Editorial FMI NO STRESS 13

Contents

1 Comisie

Dificultate: 1/5
Propunători: Dumitru Ilie
Developer: Dumitru Ilie
Editorialist: Dumitru Ilie
First solve: stefdasca după 00:01:18

Problema ne cere să calculăm linia cu sumă maximă și linia cu sumă minimă dintr-o matrice.

2 aiacusumacifrelor

Dificultate: 1/5
Propunători: Matteo-Alexandru Verzotti
Developer: Matteo-Alexandru Verzotti
Editorialist: Matteo-Alexandru Verzotti
First solve: Gheal după 00:03:39

Observație. *Suma cifrelor modulo* 9 *a unui număr este egală cu numărul modulo* 9*.*

În concluzie, putem considera răspunsul la problemă ca fiind numărul de operații necesare pentru a-l transforma pe x în multiplu de 9. Vom lua 9 cazuri diferite:

x	Numărul de operații necesare	Un exemplu de operații
M_9	0	-
$M_9 + 1$	2	$(x + 2) \cdot 3$
$M_9 + 2$	1	$x + 7$
$M_9 + 3$	1	$x \cdot 3$
$M_9 + 4$	1	$x + 5$
$M_9 + 5$	2	$x + 2 + 2$
$M_9 + 6$	1	$x + 3$
$M_9 + 7$	1	$x + 2$
$M_9 + 8$	2	$x + 5 + 5$

3 Chit

Dificultate: 2/5
Propunători: Dumitru Ilie
Developer: Dumitru Ilie
Editorialist: Dumitru Ilie, Giulian Buzatu
First solve: Andrei13 după 00:13:22

În cadrul unui multigraf orientat, dorim să combinăm repetitiv două arce "consecutive" într-unul singur. Dacă cele două arce au aceleași extremități atunci putem elimina cele două arce.

Lemă. *Diferența dintre gradul intern și gradul extern al fiecărui nod rămâne aceeași indiferent de câte operații facem.*

La sfârșit vrem ca gradul intern și extern al fiecărui nod să fie 0, prin urmare vrem ca și la început acestea să fie egale.

Această condiție este necesară, dar trebuie să arătăm că este și suficientă. Pentru asta vom folosi *Principiul inducției matematice complete.*

Cazurile de bază sunt când $M = 0$ și $M = 1$. În prima situație graful nu are arce, așadar se poate reduce. În a doua situație graful are exact o muchie, dar toate nodurile au gradul intern egal cu cel extern. Un astfel de graf nu poate fi redus, dar nici obținut în urma mai multor operații sau primit din input.

Să presupunem că putem reduce orice graf cu cel mult $M - 1$ arce. Un graf cu M arce are două moduri posibile de reducere:

- Reducem două arce cu extremitățile egale dar inversate. Această tranziție ne conduce la un graf cu $M - 2$ arce pe care știm că îl putem reduce datorită inducției;
- Reducem două arce cu o singură extremitate comună într-unul singur. Această tranziție ne conduce la un graf cu $M - 1$ arce pe care știm că îl putem reduce datorită inducției.

Așadar condiția este necesară și suficientă.

4 Invitația la vals

Dificultate: 2/5
Propunători: Mihnea-Valentin Neacșu
Developer: Mihnea-Valentin Neacșu
Editorialist: Mihnea-Valentin Neacșu
First solve: andrei_boaca după 00:13:01

Soluția 1 (40p)

Vom ține un vector caracteristic unde notăm disponibilitatea fetelor. Pentru fiecare băiat de la bal, parcurgem lista cu pozițiile fetelor până dăm de prima fată aflată la o distanță mai mare ca el, și o marcăm ca indisponibilă.

Dacă o astfel de fată nu există, reluăm căutarea de la începutul listei de fete. În acest caz, el își va găsi o parteneră printre cele aflate mai aproape de cancelar decât el, deci e nevoit să treacă prin fața cancelarului, deci incrementăm răspunsul.

Complexitate: $\mathcal{O}(N^2)$.

