RAUCODER 2024

PROBLEME

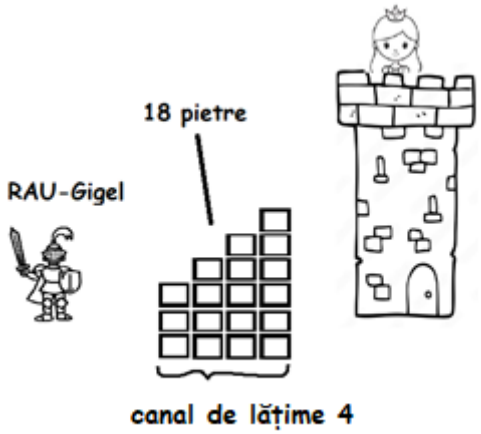
Data: **25 mai 2024**

Concursul acesta este despre RAUGigel. RAUGigel este un elev pasionat de algoritmică și programare, dar și de matematică. El vrea astăzi să vă prezinte câteva dintre pasiunile lui și vă propune trei probleme, toate având același punctaj. Și, pentru că suntem într-un an al alegerilor, veți vedea că și RAU-Gigel trebuie să ia câteva decizii...

Problema 1. Donjon

Căpcăunul cel rău o ține captivă pe frumoasa prințesă într-un castel izolat, într-un turn înalt. RAU-Gigel prinde de veste și se duce într-un suflet să o salveze. El trebuie să treacă un canal de apărare de lungime *K*, așezând *N* pietre de dimensiuni egale cu unitatea, în formă de trepte de înălțimi *h, h+1, h+2, h+3, ... h+K-1*. Prima treaptă trebuie să aibă cel puțin o piatră, iar dacă o singură piatră rămâne nedistribuită, toată construcția se face una cu pământul.

Dacă se cunosc: numărul *N* de pietre magice și distanța *K* dintre poziția lui RAU-Gigel și donjonul prințesei, putem să aflăm dacă RAU-Gigel va reuși să construiască treptele?



RAU-Gigel își încearcă șansele. Un eșec va însemna renunțarea la prințesă. O reușită însă, va conduce la o surpriză din partea căpcăunului, acesta îl va supune pe RAU-Gigel la o nouă provocare:

RAU-Gigel, flăcăul meu,
Te voi provoca la greu!
De testu'-l vei câștiga,
Pe prințesa vei avea!

Și îi propune următorul test: „*vom alege, pe rând, câte o treaptă dintre cele construite de tine și vom dărâma din ea oricâte pietre, minim una, cel mult toate. Prea-frumoasa prințesă va rămâne cu acela dintre noi, care va dărâma și ultima piatră*". RAU-Gigel va începe primul, iar atât RAU-Gigel cât și căpcăunul vor juca responsabil, adică vor face întotdeauna alegerea cea mai bună în interesul propriu. Reușește RAU-Gigel să elibereze prințesa?

De exemplu, daca avem N = 5 și K = 2, RAU-Gigel construiește 2 trepte de înălțimi 2 și 3, astfel nu rămân pietre nefolosite. Să vedem acum cine va câștiga cea de-a doua provocare. RAU-Gigel va alege primul. El poate alege prima treaptă și poate dărâma toate pietrele de pe ea, însă aceasta va fi o alegere greșita, întrucât căpcăunul va dărâma la rândul lui toate pietrele de pe treapta a doua, păstrând prințesa captivă. Așadar, RAU-Gigel nu va începe prin a dărâma toate pietrele de pe o treaptă.

O altă variantă ar fi ca RAU-Gigel să aleagă prima treaptă, de unde să dărâme o singura piatră. Aceasta iarăși ar fi o mutare greșită, căpcăunul ar dărâma la rândul său două pietre de pe treapta a doua, iar acum, orice alegere ar face RAU-Gigel, ar pierde.

Cu siguranță, RAU-Gigel va alege treapta a doua de unde va dărâma o singură piatră. Căpcăunul nu va avea ieșire din aceasta situație: dacă dărâmă ambele pietre de pe oricare treaptă, RAU-Gigel le va dărâma pe celelalte două și va câștiga, dacă căpcăunul dărâmă numai o piatră de pe oricare treaptă, RAU-Gigel va alege o piatră de pe cealaltă treaptă, apoi din nou, va lua fiecare câte o piatră, și căpcăunul nu va mai avea ce dărâma, iar RAU-Gigel va pleca cu prea frumoasa fată.

