MINISTERUL EDUCAȚIEI ȘI CERCETĂRII
AGENȚIA NAȚIONALĂ PENTRU CURRICULUM ȘI EVALUARE

PROBLEME
OLIMPIADA REPUBLICANĂ LA INFORMATICĂ

—

2023

Cuprins

CLASELE 7-9

Ziua 1

Produsul maxim

Problema "Produsul maxim" cere elaborarea unui program care să calculeze produsul maxim posibil dintre două numere dintr-o listă dată. Programul va primi o listă de numere întregi și va trebui să returneze un singur număr, care este produsul maxim posibil dintre oricare două numere din lista respectivă.

Turnul alb

Problema "Turn alb" cere dezvoltarea unui program care să determine numărul minim de mutări necesare pentru ca un turn alb să captureze un pion negru pe o tablă de șah de 8x8, având în vedere pozițiile inițiale ale tuturor pieselor și respectând regulile de mișcare ale turnului. Programul va trebui să returneze un număr întreg reprezentând numărul minim de mutări sau mesajul "no solution" dacă capturarea nu este posibilă.

Indice bursier

Problema "Indice bursier" solicită dezvoltarea unui program care să ajute la determinarea prețului celui mai ieftin indice bursier, un pachet compus din acțiuni ale *K* companii listate pe bursă. Programul va primi o listă de prețuri ale acțiunilor pentru *N* companii și va trebui să returneze un număr întreg, reprezentând costul minim al unui astfel de indice.

Produsul maxim

În timpul lecției de matematică de clasa a cincea, elevii învață să înmulțească numere în coloană. Profesoara a decis să le dea elevilor o sarcină suplimentară: a scris pe tablă o listă de numere și le-a cerut să numească două numere ale căror produs este maxim.

Sarcină: elaborați un program care calculează produsul maxim posibil dintre două numere din lista prezentată.

Date de intrare: Prima linie a intrării standard conține un număr natural N - numărul de elemente din tablou.

A doua linie conține N numere întregi a[1], ... , a[N], separate prin spațiu.

Date de ieşire: Ieșirea standard va conține doar o singură linie cu un număr - produsul maxim posibil dintre două numere din listă.

Restricţii: $1 \leq N \leq 10^6$; $|a[i]| \leq 10^9, \forall i \in [1, \ ..., \ N]$.

Limitele de timp și memorie pentru rezolvarea problemei sunt specificate în descrierea generală a problemelor. Setul de date inițial acceptă doar o soluție pentru problemă. Fișierul sursă trebuie să fie numit `maxprod.pas`, `maxprod.c` sau `maxprod.cpp`.

Exemplu:

Intrare	*Ieşire*
5 -2 4 -3 2 3	12

Explicații:

Produsul maxim posibil din lista data se formează cu numerele 4 și 3.

Turn alb

Pe tabla de șah este dată o configurare de piese, compusă dintr-un turn alb, un pion negru și o un număr de pioni albi. Mutarea turnului se definește ca deplasarea piesei pe orizontală sau verticală de-a lungul oricărui număr de celule, dar fără să sară peste alte piese. Dimensiunea tablei de șah este de 8x8 câmpuri.

Sarcină: găsiți numărul minim necesar de mutări ale turnului alb pentru a captura pionul negru.

Date de intrare: În prima linie a intrării standard sunt date în notația șahului (de exemplu, **e5**) coordonatele pionului negru și ale turnului alb.

În a doua linie este dat un număr N ($0 \leq N \leq 8$) - numărul de pioni albi de pe tabla de șah.

În continuare, în N linii, sunt date coordonatele pionilor albi.

Date de ieşire: Ieșirea standard va conține doar o singură linie cu un număr întreg – numărul minim necesar de mutări ale turnului alb pentru a captura pionul negru. În caz, dacă soluție nu există, afișați **`no solution`**.

Restricţii: $0 \leq N \leq 8$

Limitele de timp și memorie pentru rezolvarea problemei sunt specificate în descrierea generală a problemelor. Setul de date inițial acceptă doar o soluție pentru problemă. Fișierul sursă trebuie să fie numit `turnalb.pas`, `turnalb.c` sau `turnalb.cpp`.

Exemplu

1)

Intrare	*Ieşire*

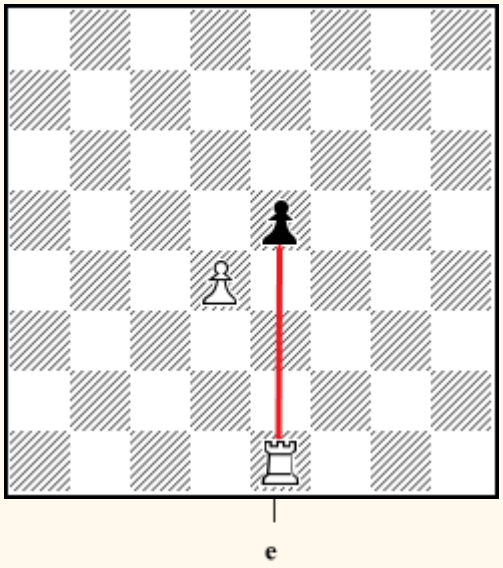
`e5 e1`
`1`
`d4`
	`1`

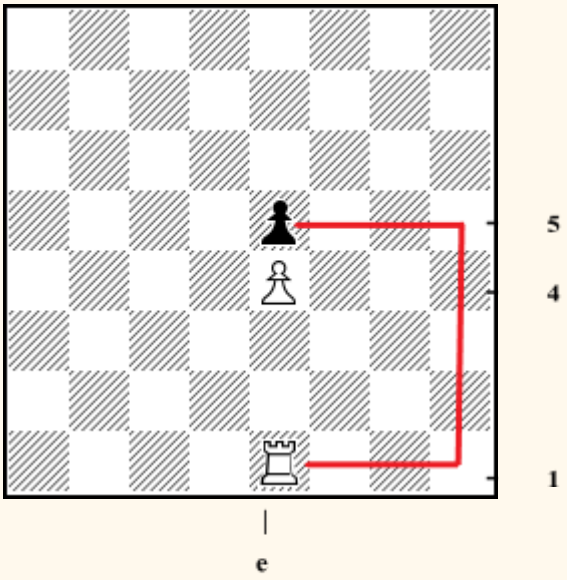
2)

Intrare	*Ieşire*

`e5 e1`
`1`
`e4`
	`3`

Explicații





Indice bursier

Ion Mecenatu a decis să investească în acțiuni. Pe bursa de valori sunt listate acțiuni pentru ***N*** companii, prețul fiecărei acțiuni este un număr întreg. Pe bursă, în afară de acțiunile unei companii, se pot cumpăra și *contracte futures*. Un *contract futures* poate fi considerat indice bursier de piață ce reprezintă un pachet de acțiuni ale ***K*** companii. Deoarece Ion Mecenatu este economicos, el a decis să cumpere cel mai ieftin indice bursier.

Sarcină: Ajutați-l pe Ion Mecenatu să determine prețul celui mai ieftin indice bursier.

Date de intrare: Prima linie a intrării standard conține un număr natural ***N*** - numărul de companii ale căror acțiuni sunt vândute pe bursa de valori.

A doua linie conține ***N*** numere întregi $a[1]$, ..., $a[N]$ separate prin spațiu, care reprezintă prețul acțiunii fiecărei companii.

A treia linie conține un număr întreg ***K*** – numărul de acțiuni în indicele bursier.

Date de ieşire: Ieșirea standard va conține doar o linie cu un număr întreg, reprezentând prețul celui mai ieftin indice bursier.

Restricţii: $1 \leq K \leq N \leq 10^6$; $|a[i]| \leq 10^6$, $\forall i \in [1, ..., N]$.

Limitele de timp și memorie pentru rezolvarea problemei sunt specificate în descrierea generală a problemelor. Setul de date inițial acceptă doar o soluție pentru problemă. Fișierul sursă trebuie să fie numit `indburs.pas`, `indburs.c` sau `indburs.cpp`.

Exemplu:

Intrare	*Ieşire*
5 -2 4 -3 2 3 3	-3

Explicații:

Pentru exemplul specificat din lista prețurilor acțiunilor, cel mai ieftin indice bursier este definit prin acțiunile cu următoarele prețuri: $\{-2, -3, 2\}$. În acest caz, prețul indicelui bursier este $(-2) + (-3) + 2 = -3$.

CLASELE 7-9

Ziua 2

Număr maxim

Problema "Număr maxim" cere dezvoltarea unui program care, primind un număr natural n, identifică și elimină o cifră din acesta astfel încât numărul rămas să fie maxim. Programul va returna numărul maxim obținut, precum și cifra eliminată și ordinul acesteia în înscrierea numărului original.

Bancherul

Problema "Bancherul" cere elaborarea unui program care să ordoneze o listă de numere naturale fragmentate într-o manieră specifică, astfel încât să se obțină cel mai mare număr posibil prin reunirea acestora. Programul va returna acest număr maxim, care poate fi foarte mare.

Lăcata

Problema "Lăcata" cere dezvoltarea unui program care să determine numărul minim de mișcări necesare pentru a deschide o lăcată cu cifru, având în vedere un cod secret și o listă de coduri care ar bloca lăcata. Programul va returna numărul minim de pași pentru a deschide lăcata sau -1 dacă acest lucru nu este posibil.

Număr maxim

Elevii clasei a IV-a învață la matematică tema „Scrierea și citirea numerelor naturale”. Pentru a-i captiva pe elevi și a-i face să înțeleagă mai ușor noțiunile de ordin și clasă, profesoara a inventat un joc prin care: pe tabla interactivă se afișează un număr natural `n`; elevii trebuie să elimine o cifră din înscrierea numărului afișat pe tablă astfel încât numărul rămas să fie maxim; elevii comunică profesoarei cifra care a fost eliminată și ordinul acestei cifre în înscrierea numărului `n`.

Sarcină: Elaborați un program care ar ajuta-o pe profesoară să-și elaboreze resursa digitală necesară pentru lecție.

Date de intrare: Prima linie a intrării standard conține un număr natural `n`.

Date de ieșire: Ieșirea standard va conține pe prima linie un număr natural `n` – numărul maxim obținut după eliminarea unei cifre, iar pe linia a doua se vor afișa două numere naturale `cifra` și `ordin` care specifică cifra eliminată și ordinul acestei cifre în înscrierea numărului `n`, separate printr-un spațiu.

Restricții: $1 \le n \le 9 \cdot 10^{999}$. Restricțiile referitoare la timpul de execuție și volumul utilizat de memorie sunt date în descrierea generală a problemelor propuse pentru rezolvare. Fișierul sursă va avea denumirea `numar.pas`, `numar.c` sau `numar.cpp`.

Notă: Pentru numărul natural $n = \overline{a_n a_{n-1} \dots a_2 a_1 a_0}$ ordinul cifrei a_i, $i = \overline{0,n}$, este egal cu i.