Observație. *Soluția poate fi optimizată folosind un set în locul vectorului caracteristic. Aceasta duce la o complexitate de* $\mathcal{O}(N \cdot \log N)$*, care obține* 100 *de puncte.*

Soluția 2(100p)

Observație. *Dacă prima persoană este băiat, iar după el există o fată, atunci partenerul de dans al fetei va fi chiar el, deci nu va trece prin fața cancelarului. Invers, dacă e fată, partenerul ei, oricine ar fi el, a trecut prin fața cancelarului.*

Folosind observația de mai sus, formăm următoarea soluție: sortăm cele $2 \cdot N$ poziții într-un singur vector, marcând și sexul persoanei lângă poziție. Apoi parcurgem cele $2 \cdot N$ poziții ale invitaților și reținem un contor, inițial egal cu 0. Dacă întâlnim o fată, adunăm 1 la contor, iar dacă întâlnim un băiat, scădem 1. Cea mai mare valoare atinsă vreodată de contor e răspunsul problemei.

Complexitate finală: $\mathcal{O}(N \cdot \log N)$.

5 Dragonul IPv4

Dificultate: 3/5
Propunători: Ilie-Daniel Apostol
Developer: Stefan-Alexandru Popescu, Ilie-Daniel Apostol
Editorialist: Ilie-Daniel Apostol
First solve: Gheal după 00:29:15

Fie $NRBIT$ numărul de biți necesari pentru a reprezenta în baza 2 ip-urile. În cazul nostru $NRBIT = 32$.

Pentru ca o adresă ip să se afle într-un subnet $a.b.c.d/e$ este nevoie ca primii e biți din reprezentarea în baza 2 a ip-ului să fie la fel ca cei ai subnet-ului.

O solutie $\mathcal{O}(N^2)$ ar fi cea în care iterăm prin toate subnet-urile noastre pentru fiecare operație de tip 3. Subnet-urile pot fi ținute fie într-un vector sau un set STL. Această solutie ar lua 19 puncte.

Atunci când $e \leq 5$, ar exista maxim 2^5 prefixe care ne interesează pentru subnet-uri. Acestea ar putea fi ținute într-un vector de frecvență și s-ar itera prin acesta la fiecare interogare de tip 3. Complexitatea timp finală este $\mathcal{O}(N \cdot 2^e)$. Această soluție ar lua 24 de puncte.

Pentru 100 de puncte, este nevoie să folosim o structură de date numită trie. Pentru o operație de tip 1 vom adăuga doar prefixul de biți de lungime e, respectiv pentru operația de tip 2 vom scoate. Pentru o operație de tip 3, căutam în trie dacă există vreun prefix al reprezentării în baza 2 a ip-ului. Complexitatea finală este $\mathcal{O}(N \cdot NRBIT)$.

Soluție alternativă.

Pentru fiecare lungime de prefix reținem ce prefixuri avem. Vom adăuga maxim un prefix la o operație de tip 1 și la fiecare operație de tip 3 vom itera prin toate lungimile pe care le poate avea masca de rețea (de la 1 la 30) și să vedem dacă pentru această lungime avem un prefix în structura noastră. Soluția se poate implementa folosind un vector de 30 de unordered map-uri sau map-uri din STL. Complexitatea timp finală este $\mathcal{O}(N \cdot NRBIT)$ sau $\mathcal{O}(N \cdot NRBIT \cdot \log N)$.

6 Friendly Numbers

Dificultate: 1/5
Propunători: Marius Stelian Brabete
Developer: Marius Stelian Brabete
Editorialist: Giulian Buzatu
First solve: buzdi după 00:09:53

Probabilitatea de a alege o bilă prietenoasă este $\frac{\#bile\_prietenoase}{N}$. Pentru a vedea dacă o bilă este prietenoasă, este suficient să facem o descompunere în factori primi până la radical. Utilizarea Ciurului lui Eratostene nu este necesară pentru a obține 100 puncte.

