am trecut prin coordonata x ($-300000 \leq x \leq 300000$), accesăm valoarea $fr[x + 300000]$.

Ca și în cazul precedent, complexitatea temporală totală rămâne: $O(P + N \times 71)$.

Subtask 4: Vom *simula*, în continuare, traseul mașinii pe șosea, însă, pentru a ne încadra în limita de timp dată, nu vom mai parcurge coordonate de pe axa numerelor, ci vom parcurge direct lista de intervale ce descriu cele P porțiuni taxabile.

Mai exact spus, vom parcurge fiecare *segment* $(i, i+1)$ ($1 \leq i < N$) al traseului. Vom itera, apoi, prin lista celor P porțiuni, un indice j ($1 \leq j \leq P$) și dacă se întâmplă să avem: $min(x_i, x_{i+1}) < st_j < max(x_i, x_{i+1})$, în lumina observației inițiale, înseamnă că porțiunea taxabilă cu indicele j a fost traversată în timp ce se făcea deplasarea de la coordonata bornei i la cea a bornei $(i+1)$. Astfel, tot într-un tablou unidimensional de frecvență (ce conține P elemente), se va contoriza de câte ori s-a trecut peste fiecare porțiune taxabilă.

Astfel, am putea avea următoarea abordare:

```
for(int i = 1; i < N; ++i) /// iterăm prin lista coordonatelor de borne
{
    int p1 = min(x[i], x[i + 1]); /// capăt "stânga" pe axa numerelor întregi
    int p2 = max(x[i], x[i + 1]); /// capăt "dreapta" pe axa numerelor întregi

    for(int j = 1; j <= P; ++j) /// iterăm prin lista de porțiuni taxabile
        if(p1 < st[j] && st[j] < p2)
```

```
            ++fr2[j]; /// marcăm trecerea peste porțiunea j
    }
```

În mod similar, ca în soluțiile descrise anterior, pentru cerința $C = 1$ se determină care este indicele minim i ($1 \leq i \leq P$) pentru care valoarea $fr2[i]$ este maximă, iar pentru cerința $C = 2$ se calculează suma:

$$\sum_{i=1}^{P} c_i \times fr2[i].$$

Complexitatea totală este: $O(P \times N)$.

Subtask 5: Remarcăm că cele P porțiuni de șosea sunt incluse complet în porțiunea reprezentată de segmentul $[-300000, 300000]$ de pe șosea. Astfel, o observație este că, dacă mașina hibrid trebuie să parcurgă un segment $[l, r]$ ($l \leq r$) de pe șosea, se poate modifica acest interval astfel încât să reprezinte intersecția: $[l, r] \cap [-300000, 300000]$; în cazul în care intersecția este vidă, se poate *ignora* segmentul $[l, r]$ de pe traseu, din moment ce parcurgerea acestuia nu va influența niciuna dintre cele P porțiuni taxabile.

De această dată, este nevoie ca *marcarea* unui interval $[l, r]$ să fie efectuată în timp constant, $O(1)$. Astfel, poate fi folosită, de exemplu, metoda *Vectorului de Diferențe* (*Difference Array* în Engleză); în România, această tehnică mai este cunoscută și sub numele de *Șmenul lui Mars* și puteți citi mai multe pe infoarena.ro sau pbinfo.ro. Întrucât l sau r pot avea valori negative, se va alege din nou o *translație spre dreapta* cu 300000 de poziții. În momentul actualizării intervalului, valoarea de pe poziția $(l + 300000)$ în tabloul unidimensional folosit pentru *Șmenul lui Mars* va crește cu o unitate, iar cea de pe poziția $(r + 300000 + 1)$ va scădea cu o unitate.

Complexitatea totală a acestei soluții este: $O(P + N + max_d)$ și garantează trecerea cu succes a tuturor testelor, unde max_d reprezintă diferența maximă dintre două coordonate vizitate în timpul efectuării traseului (considerând intersecțiile cu segmentul $[-300000, 300000]$): $max_d \leq 600000$.

Problema 2: Tema

Propusă de: prof. Daniela Lica, Centrul Județean de Excelență Prahova

Pentru cerința 1, soluția în complexitate $O(ValMax \times log(ValMax) + N)$ obține punctaj maxim, adică 50 de puncte. Folosind **Ciurul lui Eratostene**, în complexitate $O(ValMax \times log(ValMax))$, se vor determina toate numerele prime $\leq ValMax$ și, pentru fiecare număr compus, reținem cel mai mic și cel mai mare factor prim al său.

Algoritmul de *ciur* generează toate numerele prime. În momentul găsirii unui număr prim p, marcăm toți multiplii lui de la $2 \times p$ până la cel mult $ValMax$ ca fiind neprimi. Pentru fiecare multiplu (notat cu x), p devine cel mai mare factor prim al lui x (iar dacă x avea deja un factor prim, p îl suprascrie). Acest lucru este natural, căci p este cel mai mare număr prim întâlnit până la acest moment. În plus, dacă la momentul curent x era considerat prim, atunci p devine și cel mai mic factor prim al acestuia.