Exemple:

1) Intrare

```
7590163298
```

Ieșire

```
790163298
5 8
```

2) Intrare

```
9750234
```

Ieșire

```
975234
0 3
```

3) Intrare

```
3333334
```

Ieșire

```
333334
3 1
```

Explicații:

Numărul 3333334 care este compus din 6 cifre de 3 și o cifră de 4. Astfel, pentru a obține numărul maximal se poate elimina o cifră de 3. În acest caz, se va elimina cifra care are cel mai mic ordin, 1.

Bancherul

Un bancher ține în siguranță o sumă importantă de bani. Pentru a nu o uita, el a notat-o pe o foaie de hârtie. Din nefericire, într-o zi a găsit această foaie ruptă în mai multe bucăți. Toate cifrele de pe aceste foi sunt ușor de identificat.

De exemplu, suma poate fi împărțită în următoarele părți: 5, 30, 34, 9, 3. Se știe că reunirea acestor numere trebuie să rezulte în cel mai mare număr posibil, cum ar fi 9534330 (9, 5, 34, 3, 30).

Sarcină: Elaborați un program care să ordoneze o listă de numere naturale astfel încât să se obțină cel mai mare număr posibil. Numărul rezultat poate fi foarte mare. Părțile de foi nu mai pot fi împărțite și fiecare parte poate fi utilizată o singură dată în construirea numărului final.

Date de intrare: În primul rând se citește numărul natural **N,** care reprezintă câte numere sunt în listă. În următorul rând urmează **N** numere naturale X_i separate prin spațiu.

Date de ieșire: Un număr natural extrem de mare care reprezintă cel mai mare număr posibil de alcătuit din lista de numere propusă.

Restricții: $1 \le N \le 100$; $0 <= X_i <= 2*10^9$. Restricțiile referitoare la timpul de execuție și volumul utilizat de memorie sunt date în descrierea generală a problemelor propuse pentru rezolvare. Fișierul sursă va avea denumirea `bancherul.pas`, `bancherul.c` sau `bancherul.cpp`.

Exemple:

1)

Intrare	*Ieșire*
5 5 30 34 9 3	9534330

2)

Intrare	*Ieșire*
5 30 0 0 9 88	9883000

Explicații: În ambele exemple avem cinci numere, care fiind reordonate corespunzător condiției de valoare maximă vor reprezenta suma bancherului.

Lăcata

Un tânăr moștenitor primește o ladă cu lucruri prețioase de la tatăl său. Lada este închisă cu o lăcată cu cifru, având 4 rotițe, fiecare cu cifrele de la 0 la 9. Rotițele pot fi rotite în ambele direcții, iar fiecare rotire care presupune o singură schimbare de cifră și la o singură poziție se consideră o singură mișcare. Codul inițial al lăcatei este 0000. Moștenitorul cunoaște codul secret pentru a deschide lada, dar există o capcană: lăcata are anumite coduri care o vor bloca definitiv.

De exemplu, dacă codul secret este 0002 și codurile care vor bloca lăcata sunt 0001 și 0010, atunci secvența 0000 → 0001 → 0002 nu este posibilă, deoarece lăcata s-ar bloca la codul 0001. O altă secvență precum 0000 → 0009 → 0008 → 0007 → 0006 → 0005 → 0004 → 0003 → 0002 necesită 8 pași, dar nu este cea mai eficientă. Una dintre cele mai bune soluții ar fi 0000 → 0090 → 0091 → 0092 → 0002, care necesită doar 4 pași.

Dacă nu este posibil să deschidem lăcata, numărul de pași va fi considerat -1.

Sarcina: Elaborați un program care să determine numărul minim de mișcări necesare pentru a deschide lada, evitând codurile care ar bloca lăcata.

Date de intrare: Primul rând conține parola secretă, formată din 4 cifre în intervalul 0…9. Al doilea rând conține numărul de coduri **N** care blochează lăcata. Următoarele **N** rânduri conțin valorile acestor coduri blocante. Fiecare cod este reprezentat pe un rând separat și este alcătuit întotdeauna din 4 cifre.

Date de ieșire: Numărul minim de pași necesar pentru a deschide lăcata, sau -1 dacă lăcata nu poate fi deschisă.

Restricții: $1 \leq \mathbf{N} \leq 100$. Restricțiile referitoare la timpul de execuție și volumul utilizat de memorie sunt date în descrierea generală a problemelor propuse pentru rezolvare. Fișierul sursă va avea denumirea `lacata.pas`, `lacata.c` sau `lacata.cpp`.

Exemplu:

1) *Intrare*

```
0202
5
0201
0101
0102
1212
2002
```

Ieșire

```
6
```

2) *Intrare*

```
8888
8
8887
8889
8878
8898
8788
8988
7888
9888
```

Ieșire

```
-1
```

Explicații

Pentru primul exemplu codul secret pentru a deschide lăcata este 0202. Avem 5 coduri care blochează lăcata, și anume: 0201, 0101, 0102, 1212 și 2002. Pornind de la cifrul inițial 0000, încercăm să găsim o cale de a ajunge la 0202 fără a atinge niciunul dintre codurile blocante:

1. Se generează combinațiile posibile prin rotirea unei singure rotițe, în ambele direcții: 0000 → 1000, 9000, 0100, 0900, 0010, 0090, 0001, 0009. Eventual se pot exclude pentru ulterioare iterații codurile blocante;
2. Explorăm codul 1000. De aici, avem: 2000, 0000, 1100, 1900, 1010, 1090, 1001, 1009. Niciuna dintre acestea nu se află în lista de coduri blocante, deci putem continua pe fiecare din ele.
3. Din codurile anterioare, putem începe să ne îndreptăm spre codul secret 0202 prin următoarea secvență de mișcări: 0000 → 1000 → 1100 → 1200 → 1201 → 1202 → 0202. Această secvență de mișcări evită toate codurile blocante și necesită **6** pași pentru a ajunge la codul secret.
4. Sigur pot exista și alte secvențe, precum este 0000 → 1000 → 2000 → 2010 → 2020 → 1020 → 0020 → 0202, dar aceasta nu este una minimă, deci nu determină răspunsul corect.

(*) Pentru exemplul doi, răspunsul este evident **-1**, din moment ce se va observa că pentru a ajunge la codul 8888, trebuie să trecem obligator prin una din stările blocante.

CLASA 10

Ziua 1

Camioane

Problema "Camioane" solicită dezvoltarea unui program care să determine numărul minim de comenzi necesare pentru a ajusta temperatura într-o serie de camioane la nivelurile optime specificate, având în vedere că managerul poate ajusta temperatura cu o unitate într-o serie consecutivă de camioane. Programul va returna numărul minim de comenzi necesare pentru a atinge aceste temperaturi optime.

Diamant

Problema "Diamant" cere dezvoltarea unui program care să analizeze o schiță a podelei unui magazin de bijuterii și să determine numărul de "diamante" perfecte care pot fi create folosind gresie albă și neagră. Un diamant este considerat perfect dacă marginile sale sunt formate doar din gresie neagră și interiorul este complet umplut cu gresie albă. Programul va returna numărul total de astfel de diamante perfecte.

Pietoni

Problema "Pietoni" cere dezvoltarea unui program care să analizeze o imagine digitală alb-negru pentru a detecta prezența pietonilor, identificând secțiunea de dimensiune $k \times k$ care conține numărul maxim de "unități". Programul va returna numărul maxim de unități din această secțiune, precum și coordonatele colțului din stânga sus al secțiunii.

Camioane

Compania de transport „MoldCargo” exportă fructe și legume în diferite țări cu comioane frigorifice. Pentru o bună păstrare a produselor pe parcursul drumului acestea trebuie transportate la o temperatură optimă (diferite fructe și legume au cerințe diferite). Compania are o serie de N camioane, numerotate de la 1 la N, fiecare camion este încărcat cu un anumit soi de fructe sau legume care trebuie expediate către o anumită țară. Pentru fiecare camion se cunoaște temperatura optimă pentru produsele pe care le conține p_i, precum și temperatura curentă din interiorul camionului t_i.

Compania a implementat un sistem de control al temperaturii prin care managerul companiei poate gestiona regimul termic în camioane de la distanță. Astfel, managerul poate trimite comenzi către sistemul de aer condiționat din camion și ajusta temperatura din fiecare camion astfel încât ea să corespundă temperaturii optime pentru produsele din camion. Pentru a optimiza acest proces sistemul de gestiune este realizat astfel încât managerul poate da comenzi de a crește sau scădea temperatura într-o serie consecutivă de camioane cu 1 unitate. Seria de camioane poate conține un număr consecutiv de camione, seria poate conține și un singur camion.

Ajutați managerul să identifice numărul minim de comenzi necesare pentru a ajusta temperatura din camioane la temperatura optimă recomandată pentru a păstra produsele proaspete.

Sarcină: Elaborați un program, care va determina numărul minim de comenzi necesare pentru a ajusta temperatura curentă din camioane t_i la temperatura optimă p_i.

Date de intrare: Prima linie a intrării standard conține un număr natural N – numărul de camioane. A doua linie conține N numere întregi nenegative separate prin spații ce reprezintă temperatura optimă de păstrare p_i. A treia linie conține N numere întregi nenegative separate prin spații ce reprezintă temperatura curentă t_i.

Date de ieşire: Ieşirea standard va conţine pe o singură linie numărul minim de comenzi ce trebuie efectuate de manager pentru a regla temperatura din camioane.

Restricţii: $0 \leq N \leq 100000$; $0 \leq p_i, t_i < 10000$. Setul de date inițiale admite o singură soluţie a problemei. Restricțiile referitoare la timpul de execuție și volumul utilizat de memorie sunt date în descrierea generală a problemelor propuse pentru rezolvare. Fişierul sursă va avea denumirea `camion.pas`, `camion.c` sau `camion.cpp`.

Exemplu:

Intrare	*Ieşire*
5 1 5 3 3 4 1 2 2 2 1	5

Explicații:

Avem 5 camioane. Liniile 2 și 3 conțin temperaturile optime și curente. Putem observa câteva seturi de comenzi necesare pentru a aduce temperaturile curente la temperaturile optime. Primul set cuprinde comenzile:

I comandă – Crește cu 1 unitate temperatura în camioanele 2..5: 1 3 3 3 2

II comandă – Crește cu 1 unitate temperatura în camioanele 2..5: 1 4 4 4 3

III comandă – Crește cu 1 unitate temperatura în camioanele 2..5: 1 5 5 5 4

IV comandă – Scade cu 1 unitate temperatura în camioanele 3..4: 1 5 4 4 4

V comandă – Scade cu 1 unitate temperatura în camioanele 3..4: 1 5 3 3 4

Al doilea set posibil de comenzi este:

I comandă – Crește cu 1 unitate temperatura în camioanele 2..5: 1 3 3 3 2

II comandă – Crește cu 1 unitate temperatura în camionul 2: 1 4 3 3 2

III comandă – Crește cu 1 unitate temperatura în camionul 2: 1 5 3 3 2

IV comandă – Scade cu 1 unitate temperatura în camionul 5: 1 5 3 3 3

V comandă – Scade cu 1 unitate temperatura în camionul 5: 1 5 3 3 4

Astfel, numărul minim de comenzi necesare pentru a ajusta temperatura din camioane este egal cu 5.

