RoAlgo Contest 3 - Editorial

Rareș-Andrei Buzdugan, Ștefan-Cosmin Dăscălescu, Alexandru Lensu, Cosmin-Gabriel Pascale

22 Iulie 2023

1 Mulțumiri

Acest concurs nu ar fi putut avea loc fără următoarele persoane:

- Rareș-Andrei Buzdugan, Alexandru Lensu, Cosmin-Gabriel Pascale, autorii problemelor și laureați la concursurile de informatică și membri activi ai comunității RoAlgo.
- Alex Vasiluță, fondatorul și dezvoltatorul principal al Kilonova
- Andrei Paul Iorgulescu, David-Ioan Curcă, Andrei Chertes, Vlad Tutunaru, testerii concursului, care au dat numeroase sugestii și sfaturi utile pentru buna desfășurare a rundei, precum și ajutor la crearea unor teste.
- Ștefan-Cosmin Dăscălescu, coordonatorul rundei.
- Comunității RoAlgo, pentru participarea la acest concurs.

2 Problema Robotul gospodar

Autor: Lensu Alexandru

2.1 Soluție de 20 de puncte

Vom crea un vector "v" cu N poziții care inițial va fi umplut cu valoarea 0. Pentru fiecare set de date vom parcurge toate capacitățile plantelor din stânga plantei pe care se află robotul(din stânga lui X[i]) până la întâlnirea unei capacități mai mici decât capacitatea plantei pe care se află robotul(X[i]) și analog în dreapta până la întâlnirea unei capacități mai mari și vom reține pozițiile acestor plante în doi indici (st, respectiv dr). La fiecare set de date, observăm că T[i] este defapt numărul de picături care va fi adăugat și facem

un for de la st la dr și vom adăuga T[i] în vectorul creat. Creăm o variabilă în care vom memora cantitatea de picături risipită și un vector în care vom adauga cantitățile necesare pentru a ajunge la c[i], numit lipsa. La sfârșitul tuturor operațiilor vom parcurge vectorul creat și dacă $c[i] \leq v[i]$ atunci vom adaugă la risipite $v[i]$–$c[i]$, și lipsa[i] va fi egal cu 0, iar dacă $v[i] < c[i]$ lipsa[i] va fi egal cu $c[i]$–$v[i]$. Se observă foarte ușor că optim este să udăm plantele cu picăturile lipsa cele mai mici deci sortam vectorul lipsa. Parcurgem vectorul lipsa și cât timp $risipite$–$lipsa[i] >= 0$ incrementam contorul și scădem numărul de picături risipite cu lipsa[i]. La sfârșit vom afișa contorul.

Complexitate $O(K * N^2 + N * logN)$ Sursa de la această soluție poate fi accesată aici.

2.2 Soluția oficială

Vom crea trei vectori cu N poziții care inițial vor fi umpluți cu valoarea 0. Unul dintre aceștia se va numi st (unde st[i] reprezintă poziția plantei cea mai apropiată care are capacitatea mai mică decât capacitatea plantei curente) Celălalt se va numi dr (unde dr[i] reprezintă poziția plantei cea mai apropiată care are capacitatea mai mare decât capacitatea plantei curente) Și ultimul se va numi adăugate (este folosit în același scop că și "v" de la soluția 1) Pentru construirea vectorilor st și dr se folosește metodă skyline cu ajutorul unei stive. Pentru fiecare query se observă că adăugăm T[i] picături la toate plantele din intervalul [st[i],dr[i]], lucru pe care îl putem face optim cu șmenul lui mars. La sfârșitul tuturor operațiilor vom parcurge vectorul creat și dacă $c[i] \leq adăugate[i]$ atunci vom adaugă la risipite $adăugăm[i]$–$c[i]$, și lipsa[i] va fi egal cu 0, iar dacă $adăugate[i] < c[i]$ lipsa[i] va fi egal cu $c[i]$–$adăugăm[i]$. Se observă foarte ușor că optim este să udăm plantele cu picăturile lipsa cele mai mici deci sortam vectorul lipsa. Parcurgem vectorul lipsa și cât timp $0 \leq risipite$–$lipsa[i]$ incrementam contorul și scădem numărul de picături risipite. La sfârșit vom afișa contorul.
Soluție de 40 de puncte(Șmenul lui Mars): aici.
Soluție de 40 de puncte(Stivă): aici.
Soluție de 100 de puncte: aici.

