Concursul Județean de Informatică „Future for Future”
Ediția a II-a, 2025
Clasele XI-XII
Descrierea soluțiilor

1. Problema BTC

Propusă de: Alexandru Thury-Burileanu, CNMB

Subtask 1 (10 puncte)

Parola este X, $X+1$, $X+2$, ..., Y.

Subtask 2 (10 puncte)

Cunoaștem primul și ultimul termen al progresiei aritmetice, așadar putem calcula rația: $r = \frac{Y-X}{N-1}$. Parola este X, $X+r$, $X+2r$, ..., Y.

Subtask 3 (20 puncte)

Dacă $X = 1$, cum toate elementele parolei sunt cel puțin 1, primul element va fi mereu 1. Rămâne să găsim un șir care să îl conțină pe Y astfel încât ultimul element să fie minim.

Rația progresiei aritmetice divide diferența dintre doi termeni ai acesteia. Trebuie să alegem r minim astfel încât $r \mid Y-1$ și ultimul termen să fie cel puțin Y, adică $1+(N-1)r \geq Y$. Parola va fi 1, $1+r$, $1+2r$, ..., $1+(N-1)r$.

Exemplu de implementare: https://kilonova.ro/pastes/iVVMW7yaEILA

Subtaskurile 4, 5, 6 (câte 20 puncte)

Similar, $r \mid Y-X$. Definim o procedură de verificare pentru o anumită valoare a rației r, care va determina dacă există o progresie aritmetică validă de rație r și o va construi pe cea care minimizează ultimul element, M. Evident, progresia este bine definită cunoscând doar r și M: $a_i = M-(N-i)r$.

- Mai întâi, calculăm numărul de elemente dintre X și Y (inclusiv X și Y), $A = \frac{Y-X}{r}+1$. Dacă $A > N$, nu se poate construi o progresie aritmetică validă de rație r. Dacă $A = N$, atunci $M = Y$. Altfel, fie $N' = N-A > 0$.
- Din cele N' elemente rămase preferăm să adăugăm mai întâi numere înainte de X, pentru a minimiza ultimul element: $X-r$, $X-2r$, ..., cât timp $X-kr \geq 1$; cu alte cuvinte, de $B = \frac{X-r-1}{r}+1$ ori. Dacă $N' \leq B$, atunci $M = Y$. Altfel, fie $N'' = N'-B > 0$.
- Mai rămâne de adăugat N'' elemente după Y: $Y+r$, $Y+2r$, ..., astfel încât elementul maxim va fi $M = Y+N''r$.

Această procedură de verificare trebuie aplicată pentru toate rațiile r care divid $Y-X$, iar la final vom alege rația pentru care se obține M minim.

Pentru subtaskul 4, $Y-X$ este număr prim, deci este suficient să verificăm doar $r = 1$ și $r = Y-X$.

Pentru subtaskul 5, putem verifica fiecare număr d de la 1 la $Y-X$ dacă $d \mid Y-X$, caz în care aplicăm procedura de verificare pentru d.

Pentru subtaskul 6, parcurgem numerele doar până la $\sqrt{Y-X}$, iar pentru fiecare divizor d găsit aplicăm procedura de verificare pentru d și $\frac{Y-X}{d}$.

Complexitate timp: $\mathcal{O}(N)$ pentru subtaskul 4, $\mathcal{O}(N+Y-X)$ pentru subtaskul 5, $\mathcal{O}(N+\sqrt{Y-X})$ pentru subtaskul 6.

Exemplu de implementare: https://kilonova.ro/pastes/Wo24lYGWhKDW

Soluție alternativă

Rația optimă este r minim astfel încât $r \mid Y - X$ și $X + (N-1)r \geq Y$. Mai departe, aflăm elementul maxim și reconstituim parola similar cu soluția de mai sus.

Exemplu de implementare: https://kilonova.ro/pastes/qpKTSfW4Ogpz

2. Problema Future

Propusă de: Andrei-Rareș Tănăsescu, CNMB

Înainte de toate, ne este folositor să sortăm shurikenele după distanța de la origine. Pentru aceasta, se poate folosi un algoritm de sortare cu o complexitate maximă de $\mathcal{O}(N^2)$.

Subtask 1 (7 puncte)

Deoarece avem doar subzone cu punctaje negative, ne dorim ca toate shurikenele să fie în exteriorul zonei de punctare.

Putem încerca, astfel, să plasăm zona de punctare între origine și primul shuriken sau între oricare două shurikene consecutive, preferând poziția de start cea mai apropiată de origine. Dacă distanțele între shurikenele consecutive sunt prea mici și nu permit plasarea zonei de punctare în intervalele menționate, atunci zona va începe la prima unitate după ultimul shuriken.

Complexitate timp: $\mathcal{O}(N + M)$.

Subtask 2 (11 puncte)

Având un singur shuriken, ne dorim ca acesta să fie cuprins într-o subzonă de punctaj cât mai mare. Putem testa pentru fiecare subzonă dacă poate cuprinde shurikenul respectiv fără a muta punctul de start al primei subzone înainte de origine, ținând o variabilă de tip maxim pe care o vom afișa la sfârșit.