Diamant

Indira este un designer de interior care lucrează în prezent la un proiect de decorare a magazinului de bijuterii cu pietre prețioase „Diamant”. În proiect podeaua magazinului va fi decorată cu gresie albă și neagră. Indira dorește să așeze gresia în așa mod încât pe podea să se formeze forme specifice numite „diamante”. O formă de diamant se formează prin rotirea unui pătrat cu 45 de grade. Pentru ca designul să fie atrăgător din punct de vedere vizual, Indira vrea să se asigure că diamantele au o formă perfectă. Un diamant este considerat perfect dacă cele patru margini ale sale sunt formate doar din gresie neagră, iar interiorul său este complet umplut cu gresie albă.

Indira are o schiță a podelei sălii cu dispunerea plăcilor de gresie. Ea trebuie să știe câte diamante perfecte poate crea în sală. Deoarece deschiderea magazinului se apropie, ea are nevoie de ajutorul dvs pentru a scrie un program care să numere numărul de diamante perfecte de pe schița podelei.

Schița conține dimensiunile podelei magazinului, reprezentate prin n rânduri și m coloane. Fiecare celulă este fie neagră (#), fie albă (.). Sarcina dvs. este de a scrie un program care să primească ca intrare schița și care să returneze numărul de diamante perfecte.

Sarcină: Elaborați un program, care va determina numărul de diamante perfecte din schița podelei.

Date de intrare: Prima linie a intrării standard conține două numere naturale n și m – dimensiunea podelei. Următoarele n linii conțin m caracterele # sau . ce descriu schița podelei.

Date de ieşire: Ieşirea standard va conţine pe o singură linie numărul de diamante perfecte din schiță.

Restricţii: $1 \leq n, m \leq 2000$. Setul de date iniţiale admite o singură soluţie a problemei. Restricțiile referitoare la timpul de execuție și volumul utilizat de memorie sunt date în descrierea generală a problemelor propuse pentru rezolvare. Fişierul sursă va avea denumirea `diamant.pas`, `diamant.c` sau `diamant.cpp`.

Exemplu:

Intrare	*Ieşire*
5 8 .#...#.. #.#.#.#. .#.#...# ..#.#.#. .....#..	3

Explicații:

În exemplu prezentat pot fi evidențiate trei diamante perfecte; două mici (ce conțin o singură placă albă – un singur punct) și un diamant mai mare (ce conține 5 plăci albe – 5 puncte).

Pietoni

Opticron este un inginer de vedere automată care lucrează pentru o companie ce dezvoltă vehicule autonome. Pentru a detecta pietonii pe drum Opticron vrea să dezvolte un algoritm de detectare a pietonilor. Pentru a realiza acest lucru, el a luat un set de imagini digitale capturate de camerele de bord ale vehiculului autonom. Imaginile sunt captate în alb-negru, unde fiecare pixel este fie negru, fie alb (0 sau 1). Analizând imaginile, Opticron a observat că pietonii din imagini au o anumită dimensiune și formă, ceea ce l-a determinat să caute pietonii într-o secțiune de dimensiune fixă (k x k). Secțiunea cu numărul maxim de unități va indica poziția pietonului în imagine.

Ajutați inginerul să detecteze eficient prezența unui pieton în imagine prin determinarea unei secțiuni cu numărul maxim de unități (reprezentând pietonul) și să afișeze coordonatele poziției pietonului (rândul și coloană colțului din stânga sus al secțiunii). *Dacă au fost detectați mai mulți pietoni să se returneze pietonul pentru care indicii de coloană, prioritar, și apoi de rând sunt cei mai mici.*

Sarcină: Elaborați un program, care va determina coordonatele colțului din stânga sus a secțiunii de dimensiunea k x k ce conține numărul maxim de unități.

Date de intrare: Prima linie a intrării standard conține trei numere naturale n, m și k. n și m reprezintă dimensiunea imaginii, k este dimensiunea secțiunii. Următoarele n linii conțin m cifre (0 sau 1).

Date de ieșire: Ieșirea standard va conține două linii. Pe prima linie se va afișa numărul maxim de unități din secțiuni. Pe a doua linie se vor afișa coordonatele colțului din stânga sus a secțiunii ce conține numărul maxim de unități.

Restricţii: $1 \leq n,\ m \leq 500;\ 1 \leq k \leq 500; k \leq (n, m)$. Setul de date inițiale admite o singură soluție a problemei. Restricțiile referitoare la timpul de execuție și volumul utilizat de memorie sunt date în descrierea generală a problemelor propuse pentru rezolvare. Fișierul sursă va avea denumirea `pieton.pas`, `pieton.c` sau `pieton.cpp`.

Exemplu:

Intrare

```
6 5 3
0 0 1 1 1
1 0 1 0 1
0 1 1 1 1
1 1 1 0 0
1 1 1 1 1
0 0 0 0 1
```

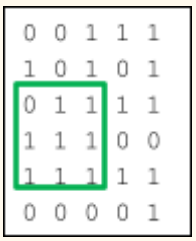
Ieșire

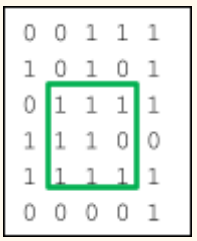
```
8
3 1
```

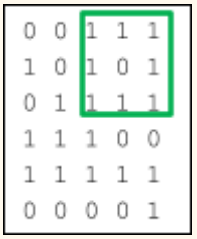
Explicații:

Avem o imagine de dimensiunea 6 x 5, iar secțiunea este de dimensiunea 3 x 3. În prima secțiune numărul de unități este egal cu 5.

Analizând matricea se poate observa că numărul maxim de unități este în trei secțiuni (a se vedea imaginile de mai jos).







În cazul când se determină mai multe secțiuni cu număr maxim de unități se va afișa coordonatele colțului din stânga sus a secțiunii cu cei mai mici indici de coloană și rând. Astfel prima secțiune verde are coordonatele 3:1, a doua secțiune verde are coordonatele 3:2, iar a treia secțiune verde are coordonatele 1:3, deci soluția este 3:1 pentru că are indicile coloanei cel mai mic.

CLASA 10

Ziua 2

Puterea

Problema "Puterea" cere dezvoltarea unui program care să ajute la calcularea puterii la care apare un număr în descompunerea în factori primi a unui factorial. Programul va returna acest exponent ca un număr natural.

Practica de specialitate

Problema "Practica de specialitate" cere dezvoltarea unui program care să determine numărul maxim de studenți care pot fi trimiși pentru stagiul de practică la hoteluri pe litoralul Mării Mediterane, suma totală minimă pe care universitatea trebuie să o plătească, și rețeaua specifică de hoteluri unde studenții își vor desfășura practica, având în vedere anumite reguli de selecție a hotelurilor.

Spațiul de stocare

Problema "Spațiul de stocare" cere dezvoltarea unui program care să găsească aria maximă a unui dreptunghi care poate fi format din barele adiacente ale unei histograme, unde fiecare bară reprezintă înălțimea unui raft din depozitul unei companii de logistică. Programul va returna această arie maximă.

Puterea

Gabriela este foarte pasionată de matematică, în special de descompunerea numerelor naturale în produs de factori primi, calcularea factorialului unui număr etc. Ea își propune să calculeze puterea numărului prim `p` în descompunerea în factori primi a numărului $n!$.

Sarcină: Elaborați un program care ar ajuta-o pe Gabriela să calculeze la ce putere apare `p` în descompunerea în factori primi a lui $n!$.

Date de intrare: Intrarea standard conține pe o singură linie două numere naturale `n` și `p` separate printr-un spațiu.

Date de ieșire: Ieșirea standard va conține pe o singură linie un număr natural `k` – puterea la care apare `p` în descompunerea în factori primi a lui $n!$.

Restricții: `1 ≤ n ≤ 2147483647, 2 ≤ p < 100000.` Restricțiile referitoare la timpul de execuție și volumul utilizat de memorie sunt date în descrierea generală a problemelor propuse pentru rezolvare. Fișierul sursă va avea denumirea `putere.pas, putere.c` sau `putere.cpp`.

Exemple:

1)

Intrare	*Ieșire*
`15 7`	`2`

2)

Intrare	*Ieșire*
`25 5`	`6`

Explicație:

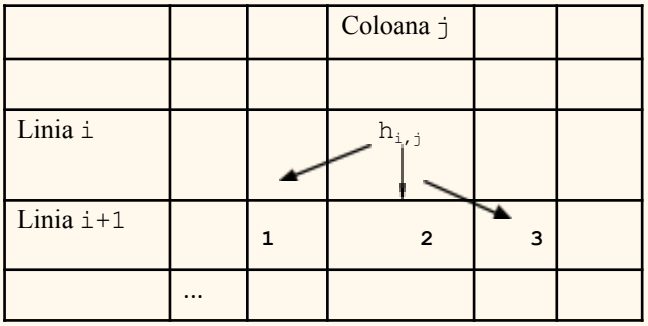
$15! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6 \cdot 7 \cdot 8 \cdot 9 \cdot 10 \cdot 11 \cdot 12 \cdot 13 \cdot 14 \cdot 15 = 1307674368000 =$

$= 2^{11} \cdot 3^6 \cdot 5^3 \cdot 7^2 \cdot 11 \cdot 13$. Deci, în descompunerea în factori primi a numărului 15!, numărul prim 7 are puterea 2.

Practica de specialitate

Universitatea pregătește specialiști la programul de studii *Servicii hoteliere, turism şi agrement* și intenționează să trimită studenții, pentru a-și desfășura stagiul de practica de specialitate, pe litoralul mării Mediterane, în apropiere de orașul Antalya, unde sunt mai multe hoteluri Ultra All Inclusiv. Hotelurile sunt aranjate în `n` linii și `m` coloane. Pentru fiecare student, universitatea achită o sumă `S` care depinde de linia în care se află hotelul. Un hotel poate asigura practica de specialitate pentru `x` studenți. Se știe că sunt hoteluri care nu primesc studenți la practica de specialitate. Decanul facultății, preconizează să trimită la practică pe acel litoral un număr maxim de studenți respectând următoarele condiții:

- în mod obligatoriu va fi ales un hotel din prima linie;
- hotelurile alese reprezintă o rețea formată dintr-un șir de linii consecutive alese astfel: dacă a fost ales hotelul $h_{i,j}$ din linia `i`, atunci din linia `i+1` se poate alege un hotel care se află sau pe diagonală la stânga sau vertical în jos sau pe diagonală la dreapta (vezi figura și ordinea alegerii hotelurilor (1,2,3)) numai în cazul când hotelurile din linia `i+1` și coloanele `j-1,j,j+1` primesc studenți;



Sarcină: Elaborați un program care ar calcula numărul maxim de studenți pe care Universitatea poate să-i trimită pentru a-și desfășura stagiul de practica de specialitate pe litoralul mării Mediterane, suma totală minimă pe care trebuie s-o achite Universitatea și rețeaua de hoteluri în care își vor desfășura practica studenții.