7 Celestial

Dificultate: 3/5
Propunători: Marius Stelian Brabete
Developer: Marius Stelian Brabete
Editorialist: Dumitru Ilie
First solve: VSebastian8 după 00:41:29

Pentru a verifica dacă o lanternă acoperă o stea vom folosi următoarea proprietate a produsului scalar a doi vectori:

$$\vec{u} \cdot \vec{v} = |\vec{u}| \cdot |\vec{v}| \cdot \cos(\measuredangle \vec{u}\vec{v})$$

Rescriind egalitatea obținem:

$$\cos(\measuredangle \vec{u}\vec{v}) = \frac{\vec{u} \cdot \vec{v}}{|\vec{u}| \cdot |\vec{v}|}$$

Ne vom folosi de asemenea de faptul că pe intervalul $[0, \pi]$ funcția cosinus este descrescătoare.

Astfel, pentru a verifica dacă o lanternă îndreptată spre punctul P cu unghiul de acțiune ψ acoperă steaua de la punctul Q avem următoarea condiție:

$$\frac{\overrightarrow{OP} \cdot \overrightarrow{OQ}}{\left|\overrightarrow{OP}\right| \cdot \left|\overrightarrow{OQ}\right|} > \cos\psi$$

8 Sushi Cat

Dificultate: 4/5
Propunători: Matteo-Alexandru Verzotti
Developer: Ilie Dumitru, Stefan-Alexandru Popescu
Editorialist: Matteo-Alexandru Verzotti
First solve: ghober după 00:39:20

Pentru a păstra simplitatea formulelor, în editorial vom efectua împărțiri propriu-zise, fără folosirea inversului modular.

Pentru $K = 0$, vom porni un DFS din rădăcina arborelui și vom actualiza fiii după formula dată. Fie $prob_u$ probabilitatea ca Sushi Cat să ajungă în nodul u. Notăm cu sum_w suma greutăților muchiilor adiacente lui u, mai puțin cea către părintele său. Atunci, pentru fiecare fiu v al lui u, calculăm probabilitatea sa ca fiind:

$$prob_v = prob_u \cdot \frac{w_{(u,v)}}{sum_w}$$

unde, cu $w_{(u,v)}$ am notat greutatea muchiei de la u la v.

Observație. *$prob_1$ trebuie setat egal cu 1 la începutul parcurgerii pentru a se putea propaga spre fii. La finalul algoritmului, atât probabilitatea de a ajunge în nodul 1, cât și către orice alt nod care nu este frunză trebuie setată la 0.*

La final, afișăm toate valorile $prob_i$ unde i reprezintă o frunză.

Pentru $K > 0$, putem înlocui vectorul anterior *prob* cu o matrice, unde $prob_{i,j}$ reprezintă probabilitatea ca, pornind cu probabilitate 1 din nodul i, să ajungem în nodul j. Precalcularea acestei matrice se poate face în $\mathcal{O}(N^2)$ rulând algoritmul descris mai sus din fiecare frunză și din rădăcină. Trebuie avut grijă, ca la observația anterioară, să reținem doar valorile ce corespund frunzelor și nodului 1.

Fie $dp_{i,j,k}$ probabilitatea să ajungem din nodul i în nodul j după k repetări. Evident, $dp_{i,j,1} = prob_{i,j}$. Avem următoarea formulă de recurență:

$$dp_{i,j,k} = \sum_{u=1}^{n} dp_{i,u,k-1} \cdot dp_{u,j,k-1}$$

Această recurență ne aduce aminte de algoritmul pentru înmulțirea a două matrice. Putem elimina a treia dimensiune, întrucât la pasul k avem nevoie doar de pasul $k-1$. În concluzie, dacă ridicăm matricea $prob_{i,j}$, calculată anterior, la puterea $K+1$ (jocul inițial $+K$ repetări ale sale), răspunsul nostru se va afla pe linia 1 a matricei. Pentru 100 de puncte, vom folosi ridicarea la putere în timp logaritmic.

Complexitate finală: $\mathcal{O}(N^3 \cdot \log K)$.