3 Problema Expresie

Autor: Lensu Alexandru

3.1 Solutie de 30 de puncte

Observăm că dacă $4 \leq x \leq 100$ putem foarte ușor să parcurgem toate valorile lui x și să calculăm E(x). De asemenea putem să simplificăm expresia

prin scăderea din K a lui 10 și înmulțirea lui K cu 1000 la
$E(x) = x^4 - 3 * x^3 - 2 * x^2 + 5$. Pentru fieccare x unde se respectă inecuatia $E(x) \leq K$ creștem contorul pe care îl vom afișa. Sursa de la această soluție poate fi accesată aici.

Complexitate O(Q * xMAX).

3.2 Soluția oficială

Se observă că pentru oricare $4 \leq x$ atunci când x crește și E(x) crește, altfel spus $f(x) = \frac{x^4-3x^3-2x^2+5}{1000} + 10$ este strict crescătoare pentru $4 \leq x$. Datorită acestui fapt este posibil să căutăm binar cea mai mare valoare a lui x pentru care $E(x) \leq K$, iar răspunsul nostru va fi (x – 4) + 1.

Complexitate O(Q * log xMAX).

Sursa de la această soluție poate fi accesată aici.

4 Problema Opt

Autor: Buzdugan Rareș-Andrei

4.1 Subtask 1 - 17 puncte

Dacă $K = 0$, atunci nu vom putea să aplicăm operația de Upgrade pe niciun punct, deci va trebui doar să aflăm aria totală.
Observăm că un poligon consecutiv format din punctele (X_i, Y_i) și (X_{i+1}, Y_{i+1}) poate avea forma unui trapez dreptunghic cu bazele Y_i și Y_{i+1} și înălțimea $X_{i+1} - X_i$. Așadar, aria unui poligon consecutiv este egală cu $\frac{(Y_i+Y_{i+1})*(X_{i+1}-X_i)}{2}$
Se poate observa că această formulă funcționează și când poligonul este un triunghi sau un segment. Pentru a afla aria totală, vom însuma aceste arii ale poligoanelor consecutive.
Complexitatea temporală - $O(N)$

4.2 Subtask 2 - 22 puncte

Observăm că atunci când incrementăm coordonata Y a unui punct i, acesta va modifica aria poligonului consecutiv format din punctele $(i-1, i)$ și punctele $(i, i+1)$. În cazul în care $i = 1$ sau $i = N$, operația va modifica aria unui singur poligon consecutiv.
Astfel, incercăm să găsim care este punctul care crește aria totală cu cea mai mare valoare. Vom lua fiecare punct pe care putem aplica operația de Upgrade și vom calcula aria totală maximă obținută după aplicarea operației. După

care, vom aplica operația de Upgrade pe punctul unde am obținut aria totală maximă. Repetăm acest proces cât timp mai putem aplica operația pe vreun punct sau nu am aplicat K operații în total.
Complexitatea temporală - $O(N * K)$
Sursa de la această soluție poate fi accesată aici.

4.3 Subtask 3 - 61 puncte

Încercăm sa determinăm cu cât va crește aria poligonului consecutiv format din punctele $(i, i+1)$ dacă aplicăm operația de Upgrade pe punctul i. Acest lucru se poate afla calculând diferența dintre :

- $\frac{(Y_i+Y_{i+1}+1)*(X_{i+1}-X_i)}{2}$ = Aria poligonului consecutiv după aplicarea operației.
- $\frac{(Y_i+Y_{i+1})*(X_{i+1}-X_i)}{2}$ = Aria poligonului consecutiv inițial.