Date de intrare: Prima linie a intrării standard conține două numere naturale despărțite printr-un spațiu: *n* (numărul de linii) și `m` (numărul de coloane). Următoarele `n` linii vor conține câte `m` numere naturale pe fiecare linie separate printr-un spațiu – numărul `x` de studenți pe care îi poate primi hotelul pentru a-și desfășura practica de specialitate (pentru hotelurile care nu primesc studenți la practică de specialitate se indică 0). Linia `n+2` va conține `n` numere naturale – suma pe care o achită universitatea per student începând cu linia de hoteluri 1.

Date de ieșire: Ieșirea standard va conține pe prima linie un număr natural – numărul maxim de studenți pe care decanul îi poate trimite pe litoralul mării Mediterane pentru a

desfășura practica de specialitate. Pe linia a doua se va afișa suma totală minimă pe care trebuie s-o achite Universitatea pentru practica studenților. Începând cu linia a treia, se vor afișa coordonatele hotelurilor începând cu hotelul din prima linie (numărul liniei, numărul coloanei separate printr-un spațiu). Se va ține cont de următoarele: dacă suma totală minimă pentru numărul maxim de studenți se poate obține în mai multe moduri, atunci se va afișa soluția pentru care numărul de ordine a coloanei în care se află primul hotel este cel mai mic.

Restricții: `4 ≤ n,m ≤ 100, 0 ≤ x ≤ 100, 1 ≤ S ≤ 250.` Restricțiile referitoare la timpul de execuție și volumul utilizat de memorie sunt date în descrierea generală a problemelor propuse pentru rezolvare. Fișierul sursă va avea denumirea `hotel.pas, hotel.c` sau `hotel.cpp`.

Exemple:

Intrare	*Ieşire*
4 6 9 21 8 41 5 24 20 0 0 0 1 18 0 1 0 0 0 0 0 0 0 5 5 5 100 75 50 25	42 3650 1 2 2 1 3 2
5 5 0 0 2 0 0 0 3 1 3 0 0 0 2 0 0 0 0 3 0 0 0 4 0 4 0 100 75 50 25 10	14 640 1 3 2 2 3 3 4 3 5 2

Explicații: Pentru primul exemplu, numărul maxim de studenți se poate obține în trei moduri alegând hotelurile: (a) (1,2), (2,1), (3,2) cu suma totală 3650=21*100+20*75+1*50; b) (1,4), (2,5) cu suma totală 4175=41*100+75; (c) (1,6), (2,6) cu suma totală 3750=24*100+18*75. Așa cum, suma totală minimă este 3650, se va alege rețeaua de hoteluri (a). Se poate observa că din hotelurile (1,3), (3,2), (2,5), (2,6) nu se poate efectua deplasarea spre linia următoare.

Pentru exemplul 2, numărul maxim de studenți se poate obține în următoarele moduri alegând hotelurile:

a) (1,3), (2,2), (3,3), (4,3), (5,2); b) (1,3), (2,2), (3,3), (4,3), (5,4);

c) (1,3), (2,4), (3,3), (4,3), (5,2); d) (1,3), (2,4), (3,3), (4,3), (5,4).

Ținând cont de ordinea alegerii hotelurilor (1,2,3, vezi figura) se va afișa varianta a).

Spațiul de stocare

Compania de logistică "DepoMax" dorește să optimizeze spațiul de stocare din depozitul său pentru a face față cererii în creștere de produse. Depozitul are o serie de rafturi cu înălțimi diferite, dispuse într-o configurație complexă. Compania trebuie să găsească cea mai mare suprafață dreptunghiulară disponibilă pentru stocarea produselor.

Sarcină: Fiecare raft din depozit poate fi reprezentat printr-o bară într-o histogramă, unde înălțimea barei corespunde înălțimii raftului. Sarcina constă în a determina aria maximă a unui dreptunghi care poate fi încadrat în această histogramă. Dreptunghiul trebuie să fie format din barele adiacente ale histogramei, iar înălțimea dreptunghiului este dată de cea mai mică bară inclusă.

Date de intrare: Primul rând conține un număr întreg **N**, care reprezintă numărul de bare în histogramă. Pe al doilea rând urmează **N** numere naturale $\mathbf{H_i}$, separate prin spațiu, care reprezintă înălțimile barelor.

Date de ieșire: De afișat aria **A** pentru cel mai mare dreptunghi din histograma dată.

Restricții: $1 \le \mathbf{N} \le 30000$, $0 \le \mathbf{H_i} \le 2*10^9$, $0 \le \mathbf{A} \le 2*10^9$. Restricțiile referitoare la timpul de execuție și volumul utilizat de memorie sunt date în descrierea generală a problemelor propuse pentru rezolvare. Fișierul sursă va avea denumirea `histograma.pas`, `histograma.c` sau `histograma.cpp`.

Exemplu:

1)

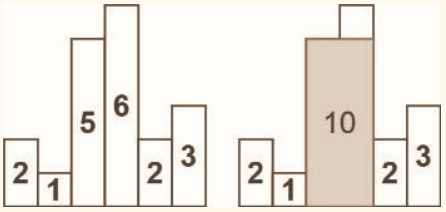
Intrare	*Ieșire*
6 2 1 5 6 2 3	10

2)

Intrare	*Ieșire*
2 2 4	4

Explicații

În primul exemplu, avem o histogramă cu 6 bare și înălțimile următoare: 2, 1, 5, 6, 2, 3. Dreptunghiul cu aria maximă este format din barele 5 și 6. Acest dreptunghi are o lățime de două unități și o înălțime de cinci unități (bara mai mică), rezultând într-o arie de 10 unități. Acesta este cel mai mare dreptunghi care poate fi obținut în histogramă.



Similar pentru exemplul 2: barele 2 și 4 vor fi incluse în dreptunghi, dar înălțimea este determinată de bara 2, deci avem aria 4.

CLASA 11

Ziua 1

Lego

Problema "Lego" cere dezvoltarea unui program care să ajute un copil să construiască un gard din piese Lego de lungimi diferite, astfel încât să minimizeze diferența absolută între lungimea dorită a gardului și lungimea reală a gardului construit. Programul va returna această diferență minimă.

Experiment

Problema "Experiment" cere crearea unui program care să ajute o studentă pasionată de fizică să determine dacă un set de măsurări face ca experimentul să fie considerat reușit sau nu. Un experiment este reușit dacă măsurările sunt numere distincte care pot fi rearanjate într-un șir de numere consecutive. Programul va afișa "DA" și valoarea maximă din măsurători dacă experimentul este reușit, sau "NU" și numărul de măsurători distincte dacă experimentul nu este reușit.

Drumul la școală

Problema "Drumul la școală" cere dezvoltarea unui program care să ajute o elevă, Alexandra, să determine lungimea celui mai scurt drum posibil de la casa ei la școală și numărul de astfel de drumuri scurte, distincte prin cel puțin o microzonă, pe care le poate lua. Drumurile sunt reprezentate pe o hartă a orașului, care este un dreptunghi împărțit în microzone ce pot fi fie libere, fie ocupate de clădiri. Programul va calcula și returna lungimea celui mai scurt drum și numărul acestor drumuri, ajustat printr-o operație specificată.

Lego

De ziua sa de naștere, Alexandru a primit un set specific de lego. Fiecare piesă lego din set dat este de lungimea a_1, a_2 sau a_3. In total, în set sunt b_1 piese de lego de lungime a_1, b_2 de piese de lego de lungime a_2 si b_3 de piese lego de lungime a_3. Prima construcție pe care vrea Alexandru să o construiască este un gard din piese lego de lungime K. Din păcate, acest lucru nu este permanent posibil cu lungimile pieselor de lego pe care le are în set. De aceea el încearcă să construiască un gard de lungime cât mai aproape de K. De exemplu, să presupunem ca Alexandru a construit un gard de lungime T diferită de K. Scopul lui Alexandru este să minimizeze valoarea $|T - K|$.

Sarcină. Elaborați un program care având numărul de piese lego de fiecare tip și lungimea K, determină valoarea minim posibilă a expresiei $|T - K|$.

Date de intrare. Prima și unica linie a intrării standard conține șapte numere întregi separate prin spațiu: $a_1\ a_2\ a_3\ b_1\ b_2\ b_3\ K$.

Date de ieșire. Ieșirea standard va conține pe o singură linie valoarea minimă a expresiei $|T - K|$ pe care o poate obține Alexandru.

Restricții. $2 \le K \le 3 \bullet 10^{13}$, $1 \le a_i \le 10^9$, $1 \le i \le 3$ $1 \le b_i \le 10^4$, $1 \le i \le 3$. Restricțiile referitoare la timpul de execuție și volumul utilizat de memorie sunt date în descrierea generală a problemelor propuse pentru rezolvare. Fișierul sursă va avea denumirea `lego.pas`, `lego.c` sau `lego.cpp`.

Exemplu 1.

Intrare	*Ieșire*
`17 29 31 10 10 10 245`	`0`

Explicație: Putem utiliza *2* piese lego de lungime *17, 3* piese lego de lungime *29* și *4* piese lego de lungime *31*. În acest caz vom avea *T=2*17+3*29+4*31=245*, valoare care coincide cu valoarea lui *K*. Respectiv $|T - K|=0$

Exemplu 2.

Intrare	*Ieșire*
`17 29 31 10 10 10 55`	`3`

Explicație: Soluția optimă este utilizarea a *2* piese lego de lungime *29*. În cazul dat *T=2*29=58*, iar $|T - K|=|58 - 55|=3$

Punctarea. Testele vor fi grupate în patru categorii, după cum urmează:

Cotă Punctaj	a_i	b_i	K
21%	$\leq 10^9$	$\leq 10^2$	$\leq 3*10^{11}$
7%	≤ 2	$\leq 10^4$	$\leq 6*10^4$
37%	≤ 500	$\leq 10^4$	$= 15*10^6$
35%	$\leq 10^9$	$\leq 10^4$	$\leq 3*10^{13}$

Experiment

Anabela este pasionată de fizică. În fiecare zi după lecții ea muncește la un experiment și efectuează N măsurări $M=(M_1, M_2, ..., M_N)$ pentru a vedea dacă experimentul a reușit ori nu. Rezultatul măsurărilor Anabela le notează în carnet apoi le analizează. Experimentul se consideră *reușit* dacă șirul măsurărilor $M=(M_1, M_2, ..., M_N)$ este format din numere distincte care pot fi rearanjate astfel încât să alcătuiască un șir de numere consecutive.