9 En passant

Dificultate: 2/5
Propunători: Ilie-Daniel Apostol, Matteo-Alexandru Verzotti
Developer: Matteo-Alexandru Verzotti, Ilie-Daniel Apostol
Editorialist: Ștefan-Alexandru Popescu, Matteo-Alexandru Verzotti
First solve: robert.barbu27 după 00:04:47

Problema este una de simulare. Presupunem că Ernest joacă cu piesele albe (pentru negru se procedează echivalent). Pentru fiecare pion alb de pe rangul al doilea (linia a șaptea a matricei), trebuie să verificăm dacă acesta are cele două căsuțe din fața lui libere și dacă pe rangul al patrulea se află un pion negru adiacent cu acesta.

Spre exemplu, dacă pionul se află pe c2, trebuie să verificăm dacă `c3` și `c4` sunt disponibile și dacă există vreun pion negru pe `b4` sau `d4`.

```
cine_joaca = input().strip()
tabla = [input().strip() for _ in range(8)]

if cine_joaca == 'W':
  tabla.reverse()

cine_sta = chr(ord('W') ^ ord('B') ^ ord(cine_joaca))
cnt = 0
for i in range(8):
  if tabla[1][i] == cine_joaca and tabla[2][i] == '.' and tabla[3][
    i] == '.':
    if (i > 0 and tabla[3][i - 1] == cine_sta) or (i < 7 and tabla
    [3][i + 1] == cine_sta):
      cnt += 1
print(cnt)
```

Listing 1: Cod 100p in Python

10 Ants in the Matrix

Dificultate: 5/5
Propunători: Marius Stelian Brabete
Developer: Vlad Mihai Bogdan, Marius Stelian Brabete
Editorialist: Marius Stelian Brabete
First solve: Gheal după 02:59:18

Un prim aspect care trebuie discutat la această problemă este citirea datelor de intrare, deoarece pot apărea confuzii din cauza elementelor din problemă. Elementele din matrice sunt date în ordinea în care apar în matrice, cum se văd de către observator când se privește planul cartezian cu elementele reprezentate. Primul element din matrice va avea colțul stânga-sus în punctul de coordonate $(0, N)$. De asemenea, colțul dreapta-sus al matricii se va afla în punctul de coordonate (M, N).

Pentru două puncte P și Q, ne interesează căsuțele prin care trece furnica. Pentru a obține asta, putem lua fiecare y, cu $y_P \leq y \leq y_Q$, și să calculăm intersecția acestor y cu dreapta determinată de cele două puncte P și Q. Astfel, având punctele de intersecție cu dreptele $y = \alpha$ și $y = \alpha + 1$, putem determina pentru o linie din matrice, care elemente vor fi traversate de furnică.

O optimizare posibilă este precalcularea sumelor pentru fiecare linie cu ajutorul sumelor parțiale. Pentru două puncte consecutive efectuăm algoritmul descris mai sus, și trebuie eliminată căsuța în care se află punctul P, în cazul în care ea a fost luată în considerare în perechea anterioară de puncte.

11 Mega

Dificultate: 2/5
Propunători: Matteo-Alexandru Verzotti
Developer: Matteo-Alexandru Verzotti
Editorialist: Matteo-Alexandru Verzotti
First solve: DumitrescuEduard după 00:09:56

Vom păstra o structură de date de tip heap pentru evenimente. Aceasta va ține timpii de final ai fiecărui cumpărător.

Inițial, vom adăuga în heap primele N valori t_i. Datorită structurii sale, timpul primului om care pleacă se va afla în nodul 1. Astfel, putem șterge din heap acest timp, fie el t_0, și vom adăuga înapoi în heap timpul de final al următorului om de la coadă, adică $t_0 + t_{N+1}$. Acest procedeu se repetă până când nu mai există oameni la coadă.

Întrucât heap-ul nu ține timpii t_i, ci timpii de final, răspunsul se va afla în rădăcina sa.

Complexitate finală: $\mathcal{O}(M \cdot logN)$.