Determinarea secvenței de lungime maximă, pentru care costul acesteia este mai mic sau egală cu K, se realizează în complexitate $O(N)$, folosind *doi indici*. Pentru fiecare indice curent i, considerat capăt dreapta al secvenței curente, determinăm indicele de început (din stânga) al secvenței pentru care produsul nu depășește K. Dacă produsul este mai mare, se renunță la $A[st]$, incrementând st și actualizând valoarea produsului. Se va reține lungimea maximă a secvenței ce respectă cerința. Odată determinat st pentru poziția i, dorim să calculăm valoarea st' pentru poziția $i + 1$. Încorporăm $A[i + 1]$ în costul curent. Cât timp costul depășește K, este clar că pentru niciun i viitor $A[st]$ nu va mai fi parte din soluție (căci, cu cât vom deplasa i spre dreapta, cu atât mai mult vom depăși costul K). Deci, eliminăm $A[st]$ și încercăm pe rând valorile $st + 1, st + 2, \ldots$ până revenim sub costul K. Prima valoare acceptabilă este capătul stâng st' corespunzător lui $i + 1$.

Pentru o soluție care determină în $O(N^3)$ secvența, se obțin aproximativ 20 de puncte.

Pentru o soluție care determină în $O(N^2)$ secvența, se obțin aproximativ 35 de puncte.

Pentru cerința 2, soluția în complexitate $O(N)$ obține punctaj maxim, adică 50 de puncte.

Considerând șirul obținut după transformarea elementelor compuse, vom determina, pentru fiecare poziție i din șir, cât de mult ne putem *extinde* spre stânga și spre dreapta cu o secvență care conține cel mai mic factor prim al lui $A[i]$ (notat *fpmin[A[i]]*), respectiv cel mai mare factor prim a lui $A[i]$ (notat *fpmax[A[i]]*), astfel:

- $st[0][i]$ - cel mai mic indice din stânga unde regăsim *fpmin[A[i]]* ca factor în toate elementele dintre acel indice și i;
- $st[1][i]$ - cel mai mic indice din stânga unde regăsim *fpmax[A[i]]* ca factor în toate elementele dintre acel indice și i;
- $dr[0][i]$ - cel mai mare indice din dreapta unde regăsim *fpmin[A[i]]* ca factor în toate elementele de la i până la acea poziție;
- $dr[1][i]$ - cel mai mare indice din dreapta unde regăsim *fpmax[A[i]]* ca factor în toate elementele de la i până la acea poziție.

Pentru fiecare element $A[i]$, lungimea maximă a secvenței care îl conține și care are cel mai mare divizor comun diferit de 1 este: $max((dr[0][i] - st[0][i] + 1), (dr[1][i] - st[1][i] + 1))$.

Pentru o soluție care determină în $O(N^3)$ secvența, se obțin aproximativ 20 de puncte.

Pentru o soluție care determină în $O(N^2)$ secvența, se obțin aproximativ 35 de puncte.

O altă abordare în $O(N)$ folosește aceeași tehnică a celor *doi indici* de la cerința 1. Pentru un capăt dreapta i, presupunem că am calculat capătul stâng st care produce cea mai lungă secvență cu c.m.m.d.c. diferit de 1. Cum actualizăm secvența pentru $i + 1$? Putem menține un vector de frecvențe al tuturor factorilor primi din secvența curentă. Când adăugăm sau eliminăm un element în/din secvență, incrementăm sau decrementăm frecvențele celor doi factori (dacă factorii unui element sunt egali, modificăm frecvențele doar cu 1, nu cu 2). Atunci, o secvență cu c.m.m.d.c. diferit de 1 are proprietatea că frecvența cel puțin a unui factor de la capătul secvenței este egală cu lungimea secvenței. Când avansăm de la i la $i + 1$, verificăm dacă secvența $[st, i + 1]$ mai are această proprietate. Cât timp nu o are, eliminăm capătul stâng al secvenței.

Echipa

Problemele pentru această etapă au fost pregătite de:

- Prof. Șerban Marinel, Colegiul Național "Emil Racoviță", Iași
- Prof. Ciurea Stelian, Colegiul Național "Samuel von Brukenthal", Sibiu
- Prof. Gorea-Zamfir Claudiu, Inspectoratul Școlar Județean Iași
- Prof. Coman Isabela, Colegiul Național de Informatică "Tudor Vianu", București
- Inst. Frâncu Cătălin, Clubul Nerdvana, România
- Prof. Manolache Gheorghe, Colegiul Național de Informatică, Piatra Neamț
- Prof. Anton Cristina, Colegiul Național "Gheorghe Munteanu Murgoci", Brăila
- Stud. Popa Bogdan Ioan, Facultatea de Matematică și Informatică, Universitatea București
- Stud. Popa Sebastian, Facultatea de Matematică și Informatică, Universitatea București
- Stud. Onuț Andrei, Universitatea Yale, S.U.A.
- Prof. Lica Daniela, Centrul Județean de Excelență Prahova