De exemplu, experimentul cu măsurările $M=(8, 4, 5, 6, 7, 10, 9)$ este reușit deoarece valorile obținute pot fi rearanjate astfel încât să formeze un șir de numere consecutive $(4, 5, 6, 7, 8, 9, 10)$, pe când experimentul cu măsurările $M=(9,5,6,5,8)$ nu este un experiment reușit deoarece valorile obținute $(9,5,6,5,8)$ nu pot fi aranjate astfel încât să formeze un șir de numere consecutive.

Sarcină. Elaborați un program care pentru N măsurări determină dacă experimentul este reușit sau nu. Dacă experimentul este reușit atunci se va afișa expresia „DA” și elementul maximal din măsurările $M=(M_1, M_2, ..., M_N)$, altfel se va afișa expresia „NU” și numărul de măsurări cu valori distincte.

Date de intrare. Prima linie a intrării standard conține un număr întreg N – numărul de măsurări efectuate de Anabela. Următoarea linie conține N întregi separate prin spațiu, valorile măsurărilor înregistrate de Anabela.

Date de ieșire. Ieșirea standard va conține 2 linii. Dacă experimentul este reușit atunci pe prima line se va afișa „DA”, iar pe a doua linie măsurarea cu valoarea maximă. Dacă experimentul nu este reușit atunci pe prima line se va afișa „NU”, iar pe a doua linie numărul de măsurări distincte.

Restricții. $1 \leq N \leq 5000$, $1 \leq M_i \leq 1000$, $1 \leq i \leq N$. Restricțiile referitoare la timpul de execuție și volumul utilizat de memorie sunt date în descrierea generală a problemelor propuse pentru rezolvare. Fișierul sursă va avea denumirea `experiment.pas`, `experiment.c` sau `experiment.cpp`.

Exemplu 1.

Intrare	*Ieșire*
5 66 64 65 68 67	DA 68

Explicație: Secvența de măsurări formează un experiment reușit deoarece valorile obținute pot fi aranjate astfel încât să obținem un șir de numere consecutive $(64, 65, 66, 67, 68)$, iar măsurarea cu valoarea maximă din acest șir este 68.

Exemplu 2.

Intrare	*Ieșire*
7 66 44 55 57 55 44 83	NU 3

Explicație: Secvența de măsurări nu reprezintă un experiment reușit deoarece valorile obținute nu pot fi aranjate astfel încât să obținem un șir de numere consecutive. Numărul de măsurări cu valori distincte este 3: *66 57 83*

Drumul la școală

Fiind o adeptă al modului sănătos de viață, Alexandra se duce la școală doar pe jos.

Harta orașului în care locuiește Alexandra reprezentă un dreptunghi împărțit în microzone de formă pătrată cu laturi de o unitate de lungime.

În memoria calculatorului această hartă este reprezentată printr-un tablou cu n rânduri și m coloane. Elementele acestui tablou sunt caractere ce au următoarea semnificație:

`'.'` − microzonă liberă, prin ea Alexandra poate trece;

`'X'` − microzonă ocupată de clădiri, prin ea Alexandra nu poate trece.

În fiecare zi, Alexandra pornește la școală din microzona în care se află casa în care ea locuiește și ajunge în microzona în care se află școala în care ea învață.

Prin x_c și y_c vom nota, respectiv, numărul rândului și numărului coloanei în care se află elementul tabloului ce reprezintă microzona în care se află casa Alexandrei.

Prin x_s și y_s vom nota, respectiv, numărul rândului și numărului coloanei în care se află elementul tabloului ce reprezintă microzona în care se află școala în care învață Alexandra.

Prin definiție, microzonele în care se află casa Alexandrei și școala la care ea învață, se consideră ca fiind libere.

În drumul spre școală, la fiecare pas, Alexandra trece din microzonă curentă în una din microzonele vecine, care, evident, trebuie să fie liberă și să aibă o latură comună cu cea curentă.

Lungimea drumului parcurs de Alexandra de acasă până la școală se definește ca numărul de microzone distincte prin care ea a trecut, inclusiv și cele în care se află casa și școala.

Dorind să admire cât mai multe clădiri frumoase ale orașului în care locuiește, Alexandra și-ar dori ca în fiecare zi să meargă la școală pe un drum nou, adică pe un drum care s-ar deosebi de cele parcurse anterior de ea prin cel puțin o microzonă. Totodată, pentru a economisi timpul, Alexandra dorește ca drumurile parcurse de ea să fie de una și aceiași lungime minimal posibilă.

Prin L vom nota lungimea celor mai scurte drumuri, iar prin M − numărul tuturor drumurilor de lungime L.

Sarcină. Scrieți un program care calculează lungimea celui mai scurt drum L și numărul M unor astfel de drumuri. În scopuri didactice programul va furniza la ieșirea standard nu numărul M propriu-zis, ci numărul

$$D = M\,(10^9 + 7)\,.$$

Amintim că operația **mod** din formula de mai sus calculează restul împărțirii numărului întreg M la numărul întreg $(10^9 + 7)$.

Date de intrare. Prima linie a intrării standard conține numerele n, m, x_c, y_c, x_s, y_s, separate prin spațiu. Următoarele n linii ale intrării standard conțin elementele tabloului ce reprezintă microzonele hărții. Fiecare din aceste linii conține câte un șir de caractere de lungimea m format din caracterele '.' și 'X'.

Date de ieșire. Ieșirea standard va conține pe o singură linie numerele întregi L, D, separate prin spațiu.

Restricții. $1 \le n, m \le 1000$; $1 \le x_c, x_s \le n$; $1 \le y_c, y_s \le m$; $(x_c, y_c) \ne (x_s, y_s)$. Se garantează existența cel puțin a unui drum ce corespunde condițiilor din enunțul problemei. Restricțiile referitoare la timpul de execuție și volumul utilizat de memorie sunt date în descrierea generală a problemelor propuse pentru rezolvare. Fișierul sursă va avea denumirea `drumul.pas`, `drumul.c` sau `drumul.cpp`.

Exemplul 1.

Intrare

```
4 5 1 1 4 5
.....
.XXX.
.XXX.
.....
```

Ieșire

```
8 2
```

Explicație: Alexandra poate merge pe 2 drumuri distincte fiecare de lungime minimă 8

```
O....
OXXX.
OXXX.
OOOOO
```

```
OOOOO
.XXXO
.XXXO
....O
```

Exemplul 2.

Intrare

```
4 5 1 1 4 5
.....
.X.X.
.X.X.
.....
```

Ieșire

```
8 3
```

Explicație: Alexandra poate merge pe 3 drumuri distincte fiecare de lungime minimă 8

```
O....
OX.X.
OX.X.
OOOOO
```

```
OOOOO
.X.XO
.X.XO
....O
```

```
OOO..
.XOX.
.XOX.
..OOO
```

Punctarea. Testele vor fi grupate in trei categorii, după cum urmează:

A. 20% din punctaj: $n, m \le 8$.

B. 40% din punctaj: $n \le 8$.

C. 40% din punctaj: fără restricțiile din categoriile precedente.

CLASA 11

Ziua 2

Poster

Problema "Poster" cere dezvoltarea unui program care să ajute Agenția Națională pentru Curriculum și Evaluare să determine în câte moduri diferite poate fi creat un poster specific, "ORI2023", dintr-un set de panouri care conțin caractere diverse. Programul va lua un șir de caractere ca intrare și va returna numărul de moduri în care posterul poate fi format, ajustat printr-o operație matematică specificată.

Turneu

Problema "Turneu" implică organizarea unui turneu de trântă cu scopul de a-l face cât mai interesant pentru telespectatori. Organizatorii trebuie să selecteze participanții bazându-se pe două caracteristici: puterea și faima. Scopul este de a maximiza suma faimei participanților, în timp ce minimizează diferența de putere între cel mai puternic și cel mai slab concurent. Programul trebuie să calculeze valoarea optimă a acestei formule pentru a ajuta organizatorii în luarea deciziilor.

Orașe

Problema "Orașe" se concentrează pe optimizarea unei călătorii a lui Petrică prin Europa pentru a maximiza satisfacția sa. Începând din orașul 1 și dorind să ajungă în orașul N, Petrică poate alege să călătorească cu trenul la orașul imediat următor sau să zboare către un alt oraș. Fiecare modalitate de transport are un cost asociat în unități de satisfacție. Programul trebuie să determine maximul de satisfacție pe care Petrică îl poate obține.

O	R	I	2	0	2	3

Poster

Cu ocazia Olimpiadelor Republicane, Agenția Națională pentru Curriculum și Evaluare (ANCE) pregătește postere pentru fiecare olimpiadă. Un poster este format din mai multe panouri fiecare având imprimată câte un caracter. De exemplu pentru Olimpiada Republicană la Informatică 2023, posterul va conține 7 panouri formând șirul de caractere „ORI2023". Având N panouri aranjate secvențial care conțin și caractere de la abrevierile altor olimpiade, ANCE se întreabă în câte moduri M se poate de creat posterul „ORI2023" din cele N panouri, dacă vom utiliza panourile în ordinea în care sunt aranjate.

Sarcină. Elaborați un program care pentru un șir de caractere a de lungime N, determină în câte moduri M poate fi creat posterul „*ORI2023*". Având un număr mare de olimpiade, ANCE înțelege că M poate fi destul de mare, respectiv rezultatul afișat va fi M ***mod*** (10^9+7). Amintim că operația **mod** calculează restul împărțirii numărului întreg M la numărul întreg $(10^9 + 7)$.

Date de intrare. Prima și singura linie a intrării standard conține un șir de lungime N format din cifre și litere ale alfabetului englez.

Date de ieșire. Ieșirea standard va conține pe o singură linie un număr întreg – restul împărțirii la 10^9+7 a numărului de moduri în care se poate forma posterul ORI2023.

Restricții. $7 \leq N \leq 10^5$. Restricțiile referitoare la timpul de execuție și volumul utilizat de memorie sunt date în descrierea generală a problemelor propuse pentru rezolvare. Fișierul sursă va avea denumirea `poster.pas`, `poster.c` sau `poster.cpp`.

Exemplu.