12 PHF

Dificultate: 4/5
Propunători: Dumitru Ilie
Developer: Dumitru Ilie
Editorialist: Dumitru Ilie, Matteo-Alexandru Verzotti
First solve: andrei_boaca după 01:44:00

Vom nota pentru simplitate șirul nostru în felul următor:
$(\dots(((a_1 \cdot a_2) \cdot a_3) \cdot a_4)\dots) \cdot a_N$
Pentru a înțelege motivația soluției vom inversa șirul, vom scoate operatorii și vom adăuga încă un set de paranteze. Acesta devine:
$a_N(\dots(a_3(a_2(a_1)))\dots)$

Observație. *Putem face această modificare deoarece jocul de PHF este comutativ (nu contează dacă un jucător joacă cu P și altul cu H, sau invers, câștigătorul va fi H).*

Ne putem imagina acum că fiecare element este de fapt o funcție. Dacă facem asta, ne putem imagina cum fiecare funcție modifică rezultatul precedent.

Acum tot ce mai avem de făcut este să modificăm o funcție și să obținem rezultatul final. Datorită proprietății de asociativitate a compunerii funcțiilor putem face asta cu un arbore de intervale. Fiecare interval va conține semnul câștigător pentru fiecare element posibil din domeniu (deoarece sunt doar 3, compunerea este foarte ușoară).

Soluție alternativă.

Structura stereotipică de update/query ne duce cu gândul la folosirea unui arbore de intervale. Presupunem că nodul i menține rezultatul câștigătorului pe intervalul $[l..r]$, unde l joacă inițial cu $l+1$, câștigătorul cu $l+2$, etc.

Fiii acestui nod vor ține rezultatele pentru $[l..mid]$ și $(mid..r]$, unde prin mid am notat jumătatea intervalului $[l..r]$ rotunjită în jos. Acum, este clar că învingătorul aflat în nodul din stânga va juca cu jucătorul aflat pe poziția $mid+1$. Structura definită de noi nu permite aceste operații, întrucât ea presupune că $mid+1$ va juca neapărat cu $mid+2$, fără a lua în considerare că acesta ar putea pierde întâi cu mid.

În concluzie, trebuie modificată structura arborelui de intervale. Vom reține în fiecare nod i, care acoperă un interval $[l..r]$, 3 valori:

$$\begin{cases} a[i].P & = \text{semnul câștigător pe intervalul } [l..r] \text{ dacă întâi } l \text{ joacă contra } P \\ a[i].H & = \text{semnul câștigător pe intervalul } [l..r] \text{ dacă întâi } l \text{ joacă contra } H \\ a[i].F & = \text{semnul câștigător pe intervalul } [l..r] \text{ dacă întâi } l \text{ joacă contra } F \end{cases}$$

Acum, compunerea a două sau mai multe noduri poate fi făcută fără niciun fel de problemă. Răspunsul pentru fiecare interogare se află în rădăcina arborelui.

13 XORCards

Dificultate: 5/5
Propunători: Mihnea-Valentin Neacșu
Developer: Mihnea-Valentin Neacșu
Editorialist: Mihnea-Valentin Neacșu
First solve: ghober după 02:55:07

Observație. *Ross și Rachel nu pot face remiză.*

Să observăm că $0 \oplus 1 \oplus 2 \oplus \cdots \oplus (2^N - 1) = 0$, oricare ar fi $n \geq 2$. Aceste 2^N cărți se împart în 2 submulțimi disjuncte, cărțile lui Ross și cele ale lui Rachel.

Notăm XOR-ul cărților lui Ross cu A și XOR-ul cărților lui Rachel cu B. Deoarece $A \oplus B = 0$, avem $A = B$. Valoarea comună a XOR-urilor este un număr natural între 0 și $2^n - 1$, și, în consecință, apare pe o singură carte, care se află în mâna unuia dintre ei. Prin urmare, acela câștigă.

Cazul $n = 2$.

