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

COMISIA ȘTIINȚIFICĂ

Problema 1: Proeminența

Propusă de: Lector dr. Paul Diac, Facultatea de Informatică, Universitatea „Alexandru Ioan Cuza" Iași

Pentru ambele cerințe trebuie eliminate altitudinile consecutive egale. Acest lucru se poate face prin citirea altitudinilor într-o variabilă temporară x și adăugarea în vector doar a valorilor diferite de ultima adăugată:

```
if (x != a[n-1]) { a[n++] = x; }
```

Pentru cerința $C = 1$:
Se parcurg altitudinile $0 < i < n-1$ și se numără vârfurile (pe poziția i este un vârf dacă $a[i-1] < a[i] > a[i+1]$).

Pentru cerința $C = 2$:

Soluție $\mathcal{O}(N^3)$. O primă soluție pentru a determina proeminența unui vârf de pe poziția i este să aplicăm definiția în mod direct. Parcurgem orice alt vârf j și le procesăm pe cele cu $a[j] > a[i]$. Pentru a determina cea mai adâncă vale dintre i și j, se parcurg pozițiile k dintre $min(i, j)$ și $max(i, j)$ și se reține cea mică valoare $a[k]$. Proeminența vârfului i va fi diferența minimă dintre $a[i]$ și valorile $a[k]$ obținute astfel. Soluția se încadrează în timp pe testele cu $N \leq 1000$.

Soluție $\mathcal{O}(N^2)$. După ce fixăm un vârf i pentru care calculăm proeminența, putem parcurge vârfurile j începând de la pozitia i prima dată în stânga, apoi în dreapta. Prin aceste parcurgeri putem determina vârfuri $a[j] > a[i]$ concomitent cu a determina cea mai adâncă vale dintre ele. Reținem acestă altitudine minimă într-o variabilă pe care o folosim și pentru următoarele poziții j.
Optimizare: Analizând mai atent definiția proeminenței, deducem că pe un profil proeminența unui vârf i va fi dată de **primul** vârf mai mare decât $a[i]$ situat în stânga sa $a[s] > a[i]$ $(s < i)$ sau de **primul** vârf mai mare decât $a[i]$ situat în dreapta sa $a[d] > a[i]$ $(d > i)$. Justificarea constă în faptul că, dacă ar exista alte vârfuri $a[k] > a[i]$ cu $k < s$ sau $k > d$ până la care văile ar fi și mai adânci, ele nu vor fi relevante, deoarece atunci când se alege minimul dintre diferența $a[i]$ și cea mai adâncă vale, ele dau o diferență mai mare, care nu modifică minimul. Soluțiile de complexitate $\mathcal{O}(N^2)$ se încadrează în timp pe un număr semnificativ mai mare de teste, în funcție de optimizările efectuate.

Soluție $\mathcal{O}(N)$. Pentru a proiecta o soluție de complexitate liniară determinăm mai întâi *„proeminența la stânga"*, parcurgând vârfurile de la stânga către dreapta. Procedăm apoi similar pentru a determina *„proeminența la dreapta"*, parcurgând vârfurile de la dreapta către stânga și la final afișăm minimul dintre cele două valori obținute pentru proeminență.
Pentru a calcula *proeminența la stânga*, parcurgem vârfurile în ordinea crescătoare a indicilor i. Pentru orice vârf i doar cel mai apropiat vârf j din stânga lui i $(j < i)$, strict mai înalt decât vârful i este relevant, prin urmare toate vârfurile dintre acestea $j < k < i$ care au altitudini mai mici $a[k] < a[i]$, pot fi ignorate mai departe. Este deci suficient să reținem o **stivă** cu vârfuri montane în ordinea strict descrescătoare a înălțimilor. Pentru fiecare dintre aceste vârfuri, vom reține și cea mai adâncă vale situată în dreapta vârfului, până la următorul vârf adăugat în stivă, vale pe care o determinăm „din mers".
Rezumând, parcurgem profilul de la stânga la dreapta şi identificăm vârfurile montane (fie $a[i]$ altitudinea vârfului montan curent).