Intrare	*Ieșire*
`2AORII20203`	`2`

Explicație: Posterul „ORI2023" poate fi format în 2 moduri:

1. Din panourile cu indicii *3, 4, 5, 7, 8, 9, 11*
2. Din panourile cu indicii *3, 4, 6, 7, 8, 9, 11*

Punctarea. Testele vor fi grupate în patru categorii, după cum urmează:

Cotă Punctaj	N
14%	≤ 8
19%	≤ 50
37%	≤ 250
30%	$\leq 100\,000$

Turneu

Un turneu de trântă va fi organizat și televizat în întreaga țară, iar sarcina organizatorilor este să selecteze participanții astfel încât turneul să fie cât mai interesant. Drept participanți au aplicat N sportivi, fiecare din ei având doua caracteristici – puterea și faima. Astfel, al i-lea sportiv are puterea P_i și faima F_i.

Alegând participanții, organizatorii nu doresc ca puterea dintre participanți să difere prea mult, altfel turneul nu va fi palpitant. Pe de altă parte, organizatorii trebuie să țină cont de faima participanților pentru a atrage cât mai mulți spectatori. Astfel ei au decis să aleagă participanții în modul următor:

Fie P_{min} puterea celui mai slab participant, iar P_{max} puterea celui mai puternic participant. Fie F suma faimei tutor participanților. Organizatorii vor să selecteze participanții din grupul de sportivi care au aplicat, astfel încât valoarea $F - (P_{max} - P_{min})$ să fie cât mai mare. Pentru a rezolva această sarcină complicată, ei apelează la ajutorul vostru.

Sarcină. Scrieți un program care calculează valoarea maximă $F - (P_{max} - P_{min})$ dacă participanții sunt aleși în mod optim.

Date de intrare. Prima linie a intrării standard conține un număr întreg N – numărul de sportivi ce au aplicat. Următoarele N linii conțin câte două numere – a i-a linie conține numerele P_i și F_i, care reprezintă puterea și faima sportivului i.

Date de ieșire. Ieșirea standard va conține un singur număr – valoarea maximă $F - (P_{max} - P_{min})$.

Restricții. $2 \le N \le 5 \cdot 10^5$, $1 \le P_i \le 10^{15}$, $1 \le F_i \le 10^9$. Restricțiile referitoare la timpul de execuție și volumul utilizat de memorie sunt date în descrierea generală a problemelor propuse pentru rezolvare. Fișierul sursă va avea denumirea `turneu.pas`, `turneu.c` sau `turneu.cpp`.

Exemplul 1.

Intrare

```
3
3 3
12 2
5 5
```

Ieșire

```
6
```

Exemplul 2.

Intrare

```
6
5 1
2 5
11 3
10 1
5 2
6 3
```

Ieșire

```
7
```

Explicații: În primul exemplu, o selecție optimă a participanților este să alegem sportivul 1 și 3. Astfel, $P_{max} = 5$, $P_{min} = 3$ și $F = F_1 + F_3 = 8$, deci $F - (P_{max} - P_{min}) = 8 - (5 - 3) = 6$. Putem observa că nici o altă selecție nu va fi mai bună.

Punctarea. Testele vor fi grupate în 4 categorii, după cum urmează:

A. 10% din punctaj: $N \leq 16$.
B. 20% din punctaj: $N \leq 300$.
C. 20% din punctaj: $N \leq 5000$.
D. 50% din punctaj: Fără restricții adiționale.

Orașe

În vacanța de vară, Petrică își planifică o excursie în Europa. Convențional vom numerota orașele cu *1, 2, ..., N*. Inițial Petrică se află în orașul 1 și vrea neapărat să ajungă în orașul *N*, însă posibil vizitând și alte orașe în drumul său. În funcție de atracțiile orașului Petrică va primi o anumită doză de satisfacție pentru fiecare oraș vizitat. Astfel vizitând orașul k ($1 \le k < N$), Petrică va primi a_k unități de satisfacție. Aflându-se într-un oarecare oraș k ($1 \le k < N$), Petrică are următoarele posibilități de a și continua drumul către orașul *N*:

1. Poate călători către orașul k+1 cu trenul. Deoarece drumul este lung, Petrică va primi în acest caz cu C_1 unități de satisfacție mai puțin.
2. Poate călători către orașul t cu avionul, ($k < t \le N$, *k divide t*). În acest caz Petrica va primi cu $C_2 \bullet \frac{t}{k}$ unități de satisfacție mai puțin.

Sarcină. Elaborați un program care determină satisfacția maximală posibilă pe care Petrică o poate primi călătorind spre orașul *N*.

Date de intrare. Prima linie a intrării standard conține 3 numere întregi separate prin spațiu: N, C_1, C_2. Următoarea linie conține N numere întregi – a_1, a_2, ..., a_N – satisfacțiile pentru fiecare oraș.

Date de ieșire. Ieșirea standard va conține pe o singură linie un număr întreg – satisfacția maximală pe care o poate primi Petrică în drum spre orașul *N*.

Restricții. $2 \le N \le 200000$, $0 \le C_1, C_2 \le 10^9$, $0 \le a_i \le 10^9$. Restricțiile referitoare la timpul de execuție și volumul utilizat de memorie sunt date în descrierea generală a problemelor propuse pentru rezolvare. Fișierul sursă va avea denumirea `orase.pas`, `orase.c` sau `orase.cpp`.

Exemplu.

Intrare	*Ieșire*
`4 9 7` `1 2 0 100`	`80`

Explicație: Petrică va vizita orașele 1, 2 și 4. Din orașul 1 către orașul 2, Petrică va călători cu trenul, iar din orașul 2 către orașul 4 cu avionul. Astfel, va obține satisfacția totală: $1 + 2 + 100 - 9 - 7 \bullet \frac{4}{2} = 80$

Punctarea. Testele vor fi grupate în cinci categorii, după cum urmează:

Cotă Punctaj	N	C_1	C_2
14%	≤ 50	$0 \leq C_1 \leq 10^9$	$0 \leq C_2 \leq 10^9$
5%	$\leq 200\,000$	$C_1 = 0$	$0 \leq C_2 \leq 10^9$
5%	$\leq 200\,000$	$0 \leq C_1 \leq 10^9$	$C_2 = 0$
39%	≤ 2000	$0 \leq C_1 \leq 10^9$	$0 \leq C_2 \leq 10^9$
37%	$\leq 200\,000$	$0 \leq C_1 \leq 10^9$	$0 \leq C_2 \leq 10^9$

CLASA 12

Ziua 1

Hedge fund

Problema "Hedge fund" se axează pe maximizarea profitului unui fond de investiții speculativ care folosește metode cantitative și informații interne pentru a estima prețurile acțiunilor. Fondul poate alege între două strategii de tranzacționare: "Long" și "Short", fiecare cu propriile sale reguli pentru calculul profitului. Există restricții cu privire la numărul total de tranzacții permise într-un anumit număr de zile. Sarcina este de a dezvolta un program care determină profitul maxim posibil în funcție de aceste variabile.

Concursuri de informatică

Problema "Concursuri de informatică" implică găsirea numărului minim de ore pentru care Nicolae trebuie să își țină calculatorul pornit pentru a participa la toate concursurile de informatică online programate, având în vedere că poate porni calculatorul doar un număr limitat de ori și că dorește să economisească energie electrică.

Piramida

Problema "Piramida" cere identificarea numărului total de piramide distincte ce pot fi formate din celule marcate cu 1 pe o foaie de matematică cu dimensiuni date, unde o piramidă este definită ca având nivele cu celule consecutive, și fiecare nivel superior fiind format prin eliminarea celulelor de la extremitățile nivelului inferior.

Hedge fund

„Hedge fund" este un fond special care folosește metode speculative pentru a face profit de scurt timp datorită volatilității stocurilor. Multe fonduri folosesc metode cantitative pentru a aproxima prețurile stocurilor pe perioade mai mari de timp. De asemenea fiind conectate cu multe companii mari, ei pot afla informații interne, care îi va putea ajuta să aproximeze prețul stocurilor în timpul apropiat.

Folosind metodele menționate mai sus, Hedge fund a aproximat cu probabilitate mare că prețul unui stoc în ziua i va fi c_i unități convenționale (u.c.)

Pentru a face profit Hedge fund aplică următoarele două metode:

1. Long: Cumpără un stoc la prețul a u.c. și-l vinde în viitor cu prețul b u.c. (mai mare). În acest caz profitul Hedge Fund este $b - a$ u.c.
2. Short: Împrumută un stoc la prețul a u.c. și apoi imediat îl vinde. În acest caz pentru fiecare zi Hedge Fund va plăti o taxă S u.c. pentru împrumut. După t zile Hedge Fund cumpăra stocul împrumutat la prețul b u.c. (mai mic) și returnează stocul împrumutat și toate taxele aferente. Profitul în acest caz va fi $a-b-t*S$ u.c.

De exemplu presupunem că $c=(10, 20, 10, 60)$ și $S=1$ u.c. Dacă folosim prima metodă și cumpărăm un stoc în ziua *1* la preț de 10 u.c. și vindem în ziua 4 cu prețul de 60 u.c., atunci obținem un profit de 60 – 10 = 50.

Dacă folosim a doua metodă și împrumutăm stocul în ziua *2* cu *20* u.c. și imediat îl vindem, după care îl răscumpărăm în a *3*-a zi cu *10* u.c. , atunci obținem un profit de $20 - 10 - 1*1 = 9$ u.c.

Pentru a nu permite speculări pe piață, entitatea regulatoare permite cel mult M tranzacții (operații Long sau Short) de stocuri în total în cele N zile. De asemenea după cele N zile, toate stocurile trebuie să fie vândute, iar toate stocurile împrumutate să fie returnate.

Sarcină. Elaborați un program care având probabilitățile costurile stocurilor $c = (c_1, c_2, \ldots, c_N)$, taxa pentru împrumut S și numărul de tranzacții admisibile M, determină profitul maxim posibil pe care îl poate obține Hedge fund.

Date de intrare. Prima linie a intrării standard conține numărul T – numărul de teste. Următoarele $2*T$ linii conțin cele T teste. Un test va avea pe o linie valorile N – numărul de zile, M – numărul de tranzacții admisibile și S - taxa pentru o zi de împrumut separate prin spațiu. Următoarea linie din test conține cele N probabilități ale costurilor stocurilor.

Date de ieșire. Ieșirea standard va conține T linii. Fiecare linie va conține un singur număr întreg – profitul maximal pe care îl poate face Hedge fund pentru fiecare test.

Restricții. $1 \leq N \leq 10^5$, $1 \leq M \leq 10^9$, $0 \leq S \leq 10^{14}$, $1 \leq c_i \leq 10^9$, $1 \leq i \leq N$. Restricțiile referitoare la timpul de execuție și volumul utilizat de memorie sunt date în descrierea generală a problemelor propuse pentru rezolvare. Fișierul sursă va avea denumirea `hedge.pas`, `hedge.c` sau `hedge.cpp`.

