



سوال A:

[Problem - 427A - Codeforces](#)

برای حل این سوال کافیست رو به جلو حرکت کرده و دو دسته از اعداد را نگه داریم یک دسته تعداد پلیسی که تا به حال به کار گرفته نشده اند و دسته دوم جرم هایی که حساب نشده اند با گرفتن هر ورودی جدید در صورت پلیس بودن به دسته پلیس اضافه و در صورت جرم بودن اگر پلیسی باقی مانده بود از دسته پلیس کم میکنیم در غیر این صورت به جرم های حساب نشده اضافه میکنیم

C++:

```
#include <bits/stdc++.h>
using namespace std;

int main(){
    int n, police = 0, ans = 0;
    cin >> n;
    for (int i=0; i<n; i++){
        int tmp;
        cin >> tmp;

        if (tmp == -1){
            if (police >= 1) police--;
            else ans++;
        }

        else
            police += tmp;
    }

    cout << ans << endl;
}
```



برای پیدا کردن بیشترین ساعات متوالی استراحت باید تمام زمان های متوالی را حساب کرده و از بین آنها بیشترین را نگهداریم. تنها نکته ای که باید به آن دقت کنیم این است که اگر یک روز با استراحت تمام شود و روز بعدی نیز با استراحت شروع شود باید از اولین جایی که از روز قبل استراحت شروع شده تا آخرین جایی که در روز بعد استراحت ادامه دارد را محاسبه کنیم. برای اینکار میتوان دوروز را پست سر هم نوشت و بین این دوروز بیشترین را محاسبه کرد.

C++:

```
#include <bits/stdc++.h>
using namespace std;

int main(){
    int n;
    cin >> n;

    vector <int> a(2*n);

    for (int i=0; i<n; i++){
        cin >> a[i];
        a[n+i] = a[i];
    }

    int ans = 0;
    for (int i=0; i<2*n; i++)
        if (a[i] == 1){
            int tmp = 0;
            for ( ; i<2*n && a[i] == 1; i++)
                tmp++;
            ans = max (ans, tmp);
        }

    cout << ans << endl;
}
```



برای این سوال کافی است در آرایه ای برای هر فرد مقداری که آن فرد به آبنبات در آن لحظه نیاز دارد را نگهداریم و این مقدار را هر بار بعد از گرفتن آبنبات به روز کنیم.

کافی است صفی را تشکیل دهیم تمام بچه ها را درون صف قرار دهیم و هر بار اولین نفر صف را برداریم و به او آبنبات داده و درایه تعداد آبنبات او را به روز کنیم. اگر آن فرد هنوز هم آبنبات نیاز داشت او را به انتهای صف برگردانیم و اگر نداشت او را حذف کنیم. این کار را تا زمانی که به فرد واحدی برسیم تکرار میکنیم.

C++:

```
#include <bits/stdc++.h>
using namespace std;

int main(){
    int n, m;
    cin >> n >> m;

    vector <int> num(n); // array is going to keep the candies of each person

    for (int i=0; i<n; i++) // initialize the array of candies
        cin >> num[i];

    queue <int> stu; // making the queue
    for (int i=0; i<n; i++) // put the people in the line for the first time
        stu.push(i);

    while (stu.size() > 1){
        int i = stu.front();
        stu.pop();
        num[i] -= m;

        if (num[i] > 0)
            stu.push(i);
    }

    cout << stu.front()+1 << endl;
}
```



- ❖ اگر n را بتوان به شکل $2x$ یا $4x$ نوشت که x یک عدد مربع است، پاسخ YES است. در غیر این صورت NO است.
- ❖ برای تجسم این ساختار، ابتدا با ساختن یک مربع کوچکتر با استفاده از 2 یا 4 قطعه شروع می کنیم. فقط کافی است از x تا از آن مربع های کوچکتر برای ساختن یک مربع بزرگتر استفاده کنیم .
- ❖ بیایید ثابت کنیم که هیچ پاسخ دیگری وجود ندارد . بیایید هر قطعه مثلث را طوری تعریف کنیم که یک ضلع کوتاه به طول 1 و یک ضلع بلندتر به طول رادیکال 2 داشته باشد. یک ضلع مربع را در نظر بگیرید و فرض کنید که a مثلث در ضلع کوتاه و b مثلث در ضلع بلندتر دارد. طول ضلع $a+2-\sqrt{b}$ خواهد بود. مساحت مربع یک عدد گویا است زیرا مساحت هر قطعه مثلث گویا است. بنابراین $(a+2-\sqrt{b})^2$ باید گویا باشد، به این معنی که a 0 است، یا b 0 است. اگر هر کدام 0 باشد، می توانیم از ساختار پاراگراف قبل استفاده کنیم.

C++:

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
bool sq(int n){
    int a = sqrt(n);
    return a*a==n;
}
int main(){
    ll t;
    cin >> t;
    while(t--){
        ll n;
        cin >> n;
        if(n%2==0&&sq(n/2)) cout<<"YES"<<endl;
        else if(n%4==0&&sq(n/4)) cout<<"YES"<<endl;
        else cout<<"NO"<<endl;
    }
}
```

Python:

```
for i in range(int(input())):
    n=int(input())
    print('YES' if n%2==0 and n//2==round((n//2)**0.5)**2 or n%4==0 and
n//4==round((n//4)**0.5)**2 else 'NO')
```



ما به رشته های واقعی اهمیت نمیدهیم، تمام اطلاعاتی که نیاز داریم این است تعداد جفت ها $\{0,0\}$, $\{1,0\}$, $\{0,1\}$, $\{1,1\}$ آن را بشماریم و سپس الگوریتمی حریصانه را برای اولین بازیکن دنبال میکنیم: سعی کنید ایندکس با $\{1,1\}$ در صورت وجود بیابیم، و بعد $\{0,1\}$ و بعد $\{1,0\}$ و در نهایت $\{0,0\}$.

برای بازیکن دوم استراتژی مشابه: اول $\{1,1\}$ از $\{1,0\}$ از $\{0,1\}$ از $\{0,0\}$ پس از آن فقط مقایسه کنید که چه کسی 1 بیشتر دارد.

C++:

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;

int main(){
    ll n;
    string s1;
    string s2;
    cin >> n >> s1 >> s2;
    ll f=0,c=0,s=0;
    for(ll i=0;i<2*n;i++){
        if(s1[i]=='1'&&s2[i]=='1')
            c++;
        else if(s1[i]=='1')
            f++;
        else if(s2[i]=='1')
            s++;
    }
    f = c%2 ? f+1:f;
    if(s>f+1)
        cout<<"Second"<<endl;
    else if(f>s)
        cout<<"First"<<endl;
    else
        cout<<"Draw"<<endl;
}
```

Python:

```
n = int(input())
a, b = input(), input()
t = {i + j: 0 for i in '01' for j in '01'}
for i in range(2 * n): t[a[i] + b[i]] += 1
d = t['11'] & 1
d += (t['10'] - t['01'] + 1 - d) // 2
if d > 0: d = 1
elif d < 0: d = 2
print(['Draw', 'First', 'Second'][d])
```