Cerința

Se vor testa mai multe scenarii de tipul celui de mai sus.

Date de intrare

Fișierul de intrare *donjon.in* conține pe prima linie numărul *Q* reprezentând numărul de scenarii, apoi pe următoarele *Q* rânduri se vor afla câte două numere *N* și *K* separate cu un spațiu, cu semnificația din enunț.

Date de ieșire

Fișierul *donjon.out* va conține *Q* rânduri pe care se află răspunsurile la cele *Q* scenarii, astfel: dacă RAU-Gigel nu reușește să construiască treptele, se va răspunde cu litera mare *C*. Dacă reușește să construiască treptele, se va răspunde cu *x y*, separate printr-un spațiu, unde *x* este litera mare *C* dacă căpcăunul câștigă proba a doua, respectiv litera mare *G* dacă RAU-Gigel salvează prințesa, iar *y* este un număr natural nenul reprezentând numărul de pietre folosite de RAU-Gigel pentru prima treaptă.

Restricții și precizări

- Limită de timp: 0.1 secunde
- Limită de memorie: 1 MiB
- $1 \leq Q \leq 10$, $2 \leq N \leq 10^{18}$, $2 \leq K \leq 10^{9}$
- Pentru teste în valoare de 20 de puncte: $Q \leq 5$, $N \leq 10$ pentru toate scenariile
- Teste în valoare de 30 de puncte sunt favorabile lui RAU-Gigel pentru toate scenariile, în sensul că, dacă acesta trece de prima provocare și reușește să construiască treptele conform cerinței, atunci se garantează că va câștiga și cea de-a doua provocare

Exemplu:

In:	**Out**:	**Explicație:**
3 5 2 6 3 5 3	G 2 C 1 C	Avem *3* scenarii. In primul scenariu sunt *5* pietre, iar lățimea canalului este *2*. Răspunsul pentru acest scenariu va fi *G 2* pentru că RAU-Gigel reușește la ambele provocări, iar prima treaptă construită are 2 pietre (vezi explicația mai sus). Pentru scenariul al doilea, avem *N = 6* și *K = 3*. RAU-Gigel trece de prima proba si construiește cele 3 trepte: *1, 2, 3*. Dar, oricum ar încerca să le dărâme, căpcăunul va avea ultima aruncare. Răspunsul pentru acest scenariu va fi *C 1* pentru că, deși RAU-Gigel trece de prima provocare așezând o piatră ca primă treaptă, el pierde jocul final. Pentru ultimul scenariu, răspunsul este *C*, RAU-Gigel nu poate construi *3* trepte cu 5 pietre, conform cerinței, care să îi asigure trecerea către donjonul prințesei.

Problema 2. Scrabble

Fiind în vacanță, RAU-Gigel petrece mult timp jucându-se pe telefon. El are un joc cu cuvinte, de tip Scrabble, în care piesele sunt inscripționate cu litere (mici sau mari, ale alfabetului englez), fiecare literă din alfabet având o valoare cunoscută, număr natural. Valoarea unui cuvânt este egală cu suma valorilor literelor din cuvânt, fără a se ține cont de frecvența lor.

Valori litere

A, a	B, b	C, c	D, d	E, e	F, f	...	Z, z
1	2	4	8	16	32	...	33.554.432

De exemplu, valoarea cuvântului *„Abac”* este *Val(„Abac”) = Val(‘A’) + Val(’b’) + Val(’c’) = 1+2+4=7*.

Prin *unirea* a două cuvinte se obține cel mai mic (alfanumeric) cuvânt format din toate literele prezente în cele două cuvinte, fără să ținem cont de tipul literei (mică/mare) sau de numărul de apariții. Notăm acest cuvânt cu *a*b*.

Costul unirii dintre două cuvinte este obținut prin însumarea valorilor literelor prezente în *a*b*, dar care nu sunt în *a*, respectiv, care nu sunt în *b,* ignorând tipul lor.