Dacă Ross a primit cartea 0, câștigă, orice ar fi. El va avea, la final, 2 cărți, 0 și X. $0 \oplus X = X$, deci Ross are în față o carte egală cu XOR-ul tuturor cărților sale, deci e în poziție câștigătoare. Așadar, Ross nu poate pierde, și, conform *observației*, câștigă.

Invers, dacă Ross a primit orice altceva, pierde; Rachel va lua cartea 0 și îl va învinge, din același motiv ca mai sus.

Cazul $n \geq 3$.

În acest caz, vom demonstra că lucrurile stau exact invers: Ross pierde dacă a primit 0 și câștigă în orice alt caz.

Dacă Ross a primit 0, strategia câștigătoare a lui Rachel este după cum urmează: Rachel va lua cartea 1. Apoi, să zicem că Ross a luat cartea C. Rachel va lua $C \oplus 1$. După aceea, la orice carte X luată de Ross, Rachel va replica cu $C \oplus X$. Astfel, cărțile sunt partiționate în 2^{n-1} perechi fiecare cu XOR-ul C. Spunem că o mulțime e bună dacă are exact o carte din fiecare pereche.

Presupunând, fără a restrânge generalitatea, că bitul cel mai semnificativ de pe cartea C e 1, avem că mulțimea cărților de la 0 la 2^{n-1} e, evident, bună. Mai mult, XOR-ul tuturor acestor cărți este 0.

Acum, să observăm că toate cărțile lui Rachel se compune din fix aceste cărți, doar că unele au fost înlocuite cu XOR-ul dintre ele și C, C a fost scos și 1 a fost adăugat. Deci singurele XOR-uri posibile pentru mâna lui Rachel sunt 1 și $C \oplus 1$. Rachel are ambele cărți, deci câștigă orice ar fi.

Invers, când Ross începe cu o carte $non-0$, fie ea A, strategia e următoarea: fie B cartea cu care răspunde Rachel. Ross va lua $A \oplus B$. Apoi împărțim cărțile în perechi (X, Y) astfel încât $X \oplus Y = B$. După aceea, Ross va lua cărți astfel încât să aibă exact una din fiecare pereche (poate face mereu asta).

Din motive similare cu cele de la victoria lui Rachel, XOR-ul mâinii lui Ross poate fi doar A sau $A \oplus B$. Ross are ambele cărți, deci câștigă orice ar fi.

14 yYy

Dificultate: 2/5
Propunători: Mihnea Andreescu
Developer: Mihnea Andreescu
Editorialist: Dumitru Ilie
First solve: toma_ariciu după 00:04:48

Observație. *Fie $[l..r]$ o subsecvență deal. Atunci fiecare $[s..d]$ cu $l \leq s \leq d \leq r$ este de asemenea deal.*

Datorită acestui fapt putem să ignorăm secvențele *deal* nemaximale (cele care rămân *deal* dacă adăugăm elementul următor din stânga sau din dreapta).

Să presupunem că avem o subsecvență *deal* maximală cu K elemente. Numărul de subsecvențe *deal* generate de aceasta este $\frac{K \cdot (K+1)}{2}$. Putem deci calcula acest număr pentru fiecare subsecvență maximală și însuma rezultatele.

Trebuie totuși să avem grijă la capetele secvențelor. Acestea sunt numărate de două ori, atât de secvența maximală care se termină acolo, cât și de cea care începe acolo. Deci trebuie să scădem din rezultat $S - 1$, unde S este numărul de subsecvențe *deal* maximale.

15 Peak

Dificultate: 5/5
Propunători: Matteo-Alexandru Verzotti
Developer: Matteo-Alexandru Verzotti
Editorialist: Matteo-Alexandru Verzotti
First solve: andrei_boaca după 02:23:15

Cheia în a rezolva această problemă este dată de interpretarea geometrică. Putem privi $\frac{a}{b}$, $\frac{x}{y}$, și $\frac{c}{d}$ ca fiind pantele unor funcții. Astfel,

$$f(\lambda) = \frac{a}{b} \cdot \lambda$$
$$g(\lambda) = \frac{x}{y} \cdot \lambda$$
$$h(\lambda) = \frac{c}{d} \cdot \lambda$$

Definind aceste funcții, condiția $\frac{a}{b} \leq \frac{x}{y} \leq \frac{c}{d}$ este echivalentă cu $f \leq g \leq h$. Acum problema noastră se reduce la a calcula numărul de funcții g care se află între f și h.