După efectuarea calculelor, obținem că diferența este egală cu $\frac{(X_{i+1}-X_i)}{2}$, ceea ce inseamnă că aria va crește **mereu** cu această valoare si creșterea ariei **nu depinde de Y.**
Astfel, putem aplica o strategie *greedy* și să sortăm punctele descrescător după valoarea de creștere a ariei dacă se aplică operația de Upgrade. Fie $Crestere_i$ valoarea de creștere a ariei dacă se aplică operația pe punctul i.
În timp ce parcurgem punctele sortate, atunci când ne aflăm la punctul i, încercăm să aplicăm operația de Upgrade pe acest punct de un număr maxim de ori posibil. Astfel, avem două situații :

- $Operatii\_Efectuate + B_i \leq K$. Putem aplica operația de B_i ori și aria va crește cu $B_i * Crestere_i$
- $Operatii\_Efectuate + B_i > K$. Putem aplica operația de $(K - Operatii\_Efectuate)$ ori și aria va crește cu $(K - Operatii\_Efectuate) * Crestere_i$

Repetăm acest proces pentru toate punctele sau până când întâlnim a doua situație.
Se vor lucra cu atenție operațiile ce pot duce la erori de precizie!
Complexitatea temporală - $O(NlogN)$
Sursa de la această soluție poate fi accesată aici.

5 Problema Legat

Autor: Buzdugan Rareș-Andrei

5.1 Subtask 1 - 21 puncte

Pentru a calcula probabilitatea, trebuie să determinăm numărul de cazuri favorabile și numărul de cazuri posibile, după care, vom folosi inversul modular pentru a determina rezultatul modulo 1 000 000 007
Se observă că palatul poate fi reprezentat ca un graf orientat aciclic. Formal, trebuie sa determinăm numărul de drumuri formate din maxim L muchii. Acestea vor reprezenta cazurile posibile, iar cazurile favorabile vor fi drumurile formate din maxim L muchii care se termină în nodul N.
Putem lua fiecare nod ca fiind nodul de start al drumului, după care, vom folosi parcurgerea DFS, parcurgând maxim L muchii. De fiecare dată când intrăm într-un nod, am obținut un caz posibil. Dacă nodul în care ne aflăm este nodul N, atunci am obținut un caz favorabil.
Sursa de la această soluție poate fi accesată aici

5.2 Subtask 2 - 79 puncte

Putem folosi metoda programării dinamice :
$dp_{i,j}$ = Numărul de drumuri care se termină în nodul i și sunt formate din j muchii.
Vom folosi următoarea relație de recurență :

- $dp_{i,0} = 1$ (Există un singur drum cu 0 muchii care începe într-un nod i)
- $dp_{i\_v,j+1} += dp_{i,j}$, unde i_v este un nod astfel încât există arcul (i, i_v) (Putem pune nodul i_v în toate drumurile care se termină în i și au j muchii, obținându-se drumuri cu $j + 1$ muchii.)

Pentru a ne asigura că, atunci când suntem la nodul i, am terminat de calculat toate drumurile care se termină in nodul i, putem profita de faptul că graful este orientat și aciclic și să sortăm topologic nodurile, parcurgându-le în această ordine.
Numărul de cazuri posibile va fi egal cu suma tuturor valorilor $dp_{i,j}$, cu $1 \leq i \leq N$ și $0 \leq j \leq L$.
Numărul de cazuri favorabile va fi egal cu suma tuturor valorilor $dp_{N,j}$, cu $0 \leq j \leq L$.
Se calculează probabilitatea ca la Subtask 1.
Complexitatea temporală - $O((N + M) * L)$
Sursa de la această soluție poate fi accesată aici.

6 Problema szceas

Autor: Cosmin-Gabriel Pascale

6.1 Subtask 1 - 20 de puncte

Iterăm șirul cu doi indici, i și j. Pentru fiecare pereche (i, j) vom verifica dacă atunci când interschimbăm elementele a[i] și a[j] între ele, șirul devine sortat.
Complexitate temporală: $O(T * N^3)$.
Sursa de la această soluție poate fi accesată aici.