Cât timp în stivă există elemente şi $a[i] \geq$ altitudinea vârfului montan plasat la vârful stivei, extragem din stivă un element (evident, doar elementul situat la vârful stivei poate fi extras). Atunci când extragem un element din stivă, actualizăm, dacă este cazul, şi înălţimea celei mai adânci văi până la vârful montan curent.
Dacă stiva a devenit vidă, inserăm vârful montan curent în stivă şi iniţializăm înălţimea celei mai adânci văi până la momentul curent cu $a[i]$.
Dacă stiva este nevidă, atunci $a[i] <$ altitudinea vârfului plasat la vârful stivei. În acest caz:

(1) pentru vârful montan situat în vârful stivei memorăm cea mai mică altitudine a unei văi identificate până la vârful montan curent;
(2) determinăm proeminenţa la stânga a vârfului montan curent ca fiind diferenţa dintre $a[i]$ şi altitudinea minimă a unei văi până la vârful montan curent;
(3) reiniţializăm înălţimea minimă a văii curente cu $a[i]$;
(4) inserăm vârful montan curent în stivă.

Pentru a evita duplicarea codului pentru *„proeminența la dreapta”*, putem oglindi vectorul și repeta aceeași procedură.

```
#include <stdio.h>
/*
autor    Paul Diac
idee     Iteram toate inaltimile, retinem varfurile in ordine scrict
         descrescatoare si cele mai adanci vai de dupa ele. Un varf nou adaugat
         elimina toate varfurile mai mici decat el, deci vom folosi o stiva
         pentru varfuri. Pentru a evita alte cazuri, la inceput
         calculam proeminenta la stanga, si apoi la dreapta dupa o oglindire.
         La final afisam minimul dintre cele doua proeminente calculate.
memorie  O(N)
timp     O(N)
scor     100
*/
#include <stdio.h>
#define NMax 1000005
long int N, C, s;
long int a[NMax];

long int min_(long int i, long int j) { if (i < j) return i; return j; }
long int peak[NMax], valley[NMax], top = -1; // peak este stiva de varfuri
                                             // valley retine cele mai adanci vai"
long int prom[2][NMax]; // proeminenta la stanga varfului de pe pozita i este prom[0][i]
                        // si la dreapta este prom[0][n-1-i] datorita oglindirii

void popTop() { // eliminam un varf din stiva, actualizam cea mai adanca vale
                // din dreapta daca e cazul
  if (top < 0) { return; }
  if (top >= 1 && valley[top-1] > valley[top]) {
    valley[top-1] = valley[top];
  }
  top--;
}

int main() {
  freopen("proeminenta.in", "r", stdin);
  freopen("proeminenta.out", "w", stdout);

  scanf("%ld %ld", &C, &N);
  for (int i = 0; i < N; i++) {
    scanf("%ld", &a[i]);
```

```
    if (i > 0 && a[i] == a[i-1]) {
      i--; N--; // eliminam duplicatele
    }
  }
  if (C == 1) {
    for (int i = 1; i < N-1; i++) {
      if (a[i-1] < a[i] && a[i] > a[i+1]) { s++; } // este varf
    }
    printf("%ld\n", s);
    return 0;
  }

  for (int d = 0; d <= 1; d++) { // d=0 va fi parcurgerea stanga-dreapta, si d=1
                                 // dupa oglindire, adica dreapta-stanga
    for (int i = 1; i < N-1; i++) {
      if (a[i-1] < a[i] && a[i] > a[i+1]) { // este varf
        prom[d][i] = a[i];
        while (top >= 0 && a[peak[top]] <= a[i]) { // elimim varfurile mai
                                                   // mici din stiva pe rand
          popTop();
        }
        if (top >= 0 && a[peak[top]] > a[i]) {
          prom[d][i] = a[i] - valley[top];
          // am gasit primul varf mai mare din stanga
        }
        peak[++top] = i;
        valley[top] = a[i]; // adaugam noul varf
      }
      if (top >= 0 && valley[top] > a[i]) { valley[top] = a[i]; }
    }
    for (int i = 0; i < N / 2; i++) { // oglindim altitudinile
      long aux = a[i]; a[i] = a[N-1-i]; a[N-1-i] = aux;
    }
    top = -1;
  }
  for (int i = 1; i < N-1; i++) if (a[i-1] < a[i] && a[i] > a[i+1]) {
    if (prom[0][i] > prom[1][N-1-i]) { // retinem maximul in prom[0][i]
      prom[0][i] = prom[1][N-1-i];
    }
  }

  for (int i = 1; i < N-1; i++) {
    if (prom[0][i] > 0) { printf("%ld ", prom[0][i]); } // si il afisam
  }

  printf("\n");
  fclose(stdout);
  return 0;
}
```