De exemplu, prin unirea cuvintelor a = *Abac”* și b = *„cade”* se obține cuvântul *a*b = „ABCDE”*, iar costul unirii poate fi calculat astfel: *Val(„DE”) + Valoare(„B”) = 24 + 2 = 26.*

Aplicația lui RAU-Gigel generează un șir liniar cu *N* cuvinte, iar RAU-Gigel trebuie să unească două câte două cuvinte alăturate din șir, oricare, plătește costul necesar unirii lor, apoi înlocuiește în șir cele două cuvinte cu cuvântul obținut prin unire. La final, din șirul dat va rămâne un singur cuvânt, iar, pentru obținerea lui, RAU-Gigel va plăti suma tuturor costurilor generate pe parcurs.

Dacă, de exemplu, avem *3* cuvinte: *Abac cade DAC* , mai jos este ilustrată una dintre posibilitățile de unire:

Șir cuvinte	***Acțiune***	***Cost***
Abac cade DAC	Se unesc mai întâi cuvintele *Abac* și *cade* și se obține *ABCDE*, cu costul *26*, iar șirul devine *ABCDE DAC*	26
ABCDE DAC	Se unesc *ABCDE* și *DAC* și se obține *ABCDE*, cu costul 18 , calculat prin însumarea valorilor literelor *B* și *E*	18
ABCDE		Cost total: 44

Cel mai mic cuvânt alcătuit prin unirea celor *3* cuvinte, fără a ține cont de tipul literei (mică/mare) sau frecvență, va fi *„ABCDE”*, iar costul total necesar obținerii lui, conform alegerilor de mai sus, va fi 26 + 18 = 44.

Dar mai este o variantă: RAU-Gigel poate uni mai întâi cuvintele *„cade”* și *„DAC”* și obține *„ACDE”*, cu costul *16*, apoi unește *„Abac”* și *„ACDE”* și obține *„ABCDE”*, cu costul *26* . Costul total va fi de această dată: *16 + 26 = 42*.

Așadar, costul total minim este dat de cea de-a doua variantă și este *42*.

Cerința

Dându-se un sir de *N* cuvinte, să se afle cuvântul final rezultat în urma reuniunii a două câte două cuvinte alăturate și costul total minim necesar obținerii acestuia.

Date de intrare

Fișierul *scrabble.in* conține pe primul rând numărul natural nenul *N*, reprezentând numărul de cuvinte. Pe următoarele *N* rânduri se află cele *N* cuvinte.

Date de ieșire

Fișierul *scrabble.out* va conține, pe un sigur rând, cele două valori: cuvântul final și costul total minim necesar obținerii lui, conform cerinței. Cele două valori sunt separate printr-un spațiu.

Restricții și precizări

- Limită de timp: 0.2 secunde
- Limită de memorie: 1 MiB
- $2 \leq N \leq 10^2$, cuvintele nu au mai mult de 100 de litere
- Pentru teste în valoare de 20 de puncte, N <= 4
- Pentru teste în valoare de încă 10 de puncte, toate cele N cuvinte au aceeași valoare
- Pentru teste în valoare de încă 20 de puncte, orice cuvânt include, în componența sa, literele cuvintelor din stânga lui (ignorând tipul literei), plus, eventual, alte litere
- Tipul literei = literă mică sau mare

Exemplu:

In:	**Out**:	**Explicație:**
3 Abac cade DAC	ABCDE 42	Cel mai mic cuvânt alcătuit prin unirea celor *3* cuvinte, fără a ține cont de tipul literei (mică/mare) sau frecvență, este: „*ABCDE*”, iar costul total minim necesar obținerii lui este *42*.

Problema 3. Gcd Tree

Cerința

RAU-Gigel și Vlad sunt prieteni buni și le place tot timpul să se provoace unul pe altul. De data aceasta, RAU-Gigel a inventat o problemă interesantă de matematică.

Acesta are un arbore secret cu ***N*** noduri (numerotate de la ***1*** la ***N***), în care fiecare nod are asociată o valoare (pe lângă numărul său de ordine din arbore), care este un număr natural și ii spune lui Vlad informații despre unele drumuri din acest arbore. O informație are forma ***x, y, val*** și reprezintă faptul că lanțul din arbore de la nodul ***x*** la nodul ***y*** are cel mai mare divizor comun al valorilor asociate nodurilor acestuia egal cu ***val***, unde ***val*** este un număr natural nenul.

Vlad știe că RAU-Gigel minte câteodată și s-ar putea ca unele dintre restricțiile date să fie eronate, astfel că vrea să afle întâi folosind informațiile ce le are la îndemână dacă ar putea exista un arbore care să respecte **toate** restricțiile date de prietenul său.

