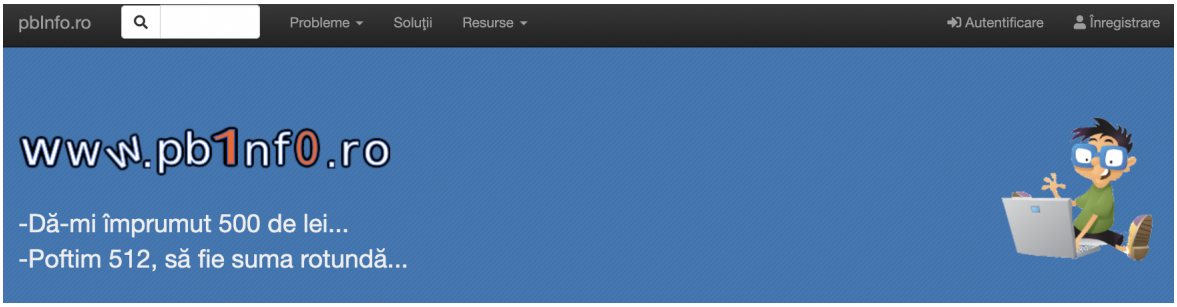
Infoleague April Fools Contest 2024 Editorial

Suma rotundă

Această problemă este o referință la unul dintre splash text-urile de pe pbinfo:



Răspunsul la această problemă este egal cu cea mai mică putere de 2 mai mare sau egală cu n.

Complexitate: $O(log(n))$

What

În enunț nu scrie nicăieri că a și b trebuie să fie numere întregi :)

Prin urmare, o soluție validă ar fi $a = 0.5$ și $b = -0.5$.

Complexitate: $O(1)$

Două mii douăzeci și patru

Știind că:

- 2024 (titlul problemei) este un an bisect
- $1 \leq n \leq 366$
- $1 \leq d \leq 31$

Putem deduce că n reprezintă indicele unei date dintr-un an bisect, iar d reprezintă ziua din data respectivă. De exemplu:

- $n = 1$: Corespunde datei de 1 ianuarie, se va afișa 1.
- $n = 31$: Corespunde datei de 31 ianuare, se va afișa 31.
- $n = 60$: Corespunde datei de 29 februarie, se va afișa 29.

Complexitate: $O(1)$

Cel mai tare hack

Enunțul aparent normal este menit să distragă de la faptul că aceasta este o problemă de April Fools, prin urmare soluția ei nu are cum să fie convențională.

În mod normal, numărul minim de query-uri necesare pentru a rezolva problema este 2:

```
int suma=cel_mai_tare_hack(-10000)-20000; // |-10000-a|+|-10000-b|=a+b+20000;
int diferenta=cel_mai_tare_hack(suma/2); // a<=suma/2<=b, diferenta=b-a

return {(suma-diferenta)/2,(suma+diferenta)/2};
```

Această rezolvare va lua aproximativ 70 de puncte.

Problema se poate de asemenea rezolva într-un singur query dacă căutăm binar pe teste valorile lui a și b. Perechile (a, b) din teste pot fi determinate unic de `cel_mai_tare_hack(69)`.

Numărul minim de query-uri necesare pentru a rezolva problema este 0. Deoarece grader-ul nu a schimbat stream-ul de citire, a și b încă pot fi citite din `stdin`:

```
std::cin.seekg(0,std::cin.beg); // reseteaza cin
int a,b;
std::cin>>a>>b;
return {a,b};
```

Complexitate: $O(1)$

A069

În primul rând, enunțul ne oferă valorile pentru o_1, o_2, o_3, o_4 și o_6. Deoarece nu știm valoarea lui o_5, putem căuta pe OEIS secvența $1, 2, 4, 7, ?, 11$.

Șirul odios este de fapt A000069 - șirul numerelor odioase (care au un număr impar de biți de 1).

Astfel, problema se reduce la a afișa termenul o_n (al n-ulea număr natural cu un număr impar de biți de 1).

A000069 ne oferă și o metodă de a afla acest număr: `"[A000069] Has asymptotic density 1/2, since exactly 2 of the 4 numbers 4k, 4k+1, 4k+2, 4k+3 have an even sum of bits, while the other 2 have an odd sum."`

Prin urmare, știm că până la $(n/2) \cdot 4 - 4$ există $\frac{(n/2) \cdot 4 - 4}{2} = (n/2) \cdot 2 - 2$ termeni din șirul o. Începând cu $(n/2) \cdot 4 - 3$, putem căuta brut o_n.

Anumite implementări ar putea să dea un răspuns greșit pentru $n = 1$, însă acest caz poate fi tratat separat.

Complexitate: $O(log(n))$

O scrisoare pierdută

La această problemă trebuie să găsim scrisoarea. Mai exact, trebuie să verificăm dacă există string-ul `"scrisoare"` în matricea dată în una dintre cele 8 orientări posibile.

Complexitate: $O(9^2)$ pe testcase

Căutare binară

Deoarece nu există două teste cu același n, putem căuta binar testele. De exemplu, următoarea sursă va da `runtime error` dacă n este impar, respectiv `wrong answer` dacă n este par:

```
#include<bits/stdc++.h>

using namespace std;

int main()
{
    int n;
    cin>>n;
    assert(n%2==0);
}
```

Astfel, putem identifica ultimul bit al lui n pentru toate testele. Se va proceda similar pentru ceilalți 11 biți :)

O metodă care poate afla mai mulți biți deodată se bazează pe manipularea timpului de execuție a programului cu `steady_clock`:

```
#include<bits/stdc++.h>

using namespace std;
typedef long long ll;
const ll NMAX=1e5+5,KMAX=105;
ll getTime(){
    return chrono::steady_clock::now().time_since_epoch().count();
}
int main()
{
    ll n,T=getTime();
    cin>>n;
    while(getTime()-T<1e8*(n&7));
    cout<<0;
    return 0;
}
```

Știind că numărul format din ultimii 3 biți ai lui n este $n\&7$, acest program va rula în **aproximativ** $100 \cdot (n\&7)$ milisecunde pentru fiecare test.

Deși această metodă este relativ imprecisă, putem afla datele de intrare a tuturor testelor în 4 surse.

Pentru a afla răspunsurile corecte pentru fiecare test, vom avea nevoie de o sursă suplimentară care afișează mereu 0. Testele unde această sursă va lua accepted au răspunsul corect 0, iar celelalte vor avea răspunsul 1.

Complexitate: $O(log(n))$ surse, $O(1)$ per sursă

April Fools Final Boss

Sursele la această problemă sunt punctate în funcție de dimensiunea lor în bytes. Dacă o sursă corectă are n bytes, atunci ea va primi:

- 100 puncte, dacă $n \leq 32$
- $100 - (n - 32) \cdot 0.71$ puncte, dacă $32 < n \leq 71$
- $72.31 - (n - 71) \cdot 3.31$ puncte, dacă $71 < n \leq 76$
- $55.76 - (n - 76) \cdot 1.77$ puncte, dacă $76 < n \leq 81$
- $45 \cdot (\frac{82}{n})^3$ puncte, dacă $82 \leq n < 290$
- 1 punct, dacă $n \geq 290$

Din câte știm noi, cea mai scurtă sursă corectă în C++ are 65 de bytes:

```
#import<iostream>
main(){long a,b;std::cin>>a>>b;std::cout<<a+b;}
```

Singurul limbaj de programare de pe kilonova care poate rezolva problema în cel mult 32 de caractere este python:

```
print(int(input())+int(input()))
```

Complexitate: $O(1)$