Problema 2: RGB

Propusă de: stud. Ioan Cristian Pop, Universitatea Politehnica București

O observație cheie care simplifică implementarea problemei este aceea că este imposibil ca doi extratereștri să aibă puteri egale în cadrul unei lupte, toți având puteri impare distincte. În cadrul unei lupte în care extratereștrii au culori diferite, puterea unuia dintre extratereștri va deveni pară (iar comparația va fi par cu impar, deci tot între valori distincte). În plus, puterile extratereștrilor de aceeași culoare sunt ordonate crescător în fișierul de intrare.

O altă observație cheie este că problema se rezolvă similar pentru fiecare culoare, deci rezolvarea pentru o anumită culoare se poate aplica și pentru celelalte două culori.

Cerința 1 — Soluție $\mathcal{O}(N)$. Pentru $C = 1$, este suficient să calculăm câte lupte va câștiga cel mai puternic extraterestru de fiecare culoare, apoi să afișăm puterea extraterestrului care câștigă un număr maxim de lupte. Pentru a calcula numărul de lupte câștigate de un extraterestru, este necesar să calculăm doar câte lupte va câștiga luptând cu extratereștrii de celelalte două culori (el fiind cel mai puternic de culoarea sa va câștiga toate luptele cu extratereștrii având aceeași culoare cu el). O atenție specială trebuie acordată cazului în care există mai mulți extratereștri care câștigă același număr maxim de lupte (se va afișa puterea cea mai mică).

Cerința 2 — Soluție $\mathcal{O}(N^2)$. Pentru $C = 2$, simulăm toate luptele dintre extratereștri și reținem câte lupte câștigă fiecare.

Cerința 2 — Soluție $\mathcal{O}(N * log_2(N))$. Fie X și Y doi extratereștri de culori diferite. Observăm că, dacă X va câștiga lupta contra lui Y, atunci X va câștiga luptele contra tuturor extratereștrilor de aceeași culoare ca Y, însă cu putere mai mică. Prin urmare, pentru fiecare extraterestru, putem căuta binar care este cel mai puternic extraterestru de fiecare culoare diferită de a sa pe care îl va învinge. O astfel de soluție poate obține mai multe puncte în funcție de cât de eficientă este implementarea.

Cerința 2 — Soluție $\mathcal{O}(N)$. Putem optimiza soluția anterioară. Fie Z următorul extraterestru de aceeași culoare ca X (cu putere mai mare). Atunci, Z va câștiga din start toate luptele pe care X le va câștiga. Deci, ce trebuie verificat este dacă mai câștigă și alte lupte în plus. Să presupunem că A este cel mai puternic extraterestru de o culoare diferită pe care X îl învinge. Atunci, Z îi va învinge pe toți până la A inclusiv și vom verifica dacă va mai învinge și pe alții de aceeași culoare, căutând secvențial, începând după A. Să presupunem că cel mai puternic extraterestru pe care îl va învinge va fi B. Atunci, pentru următorul extraterestru de aceeași culoare ca X și Z, căutarea va începe după B, și așa mai departe. Analog pentru cealaltă culoare diferită. Tehnica de rezolvare mai este cunoscută drept "Two pointers".

De precizat este că citirea poate fi parsată (datele citite ca șir de caractere și apoi transformate în numere). Nu este însă necesară pentru a obține 100 de puncte.

Problema 3: Subșir

Propusă de: prof. Ionel-Vasile Piţ-Rada, Colegiul Naţional ,,Traian", Drobeta-Turnu Severin

Cerința 1 Soluție ,,naivă" — $\mathcal{O}(N * Nr)$. Pentru fiecare pereche (x, j), se determină șirul cifrelor lui x și apoi, începând cu cifra cea mai semnificativă (cea din stânga) și continuând spre cifra cea mai puțin semnificativă (ultima, cea din dreapta), se caută succesiv prima apariție a cifrei respective și când se găsește această poziție se continuă căutarea de acolo pentru cifra următoare. Dacă se depășește poziția j sau una dintre cifre nu se mai găsește, atunci se oprește căutarea.