Notă: Există un caz particular, când toate subzonele ce pot cuprinde shurikenul au punctaj negativ și, deci, ne dorim ca punctul să fie în afara zonei de punctare.

Complexitate timp: $\mathcal{O}(M)$.

Subtask 3 (13 puncte)

Dacă subzona pe care o avem la dispoziție are punctaj negativ, aplicăm algoritmul de la subtaskul 1. Dacă are punctaj pozitiv, atunci putem aplica mai mulți algoritmi pentru a încerca să cuprindem cât mai mule shurikene în aceasta.

Ne putem folosi de metoda *two pointers*, menținând un interval de shurikene pe care putem să le includem în subzonă și încercând de fiecare dată să mai adăugăm un shuriken la dreapta intervalului. Dacă lungimea intervalului curent devine mai mare decât lungimea subzonei, atunci scoatem pe rând punctele din stânga până când distanța intervalului curent devine mai mică decât lungimea subzonei.

Înregistrăm numărul maxim de shurikene ce pot intra în subzonă și poziția minimă a subzonei pentru care se respectă această condiție.

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

Subtask 4 (39 puncte)

Putem încerca să plasăm începutul zonei de punctare la orice distanță de origine cuprinsă între 0 și poziția celui mai îndepărtat shuriken + 1. Determinăm, pentru fiecare poziție de start a zonei de punctare, în ce subzonă se află fiecare shuriken, folosind un algoritm cu complexitatea $\mathcal{O}(N + M)$ sau $\mathcal{O}(N \log(N + M))$, însumând apoi punctajele subzonelor. Afișăm prima poziție de start ce ne oferă punctajul cel mai mare.

Complexitate timp: $\mathcal{O}(N(N + M))$ sau $\mathcal{O}(N^2 \log(N + M))$.

Soluția finală

Folosim un procedeu asemănător cu *sliding window*, glisând poziția de start a zonei de punctare de la 0 la poziția celui mai îndepărtat shuriken + 1. Creăm $N \cdot M$ evenimente de tipul: *„shurikenul i va trece*

din subzona $j+1$ în subzona j în momentul în care distanța de la origine la punctul de start al zonei de punctare devine x”, pentru $i = \overline{1, N}$ și $j = \overline{1, M}$.

În plus, construim și un vector de tranziție între subzone, în care memorăm diferența de punctaj pe care o dă trecerea unui shuriken de la o subzonă $i+1$ la o subzonă i. Sortăm evenimentele după distanța x de la origine și le parcurgem aplicând schimbările de punctaj și ținând cont de prima poziție pozitivă pentru care înregistrăm un punctaj maxim.

Complexitate timp: $\mathcal{O}(NM \log(NM))$.

Exemplu de implementare: https://kilonova.ro/pastes/kMJarjiH1iC5

3. Problema Măsură

Propusă de: Ștefan Patrichi, CNMB

Pentru fiecare regulă dată de tipul a `u1 u2`, creăm o muchie de cost a de la `u2` la `u1` și una de cost $\frac{1}{a}$ de la `u1` la `u2`.

Subtaskurile 1 și 3 (42 + 18 puncte)

Forță brută: pentru fiecare întrebare parcurgem graful cu DFS, înmulțind costurile muchiilor. Eventual, menținem nodurile într-o structură DSU, pentru a verifica rapid conexitatea.

Complexitate timp: $\mathcal{O}(n)$ per întrebare per test.

Exemplu de implementare: https://kilonova.ro/pastes/NMS2iDrh35T6

Subtaskul 2 (28 puncte)

După ce am citit toate regulile de transformare, îi putem atribui fiecărei unități un coeficient de transformare într-o unitate arbitrară (pe care o numim *reprezentantă*) din componenta conexă în care se află, parcurgând graful cu DFS (o singură dată!).

Așadar, pentru a transforma din unitatea 1 în unitatea 2, presupunând că se află în aceeași componentă conexă, transformăm mai întâi din unitatea 1 în unitatea reprezentantă, înmulțind cu coeficientul unității 1, apoi din unitatea reprezentantă în unitatea 2, înmulțind cu inversul coeficientului unității 2. Astfel, obținem complexitate constantă pe întrebări.

Complexitate timp: $\mathcal{O}(n)$ per test.

Exemplu de implementare: https://kilonova.ro/pastes/XeF7gzOOtwOT

Subtaskul 4 (12 puncte) – Soluția finală

Menținem nodurile într-o structură DSU, în care, pe lângă nodul reprezentant, ținem minte pentru fiecare nod și coeficientul lui. În continuare, procedăm similar ca la subtaskul 2.

Complexitate timp: $\mathcal{O}(\alpha(n)) \approx \mathcal{O}(1)$ per test (plus complexitatea citirii și a asocierii unui număr pentru fiecare unitate – vezi mai jos).

Exemplu de implementare: https://kilonova.ro/pastes/e9nGk5HKJ41h

Detalii de implementare

Corespondența dintre numele unei unități și numărul ei de identificare se poate realiza cu un container STL (`map<string, int>`) sau considerând unitatea drept un număr în baza 27 (de ce nu funcționează baza 26?), similar cu un hash polinomial.

Nu uităm nici de posibilitatea de a transforma o unitate în ea însăși, chiar dacă nu a fost definită încă!