Pentru că știe ce programator iscusit ești, Vlad ți-ar fi foarte recunoscător daca l-ai putea ajuta cu această problemă prin a scrie un program care să o rezolve cât mai eficient.

Date de intrare

Fișierul de intrare **gcd_tree.in** conține pe prima linie numărul **T**, care reprezintă numărul de scenarii de test care trebuie rezolvate.
Pentru fiecare test, prima linie a conține ***2*** numere ***N*** și ***M***, reprezentând lungimea șirului secret și numărul de restricții pe care acesta trebuie să le respecte.
A doua linie conține ***N – 1*** numere naturale, al ***i***-lea număr reprezentând părintele nodului ***i + 1*** din arbore (se consideră că arbore are nodurile indexate de la ***1*** și rădăcina în nodul ***1***).
Fiecare dintre următoarele ***M*** linii din test conține câte ***3*** numere ***x, y, val***, capetele lanțului din arbore pe care se aplică restricția, respectiv valoarea acestei restricții.

Date de ieșire

Programul trebuie să afișeze în fișierul de ieșire **gcd_tree.out**, pe o singură linie, fără spații, un șir de lungime ***T*** format din valorile 0 sau 1, unde elementul de pe poziția ***i*** este ***1*** dacă scenariul de test cu numărul ***i*** din input admite un arbore posibil care să satisfacă toate restricțiile sau ***0*** în caz contrar.

Restricții și precizări

- Limită de timp: 2 secunde
- Limită de memorie: 512MB / 128MB
- $\mathbf{1 \leq T \leq 10}$
- $\mathbf{1 \leq N, M \leq 100.000}$
- $\mathbf{1 \leq x, y \leq N}$
- $\mathbf{1 \leq val \leq 20}$
- **Se notează cu *SN*** suma valorilor ***N*** din toate scenariile de test dintr-un fișier de intrare și cu *SM* suma valorilor ***M*** din toate scenariile de test dintr-un fișier de intrare
- $\mathbf{1 \leq SN \leq 300.000}$
- $\mathbf{1 \leq SM \leq 300.000}$
- Pentru teste în valoare de **13** puncte, arborele este un lanț simplu de forma ***1 – 2 – … – N***; $\mathbf{1 \leq N, M \leq 1.000; 1 \leq SN, SM \leq 3.000}$ și $\mathbf{val \in \{2, 4, 8, 16\}}$.
- Pentru alte teste în valoare de **19** puncte, arborele este un lanț simplu de forma ***1 – 2 – … – N***; $\mathbf{1 \leq N, M \leq 1.000}$; $\mathbf{1 \leq SN, SM \leq 3.000}$ și $\mathbf{1 \leq val \leq 20}$.

- Pentru alte teste în valoare de **17** puncte, arborele este un lanț simplu de forma ***1 – 2 – … – N*** și $\boldsymbol{val} \in \{2, 4, 8, 16\}$.
- Pentru alte teste în valoare de **24** puncte, arborele este un lanț simplu de forma ***1 – 2 – … – N***.
- Pentru alte teste în valoare de **8** puncte, $1 \leq N, M \leq 1.000$; $1 \leq SN, SM \leq 3.000$ și $\boldsymbol{val} \in \{2, 4, 8, 16\}$.
- Pentru alte teste în valoare de **6** puncte, $1 \leq N, M \leq 1.000$; $1 \leq SN, SM \leq 3.000$ și $1 \leq \boldsymbol{val} \leq 20$.
- Pentru alte teste în valoare de **9** puncte, $\boldsymbol{val} \in \{2, 4, 8, 16\}$.
- Pentru restul testelor nu există restricții suplimentare, adică se păstrează doar condițiile inițiale.

Exemplu:

In:	**Out**:	**Explicație:**
2 4 2 3 1 3 1 4 11 2 4 12 3 2 1 2 1 3 4 3 2 3	10	Pentru primul scenariu de test din exemplu, arborele secret ar putea avea valorile ***11, 12, 132, 132*** asociate nodurilor sale (scrise în ordine, de la nodul cu numărul ***1*** la nodul cu numărul ***4***). Pentru al doilea scenariu de test se poate demonstra ușor matematic că nu există soluție.

Succes!