Cerința 2 Soluție ,,,naivă" — $\mathcal{O}(N * alfa)$. Am notat cu $alfa$ suma lungimilor intervalelor de interogare, care este cel mult egală cu $Nr * 10000000$. Pentru fiecare interval $[a, b]$ și pentru fiecare număr x din interval se aplică ideea de la cerința 1, cu limita de căutare $j = N$.

Optimizări. Pentru obținerea unei performanțe mai bune este util să facem următoarele precalculări:

(1) Pentru fiecare poziție $1 \leq i \leq N$ și pentru fiecare cifră zecimală $0 \leq cif \leq 9$, determinăm $poz[i][cif]$= cea mai mică (prima) poziție din șirul S pe care apare cifra cif, după poziția i sau $N + 1$ în caz că cifra cif nu mai apare în S după poziția i.
Pentru a verifica dacă x apare ca subșir în prefixul de lungime j al lui S, parcurgem cifrele lui x de la stânga la dreapta și construim subșirul preluând pozițiile din poz (dacă cifra curentă din x este cif, iar poziția pe care apare precedenta cifra din x este i, atunci poziția în subșir a cifrei cif va fi $poz[i][cif]$. Astfel, numărul de pași efectuați pentru verificarea unei valori x este cel mult egal cu numărul de cifre din x.
Asemănător, se poate rezolva cerința 2, verificând succesiv fiecare valoare din fiecare interval $[a, b]$.

(2) O a doua precalculare foarte utilă este aceea prin care pentru fiecare număr x din intervalul $[0, 10^7]$ se determină $minpoz[x]$ = cea mai mică poziție din S cu proprietatea că numărul x apare ca subșir în prefixul de lungime $minpoz[x]$ al lui S, respectiv $minpoz[x] = N + 1$ în cazul în care x nu apare ca subșir în S.
Inițializarea se face prin $minpoz[x] = poz[1][x]$, pentru $0 \leq x \leq 9$, apoi, în ordine crescătoare, pentru fiecare număr x cu $10 \leq x \leq 10^7$, vom calcula

```
minpoz[x]=poz[minpoz[x/10]][x%10];
```

Valorile pentru $minpoz$ se determină astfel cu complexitate liniară.

(3) O a treia precalculare se poate realiza utilizând tehnica sumelor parțiale. Vom determina $nr[x]$ = numărul de numere din intervalul $[0, x]$ care apar ca subșir în S. Utilizând vectorul nr, putem determina numărul de valori x din intervalul $[a, b]$ care apar ca subșir în S ca fiind $nr[b] - nr[a - 1]$ (dacă $a > 0$) sau $nr[b]$ (dacă $a = 0$).

După aceste precalculări, la cerința 1, pentru fiecare pereche (x, j) verificarea are complexitatea $\mathcal{O}(1)$, deci complexitatea totală (pentru cele Nr perechi) este $\mathcal{O}(Nr)$. Analog, la cerința 2 pentru fiecare interval $[a, b]$ numărarea se va realiza tot în $\mathcal{O}(1)$, deci complexitatea totală (pentru cele Nr intervale) va fi $\mathcal{O}(Nr)$.

Problema admite multiple alte abordări, care obţin diferite punctaje.

Echipa

Problemele pentru această etapă au fost pregătite de:

- prof. Emanuela Cerchez, Colegiul Național ,,Emil Racoviță" Iaşi
- prof. Nistor Moţ, Școala Gimnazială ,,Dr. Luca" Brăila
- prof. Ionel-Vasile Piţ-Rada, Colegiul Naţional ,,Traian", Drobeta-Turnu Severin
- lector dr. Paul Diac, Fac. de Informatică, Universitatea ,,Alexandru Ioan Cuza" Iaşi
- prof. Filonela Bălaşa, Colegiul Național ,,Grigore Moisil" Bucureşti
- prof. Lucia Miron, Colegiul Național ,,Costache Negruzzi" Iaşi
- stud. Ioan Cristian Pop, Universitatea Politehnica București
- stud. Stelian Chichirim, Universitatea București

- prof. Daniel Popa, Liceul Teoretic ,,Aurel Vlaicu" Orăştie
- stud. Radu Muntean, ETH Zurich