Exemplu

Intrare	*Ieșire*
2 5 2 1 1 2 1 2 1 5 1 1 20 16 12 8 4	2 12

Explicație

În cazul primului test se va cumpăra stocul în prima zi și se va vinde în a doua, apoi se va cumpăra un stoc în a treia zi și se va vinde în a 4-a zi. Profitul: 2-1 + 2-1 = 2.

În cazul celui de al doilea test, se permite o singură tranzacție. Respectiv Hedge fund va aplica metoda Short, adica va împrumuta stocul din prima zi și imediat îl va vinde cu 20 u.c. timp de 4 zile va plăti 1 u.c. taxa pentru împrumut. În a 5-a zi Hedge fund va răscumpăra stocul cu 4 u.c. Profitul obținut este: 20 – 4 – 4 * 1 = 12.

Punctarea. Testele vor fi grupate în șase categorii, după cum urmează:

Cotă Punctaj	*N*	*S*
5%	≤ 2	$\leq 10^9$
5%	≤ 3	$\leq 10^9$
30%	≤ 1024	$\leq 10^9$
30%	$\leq 10^5$	$= 10^{14}$
10%	$\leq 10^5$	$= 0$
20%	$\leq 10^5$	$\leq 10^9$

Concursuri de informatică

Nicolae este un elev ambițios și dorește să participe la cât mai multe concursuri de informatică online. El știe că în timpul apropiat se vor organiza N concursuri, și că al i-lea ($1 \le i \le N$) concurs va începe peste C_i ore și va dura exact o oră.

Nicolae va participa la concursuri de pe calculatorul lui, care inițial e oprit, iar Nicolae își poate porni sau opri calculatorul în orice moment. Evident, calculatorul lui Nicolae trebuie să fie pornit în timpul fiecărui concurs, iar în rest poate fi fie oprit, fie pornit. Pe de altă parte, calculatorul lui Nicolae are un virus straniu: Nicolae își poate porni calculatorul de doar M ori.

Nicolae vrea să economisească cât mai multă energie electrică, deci vrea să afle care e numărul minim de ore cât timp calculatorul lui trebuie să rămână pornit pentru a putea participa la fiecare concurs.

Sarcină. Scrieți un program care calculează numărul minim de ore cât timp calculatorul lui Nicolae trebuie să fie pornit astfel încât calculatorul va fi pornit în timpul fiecărui concurs, iar Nicolae nu va porni calculatorul de mai mult de M ori.

Date de intrare. Pe prima linie a intrării se află două numere întregi N și M. Următoarele N linii conțin câte un număr. A i-a linie conține numărul C_i, ora la care începe concursul i.

Date de ieșire. Ieșirea standard va conține un singur număr – numărul minim de ore necesar calculatorului lui Nicolae să fie pornit.

Restricții. $1 \le M \le N \le 10^5$, $1 \le C_i \le 10^9$, $C_i < C_{i+1}$. Restricțiile referitoare la timpul de execuție și volumul utilizat de memorie sunt date în descrierea generală a problemelor propuse pentru rezolvare. Fișierul sursă va avea denumirea `concursuri.pas`, `concursuri.c` sau `concursuri.cpp`.

Exemplul 1.

Intrare

```
3 2
2
4
7
```

Ieșire

```
4
```

Explicație: Nicole poate proceda în modul următor: Nicolae va porni calculatorul la ora 2. Apoi Nicolae va participa la primele două concursuri – primul de la ora 2 la 3, și al doilea de la ora 4 la 5. Apoi Nicolae va opri calculatorul la ora 5 și îl va porni din nou la 7 pentru a participa la ultimul concurs, oprind calculatorul imediat după ora 8. Astfel, calculatorul va fi pornit de exact 2 ori, și va sta pornit în total 4 ore. Putem observa că Nicolae nu are o strategie mai optimă.

Exemplul 2.

Intrare

```
3 1
2
4
7
```

Ieșire

```
6
```

Exemplul 3.

Intrare

```
3 3
2
4
7
```

Ieșire

```
3
```

Exemplul 4.

Intrare

```
10 5
1
3
5
7
8
13
14
15
17
20
```

Ieșire

```
12
```

Punctarea. Testele vor fi grupate in trei categorii, după cum urmează:

A. 20% din punctaj: $N \leq 20$, $C_i \leq 20$.

B. 30% din punctaj: $N \leq 5000$.

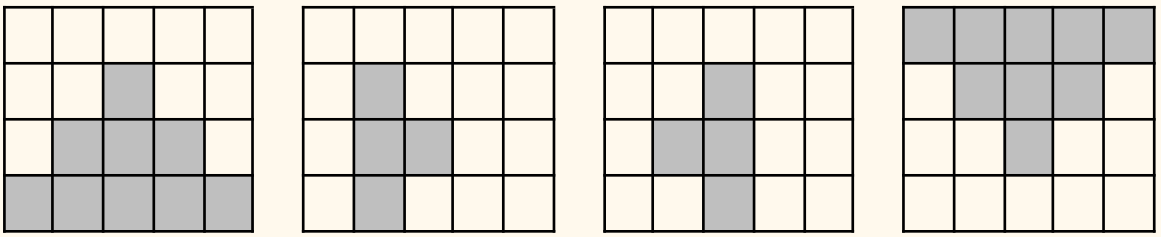
C. 50% din punctaj: Fără restricții adiționale.

Piramida

Pe o foaie de matematică de dimensiune *N* x *M* completată de fratele său mai mic cu cifre de *0* și *1*, Lenuța observă că unele celule (pătrățele) completate cu *1* formează o piramidă. În scopuri didactice vom defini o piramidă de înălțime *K (K > 1)* următoarea figură:

1. Primul nivel este format din *2K-1* celule consecutive (pătrățele de pe foaia de matematică).
2. Nivelul *t* este identic nivelului *t – 1* cu excepția lipsei celulelor extremităților nivelului *t – 1*.
3. O piramidă este formată din K nivele, iar nivelul K (ultimul) conține o singură celulă. Valoarea lui K reprezintă înălțimea piramidei.

Figurile de mai jos reprezintă piramide de înălțimea 3, 2, 2 și 3 respectiv.



Lenuța este curioasă câte piramide pe foaia de matematică poate identifica.

Sarcină. Elaborați un program care pentru un tablou bidimensional cu *N* linii și *M* coloane completate cu *0* sau *1* determină numărul de piramide distincte formate din celule completate cu *1*. O celulă completată cu 1 poate face parte din mai multe piramide.

Date de intrare. Prima linie a intrării standard conține 2 numere întregi N și M – dimensiunile tabloului. Următoarele *N* linii conțin câte *M* caractere de tip *0* sau *1*.

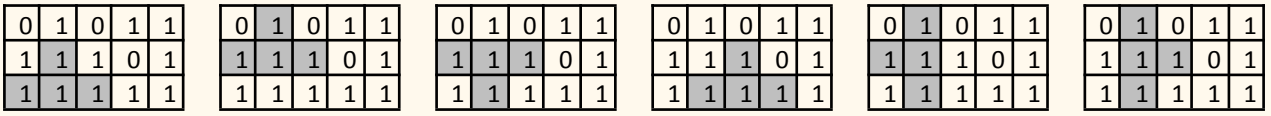
Date de ieșire. Ieșirea standard va conține pe o singură linie un număr întreg – numărul de piramide ce pot fi formate din celulele completate cu *1*.

Restricții. $2 \leq N, M \leq 5000$. Restricțiile referitoare la timpul de execuție și volumul utilizat de memorie sunt date în descrierea generală a problemelor propuse pentru rezolvare. Fișierul sursă va avea denumirea `piramida.pas`, `piramida.c` sau `piramida.cpp`.

Exemplu.

Intrare	*Ieşire*
3 5 01011 11101 11111	6

Explicație: Pot fi identificate următoarele piramide (vezi desenul de mai jos)



Punctarea. Testele vor fi grupate în patru categorii, după cum urmează:

Cotă Punctaj	*N*	*M*
13%	≤ 50	≤ 50
19%	≤ 200	≤ 200
35%	≤ 800	≤ 800
33%	≤ 5000	≤ 5000

CLASA 12

Ziua 2

Plăci tectonice

Problema "Plăci tectonice" explorează modul în care temperatura la vârful unui lanț de munți se schimbă zilnic datorită mișcărilor tectonice care alterează altitudinile vârfurilor. Programul trebuie să calculeze temperatura unui vârf specificat al lanțului pentru fiecare zi, având în vedere regulile pentru modul în care temperatura se schimbă în funcție de altitudine.

Drumuri

Problema "Drumuri" cere identificarea numărului de drumuri posibile într-un graf, astfel încât toate vârfurile pe un drum să aibă culori diferite. Graficul are un număr fix de vârfuri și muchii, iar fiecare vârf este colorat într-una dintr-un set de culori prestabilit.

Tren

Problema "Tren" cere calcularea numărului maxim de stații la care se poate ajunge începând de la prima stație într-un timp maxim dat, având în vedere trei tipuri de trenuri cu viteze și opriri diferite. Scopul este de a optimiza opririle unui nou tip de tren, numit "trenul regional", pentru a maximiza numărul de stații accesibile în timpul alocat.

Plăci tectonice

Cu toții știm că temperatura aerului scade odată cu înălțimea. Andrei este un geolog care examinează un lanț de munți, format din *N*+1 vârfuri. Altitudinea vârfului 0 este $A_0 = 0$ metri, iar altitudinea celui de-al *i*-lea vârf este A_i metri. Andrei a observat că temperatura aerului la vârful 0 este de 0 grade, iar temperatura aerului la celelalte vârfuri respectă următoarea legitate:

- Dacă $A_i > A_{i-1}$, atunci temperatura aerului la vârful i va fi cu $S \cdot |A_i - A_{i-1}|$ grade mai mică decât la vârful $i - 1$.
- Dacă $A_i \leq A_{i-1}$, atunci temperatura aerului la vârful i va fi cu $T \cdot |A_i - A_{i-1}|$ grade mai mare decât la vârful $i - 1$.

Astfel, Andrei poate calcula temperatura aerului la vârful *N*. Din păcate, din cauza activității seismice mare, plăcile tectonice sunt în continuă mișcare, astfel altitudinea vârfurilor se schimbă zilnic. Mai exact, Andrei examinează Q zile și observă că în a *j*-a zi, altitudinea vârfurilor de la L_j la R_j se va schimba cu X_j metri. În alte cuvinte, pentru fiecare k $\left(L_j \leq k \leq R_j\right)$, dacă vârful k avea înălțimea A_k în ziua precedentă, atunci în a *j*-a zi acest vârf va avea înălțimea $A_k + X_j$. Observați că X_j poate lua valori negative.

Andrei vrea să afle temperatura aerului la cel de-al *N*-lea vârf în fiecare zi.

Sarcină. Scrieți un program care calculează temperatura aerului la cel de-al *N*-lea vârf în fiecare zi.