Observație. *Fie $f'(\lambda) = \frac{x'}{y'} \cdot \lambda$. Atunci această funcție trece prin punctul de coordonate (y', x'). Cum x' și y' sunt întregi, atunci putem spune că funcția este determinată de punctul laticeal (y', x').*

Observație. *O funcție $f'(\lambda) = \frac{x'}{y'} \cdot \lambda$ poate fi determinată de o infinitate de puncte laticeale de forma (ky', kx'), unde $k \in \mathbb{N}^*$.*

În concluzie, dacă găsim un punct laticeal (y, x) "între" graficele funcțiilor f și h, atunci panta dreptei care trece prin acesta va avea valoare $\frac{x}{y}$ și va respecta condiția $\frac{a}{b} \leq \frac{x}{y} \leq \frac{c}{d}$. Răspunsul la problema noastră este deci numărul de astfel de puncte laticeale care îndeplinesc și condiția $s \leq y \leq t$. Să vedem cum ar arăta desenul în plan:

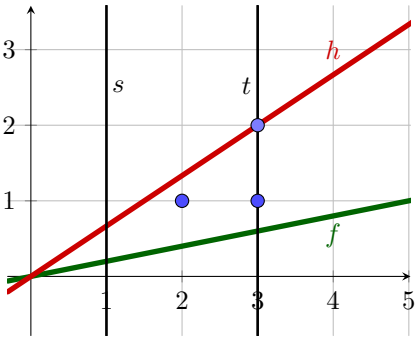

Figure 1: Exemplul din enunț

Putem observa că există doar 3 puncte laticeale care respectă condiția: $(2,1); (3,1)$ și $(3,2)$, care sunt echivalente fracțiilor $\frac{1}{2}, \frac{1}{3}$ și $\frac{2}{3}$. Astfel, problema noastră acum s-a redus la numărarea punctelor laticeale din interiorul unui trapez!

Pentru 45 de puncte, numărarea acestor puncte se putea face foarte ușor. Condiția ca s și t să fie divizibile cu cel mai mic multiplu comun al lui b și d ne garantează că vârfurile trapezului dat vor avea întotdeauna coordonate întregi. Putem aplica Teorema lui Pick (mă întreb oare de la ce vine numele *Peak*), ce studiază relația dintre punctele laticeale din interiorul unui poligon și aria respectivului poligon.

Ne rămâne doar să studiem numărul de puncte de pe marginile poligonului. Îl vom nota $ABCD$, unde $\{A, B\} \in G_f$ și $\{C, D\} \in G_h$.

- Numărul de puncte de pe $[AD]$ este $y_D - y_A + 1$;
- Numărul de puncte de pe $[BC]$ este $y_C - y_B + 1$;
- Numărul de puncte de pe $[AB]$ este egal cu numărul de puncte de pe $[OB]$ minus numărul de puncte de pe $[OA)$. Panta dreptei este $\frac{a}{b}$, deci vor exista puncte întregi din $\frac{b}{(a,b)}$ în $\frac{b}{(a,b)}$.

Mare grijă la numărarea capetelor. Acestea trebuie scăzute o dată, fiind numărate mai sus de două ori.

Complexitatea pentru subtaskul 2 este $\mathcal{O}(Q \cdot \log N)$.

Cum ducem problema mai departe?

Teorema lui Pick este foarte folositoare în cazul în care poligonul are vârfuri întregi, dar în problema noastră această condiție nu este neapărat adevărată. Se pare însă că există un algoritm care rezolvă și pentru poligoane arbitrare. Acesta este descris în detaliu pe CP-Algorithms, așa că acest editorial va prezenta doar o idee generală a acestuia.