6.2 Subtask 2 - 20 de puncte

Declarăm o matrice sortat[N][N], în care sortat[i][j], reține 1 dacă secvența i...j este sortată, și 0 în caz contrar. În continuare, pentru fiecare pereche de indici i și j, verificăm dacă secvențele [1, i - 1], [i + 1, j - 1], [j + 1, N] sunt sortate și dacă $a[i-1] < a[j] < a[i+1]$ și $a[j-1] < a[i] < a[j+1]$.
Complexitate temporală: $O(T * N^2)$.
Sursa de la această soluție poate fi accesată aici.

6.3 Subtask 3 - 60 de puncte

Sortăm crescător șirul dat și verificăm, pentru fiecare i ∈ 1, 2, ..., n, dacă a[i] este diferit față de b[i], unde B este șirul sortat. În caz afirmativ, creștem un contor, care inițial este 0. Dacă la final, contorul este exact 2, atunci șirul dat este aproape sortat.
Complexitate temporală: $O(T * N * log(N))$.
Sursa de la această soluție poate fi accesată aici.

7 Problema Intervale

Autor: Cosmin-Gabriel Pascale
Observația principală: În cazul în care cunoaștem frecvența fiecărei litere din secvența [stânga, dreapta], putem determina dacă literele din secvență se pot rearanja astfel încât să formeze un cuvânt palindromic. Pentru fiecare literă ce apare în secvență, vom verifica frecvența ei, mai departe având două cazuri:
I. frecvența ei e număr par, deci putem continua.
II. frecvența ei e număr impar. Se deduce faptul că un cuvânt palindromic nu poate avea decât maxim o literă de frecvență impară, care va(vor) fi întotdeauna în mijlocul cuvântului. Astfel că, dacă frecvența a cel puțin două litere ce apar în secvența [stânga, dreapta] este impară, atunci cuvântul nu mai poate fi palindromic.

7.1 Subtask 1 - 31 de puncte

Pentru fiecare test, iterăm de la stânga la dreapta și ținem minte într-un vector de frecvență de 26 de poziții de câte ori apare fiecare literă. La final, ne vom folosi de observația prezentată anterior.
Complexitatea temporală: $O(|S| * T)$
Sursa de la această soluție poate fi accesată aici.

7.2 Subtask 2 - 47 de puncte

Vom declara o matrice sumePartiale[26] [lungimea șirului], în care vom aplica tehnica sumelor parțiale a unui vector, doar că la fiecare literă a șirului S vom repeta această acțiune de 26 de ori, pentru fiecare literă posibilă. Astfel, putem afla în O(1) frecvența fiecărei litere ce apare în secvența [stânga, dreapta]. La final, ne vom folosi de observația prezentată anterior.
Complexitatea temporală: $O(|S| * SIGMA + T * SIGMA)$, unde SIGMA reprezintă lungimea alfabetului care cuprinde literele din șirul de caractere dat.
Sursa de la această soluție poate fi accesată aici.

7.3 Subtask 3 - 22 de puncte

Ne vom folosi în continuare de sume parțiale, dar de această dată vor fi aplicate pe biți. Într-un vector A de N elemente, elementul de pe poziția i (i ∈ 1, 2, ..., N) reține, în biții 0...25, 0 dacă frecvența din secvența [1, i] a literei respective este pară, iar 1 în caz contrar. Astfel, putem folosi un vector de int-uri, întrucât valoarea unui astfel de număr este cel mult 2^{26}. În continuare, la fiecare pas, dar și la fiecare interogare, ne vom folosi de operația pe biți XOR. Astfel, atunci când facem a[dreapta] XOR a[stânga - 1], trebuie să obții un număr care fie este 0, fie are un singur bit de 1, astfel încât răspunsul întrebării respective să fie DA. Această soluție îmbunătățește semnificativ atât timpul de execuție, cât și memoria utilizată.
Complexitatea temporală: $O(|S| + T)$
Sursa de la această soluție poate fi accesată aici.