Date de intrare. Pe prima linie a intrării se află patru numere întregi *N, Q, S, T.* Următoarele *N+1* linii conțin câte un număr – a *i*-a linie conține numărul A_{i-1}. Următoarele Q linii conțin câte trei numere – a *j*-a linie conține numerele L_j, R_j, X_j.

Date de ieșire. Ieșirea standard va conține Q linii – a *j*-a linie va conține temperatura aerului la cel de-al *N*-lea vârf în a *j*-a zi.

Restricții.

$1 \leq N, Q \leq 2 \cdot 10^5$, $1 \leq S, T \leq 10^6$, $A_0 = 0$, $|A_i| \leq 10^6$, $1 \leq L_j \leq R_j \leq N$, $|X_j| \leq 10^6$.

Restricțiile referitoare la timpul de execuție și volumul utilizat de memorie sunt date în descrierea generală a problemelor propuse pentru rezolvare. Fișierul sursă va avea denumirea `placi.pas`, `placi.c` sau `placi.cpp`.

Exemplul 1.

Intrare

```
3 5 1 2
0
3
2
7
1 2 -1
1 1 2
2 3 -5
1 2 1
1 3 -5
```

Ieșire

```
-6
-4
6
6
12
```

Explicație: După prima zi, vârfurile vor avea următoarele altitudini: 0 2 1 7. Temperatura aerului la primul vârf va fi -2, deoarece $A_1 > A_0$, astfel temperatura va fi $0 - S \cdot |A_1 - A_0| = 0 - 1 \cdot |2 - 0| = -2$. La al doilea vârf, $A_2 < A_1$, astfel temperatura va fi $-2 + T \cdot |A_2 - A_1| = -2 + 2 \cdot 1 = 0$. La ultimul vârf, $A_3 > A_2$, astfel temperatura va fi $0 - S \cdot |A_3 - A_2| = 0 - 1 \cdot |7 - 1| = -6$.

Exemplul 2.

Intrare

```
2 2 5 5
0
-6
1
1 1 -4
1 2 -8
```

Ieșire

```
-5
35
```

Exemplul 3.

Intrare

```
7 8 8 13
0
-3
8
-3
1
-2
-11
8
1 3 -9
4 5 2
3 3 -8
1 7 -5
3 6 2
5 6 -4
6 6 -10
3 7 11
```

Ieșire

```
111
121
161
226
216
236
286
143
```

Punctarea. Testele vor fi grupate in trei categorii, după cum urmează:

A. 30% din punctaj: $N, Q \leq 2000$.

B. 10% din punctaj: $S = T$.

C. 60% din punctaj: Fără restricții adiționale.

Drumuri

Vom defini un drum într-un graf ca o colecție de vârfuri unice $p_1, p_2, ..., p_t$ ($t \geq 2$, $p_i \neq p_j$ pentru $i \neq j$) astfel încât există următoarele muchii în graf $(p_1, p_2), (p_2, p_3),... (p_{t-1},p_t)$. Această listă de vârfuri este ordonată, adică drumul 1, 3, 9 este diferit de drumul 1, 9, 3 sau 9, 3, 1.

Sarcină. Se dă un graf simplu cu N vârfuri și M muchii. Fiecare vârf din graf este colorat în una din cele K culori numerotate de la 1 la K. Scrieți un program care determină numărul posibil de drumuri din graf astfel încât orice două vârfuri în drum au culori diferite.

Date de intrare. Prima linie a intrării standard conține 3 numere întregi separate prin spațiu: N, M, K. Următoarea linie conține K numere întregi între 1 și K, al i-lea număr reprezentând culoare vârfului i. Următoarele M linii conțin câte 2 numere întregi a, b ($1 \leq a, b \leq N, a \neq b$) ce reprezintă muchia (a, b).

Date de ieșire. Ieșirea standard va conține pe o singură linie un număr întreg – numărul de drumuri în care orice două vârfuri au culori diferite.

Restricții. $2 \leq N, M \leq 300000, 1 \leq K \leq 5$. Restricțiile referitoare la timpul de execuție și volumul utilizat de memorie sunt date în descrierea generală a problemelor propuse pentru rezolvare. Fișierul sursă va avea denumirea `drumuri.pas`, `drumuri.c` sau `drumuri.cpp`.

Exemplul 1.

Intrare

```
4 3 3
1 2 1 3
1 2
2 3
4 2
```

Ieșire

```
10
```

Explicație: Putem obține următoarele posibile drumuri: (1, 2), (2, 1), (2,3), (3,2), (2, 4), (4,2), (1, 2, 4), (4, 2, 1), (3, 2, 4), (4, 2, 3), (1, 2, 3), (3, 2, 1)

Drumurile (1, 2, 3) și (3, 2, 1) nu sunt valide deoarece vârfurile 1 și 3 au aceeași culoare. Deci rămân 10 drumuri.

Exemplul 2.

Intrare

```
9 11 4
1 2 3 4 1 2 1 2 2
1 2
1 3
2 3
2 4
3 6
6 2
6 5
4 3
4 5
7 8
9 8
```

Ieşire

```
70
```

Punctarea. Testele vor fi grupate în patru categorii, după cum urmează:

Cotă Punctaj	*N*	*M*	*K*
23%	*≤ 100*	*≤ 100*	*≤ 4*
20%	*≤ 300 000*	*≤ 300 000*	*≤ 3*
27%	*≤ 300 000*	*≤ 300 000*	*≤ 4*
30%	*≤ 100 000*	*≤ 100 000*	*≤ 5*

Tren

Într-un tărâm îndepărtat există o țară cu *N* orașe, care toate se află pe o linie dreaptă și numerotate de la 1 la *N*. Pentru a transporta cetățenii între orașe, regele acestei țări a ordonat construirea unei căi ferate cu câte o stație în fiecare oraș, astfel încât pentru fiecare *i* ($1 \leq i < N$), stațiile *i* și *i*+1 sunt conectate direct. Din cauza bugetului mic al departamentului de transport public, trenurile merg doar într-o singură direcție, astfel ele își încep ruta în stația 1 și călătoresc până în stația *N,* dar nu și invers. La moment în țară există două tipuri de trenuri:

- Trenul local, care se oprește la fiecare stație, și necesită *L* minute pentru a călători între două stații adiacente (de stația *i* la stația *i*+1).
- Trenul expres, care se oprește la doar *M* stații – $S_1, S_2, ..., S_M \left(1 = S_1 < S_2 < ... < S_M = N\right)$, și necesită *E* ($E < L$) minute pentru a călători între două stații adiacente (de la stația *i* la stația *i*+1). Astfel, trenul expres se oprește la prima și ultima stație, și la un anumit set din celelalte stații.

Pentru a facilita transportul mai rapid prin țară, regele a ordonat introducerea unui nou tip de tren, numit trenul regional, care necesită *R* ($E < R < L$) minute pentru a călători între două stații adiacente. Stațiile la care acest tip de tren se va opri încă nu au fost determinate, dar se știe că trenul regional trebuie să se oprească la fiecare stație la care se oprește trenul expres, și că trenul regional trebuie să se oprească la exact *K* stații în total.

Regele nu vrea să călătorească mai mult de *T* minute de la stația 1 la orice altă stație, astfel el cere ca stațiile trenului regional să fie repartizate astfel încât numărul de stații accesibile începând de la prima stație în cel mult *T* minute să fie maximizat. Pentru a calcula timpul necesar pentru a călători între două stații se ia în considerație doar timpul petrecut în mișcare, nu și timpul pentru așteptarea trenurilor.

Sarcină. Scrieți un program care calculează numărul maxim de stații (în afară de stația 1) accesibile de la prima stație în cel mult *T* minute dacă repartizăm opririle trenurilor regionale în mod optim.

Date de intrare. Pe prima linie a intrării se află trei numere întregi *N, M, K.* Pe a doua linie se află trei numere întregi *L, E, R.* Pe a treia linie se află numărul întreg *T*. Următoarele *M* linii conțin câte un număr – a *i*-a linie conține numărul S_i.

Date de ieșire. Ieșirea standard va conține un singur număr – numărul maxim de stații accesibile (în afară de stația 1).

Restricții.

$2 \leq N \leq 10^9$, $2 \leq M \leq K \leq (3000, N)$, $1 \leq E < R < L \leq 10^9$, $1 \leq T \leq 10^{18}$, $1 = S_1 < S_2 < ... < S_M = N$. Restricțiile referitoare la timpul de execuție și volumul utilizat de memorie sunt date în descrierea generală a problemelor propuse pentru rezolvare. Fișierul sursă va avea denumirea `tren.pas`, `tren.c` sau `tren.cpp`.

Exemplul 1.

Intrare

```
10 3 5
6 3 4
26
1
7
10
```

Ieşire

```
8
```

Trenul regional se va opri la 5 staţii, 3 dintre care trebuie să coincidă cu trenul expres: staţiile 1, 7 şi 10. O distribuţie optimă a staţiilor la care se va opri trenul regional este: 1, 6, 7, 9, 10. Putem observa că putem ajunge la primele 9 staţii în cel mult 26 de minute, astfel răspunsul este 8 (staţia 1 este exclusă). Mai exact, putem urma următoarele rute pentru a ajunge la timp:

- Pentru staţiile 1-5, putem merge direct cu trenul local, luând cel mult 24 de minute.
- Pentru staţiile 6-7, putem ajunge direct cu trenul regional de la staţia 1, ajungând în 20 şi 24 minute respectiv.
- Pentru a ajunge la staţia 8 la timp, începând de la staţia 1 călătorim cu trenul expres până la staţia 7, unde ne coborâm şi îmbarcăm trenul local, ajungând la staţia 8 în 24 minute. Observaţi că nu putem folosi trenul regional, deoarece nu am putea coborî la staţia 8.
- Pentru a ajunge la staţia 9 la timp, călătorim cu trenul expres până la staţia 7, unde îmbarcăm trenul regional şi ajungem la destinaţie în 26 minute.
- Cea mai rapidă cale de a ajunge la staţia 10 este să mergem direct cu trenul expres, ceea ce ar dura 30 de minute, deci ar întrece limita.

Exemplul 2.

Intrare

```
10 3 5
6 3 4
23
1
7
10
```

Ieşire

```
7
```

Exemplul 3.

Intrare

```
81 10 13
100000 1000 10000
10000
1
11
21
31
41
51
61
71
81
```

Ieşire

```
3
```

Exemplul 4.

Intrare

```
12 3 4
9 1 2
25
1
10
12
```

Ieşire

```
8
```

Punctarea. Testele vor fi grupate in trei categorii, după cum urmează:

A. 20% din punctaj: $N \leq 300$, $K - M = 2$, $A \leq 10^6$, $T \leq 10^9$.
B. 30% din punctaj: $N \leq 300$.
C. 50% din punctaj: Fără restricții adiționale.