Observație. *Numărul de puncte care aparțin poligonului $ABCD$ este de fapt numărul de puncte aflate sub (sau pe) segmentul $[DC]$ minus numărul de puncte aflate strict sub segmentul $[AB]$.*

Observație. *Fie $P(x_P, y_P), Q(x_Q, y_Q)$ capetele unui segment aflat pe graficul unei funcții f. Numărul de puncte laticeale de sub $[PQ]$ este egal cu numărul de puncte laticeale de sub segmentul $[P'Q']$, unde $P'(\lceil x_P \rceil, f(\lceil x_P \rceil))$, $Q'(\lfloor x_Q \rfloor, f(\lfloor x_Q \rfloor))$.*

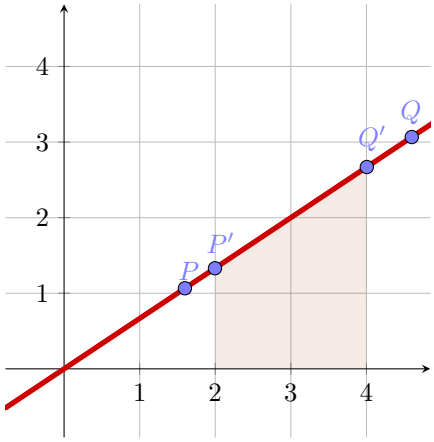

Figure 2: Reprezentare grafică a observației

În concluzie, putem forța coordonatele x ale segmentului să fie întregi. După aceea, vom aplica o translatare a coordonatelor, de la $[x'_P, x'_Q]$ la $[0, n]$, unde $n = x'_Q - x'_P$. Fie k panta segmentului nostru.

La fiecare din următorii pași vom adăuga la rezultat $\lfloor y'_P \rfloor \cdot n$ (punctele laticeale aflate în dreptunghiul de "sub" segmentul nostru). După aceea îl vom translata și pe acesta în jos, astfel încât $y'_P < 1$. Avem acum două cazuri:

1. $k \geq 1$.

   În acest caz, putem trasa o dreaptă cu pantă $\lfloor k \rfloor$ și să adunăm toate punctele laticeale aflate sub aceasta. Acestea sunt în număr de

   $$\sum_{x=0}^{n-1} x \cdot \lfloor k \rfloor = \lfloor k \rfloor \cdot \sum_{x=0}^{n-1} x = \lfloor k \rfloor \cdot \frac{n(n-1)}{2}$$

   Ne-am redus astfel problema de la $k \geq 1$ la $k' = k - \lfloor k \rfloor < 1$.

2. $k < 1$.

   Dacă $y'_Q < 1$, atunci nu există niciun punct laticeal rămas de numărat. Altfel, ne putem schimba sistemul de referință, rotind planul cu 180° și oglindindu-l față de axa Ox. Noul sistem de referință ne arată că acum am ajuns înapoi la cazul 1. Putem continua astfel recursiv.

O demonstrație riguroasă cu privire la corectitudinea algoritmului și complexitatea sa se găsește la linkul de mai sus, alături de ilustrații cu privire la sistemele de referință.

De notat faptul că, în mod normal, fiind o problemă de numărare, orice eroare de precizie cauzată de lucrul cu numere reale poate conduce la un răspuns incorect. O soluție corectă folosește cu grijă numerele reale (posibil printr-o clasă de fracții), însă adaugă un factor de log în plus. Cu toate acestea, pentru testele generate la problema dată, lucrul cu tipul de date *double* nu ar fi trebuit să schimbe rezultatul. Complexitate finală: $\mathcal{O}(Q \cdot \log N)$ sau $\mathcal{O}(Q \cdot \log^2 N)$.

Comisia:

- Dumitru Ilie
- Giulian Buzatu
- Ilie-Daniel Apostol
- Marius-Stelian Brabete
- Matteo-Alexandru Verzotti
- Mihnea Andreescu
- Mihnea-Valentin Neacșu
- Ștefan-Alexandru Popescu
- Vlad-Mihai Bogdan