|  |  |
| --- | --- |
| [A](https://codeforces.com/group/Vk6pDIfre6/contest/315142/problem/A) | [Make All Odd](https://codeforces.com/group/Vk6pDIfre6/contest/315142/problem/A) |

По сути, нужно посчитать количество нечетных чисел в массиве.

**Решение C++**

#include<cstdio>

**using** **namespace** std**;**

int main**(){**

int n**,** T**,** i**,** a**,** b**;**

scanf**(**"%d"**,** **&**T**);**

**while(**T**--){**

scanf**(**"%d"**,** **&**n**);**

**for(**i **=** 1**,** b **=** 0**;**i **<=** n**;**i**++){**

scanf**(**"%d"**,** **&**a**);**

b **+=** a **&** 1**;**

**}**

printf**(**"%d\n"**,** b **?** n **-** b **:** **-**1**);**

**}**

**return** 0**;**

**}**

**Решение Python**

t **=** int**(**input**())**

**for** i **in** range**(**t**):**

n **=** int**(**input**())**

nn **=** input**().**split**()**

odds **=** sum**(**int**(**x**)** **%** 2 **for** x **in** nn**)**

**print(-**1 **if** odds **==** 0 **else** n **-** odds**)**

|  |  |
| --- | --- |
| [B](https://codeforces.com/group/Vk6pDIfre6/contest/315142/problem/B) | [Nezzar и разноцветные шары](https://codeforces.com/group/Vk6pDIfre6/contest/315142/problem/B) |

Решение задачи сводится к подсчету количества встречаемости каждого числа. После чего нужно найти максимум.

**Решение C++**

#include <iostream>

#include <vector>

#include <algorithm>

**using** **namespace** std**;**

int cnt**[**111**];**

int main**()** **{**

int tst**;**

cin **>>** tst**;**

**while** **(**tst**--){**

**for** **(**int i **=** 0**;** i **<** 111**;** i**++)**

cnt**[**i**]** **=** 0**;**

int res **=** 0**;**

int n**,** v**;**

cin **>>** n**;**

**for** **(**int i **=** 0**;** i **<** n**;** i**++){**

cin **>>** v**;**

cnt**[**v**]++;**

**if** **(**cnt**[**v**]** **>** res**)**

res **=** cnt**[**v**];**

**}**

cout **<<** res **<<** endl**;**

**}**

**return** 0**;**

**}**

**Решение Python**

**for** \_ **in** range**(**int**(**input**())):**

n**=**int**(**input**())**

l**=**list**(**map**(**int**,** input**().**split**()))**

x**=**max**(**l**,**key**=**l**.**count**)**

**print(**l**.**count**(**x**))**

|  |  |
| --- | --- |
| [C](https://codeforces.com/group/Vk6pDIfre6/contest/315142/problem/C) | [Путешествие блохи](https://codeforces.com/group/Vk6pDIfre6/contest/315142/problem/C) |

В этой задаче требовалось лишь провести эмуляцию ходов блохи достаточное количество раз. Действительно, после 2*n* шагов блоха переместится на 1 + 2 + ... + 2*n* = *n*(2*n* + 1) кочек по часовой стрелке, то есть вернется в исходную позицию. Более того, следующие ее прыжки будут такими же как в начале, так как 2*n* ≡ 0(*modn*). Поэтому после 2*n* прыжков блоха не посетит никакие новые кочки. Таким образом, достаточно проэмулировать первые 2*n* шагов и проверить посещены ли все кочки.

На самом деле, нетрудно доказать, что блоха посетит все кочки в точности тогда, когда *n* --- степень двойки и получить альтернативное решение. Например, такое: printf("%s", n&(n-1) : "NO" ? "YES");

**Решение C++**

#include<iostream>

**using** **namespace** std**;**

int main**()**

**{**

int n**;**

cin **>>** n**;**

cout **<<** **(**n**&**n**-**1 **?** "NO" **:** "YES"**);**

**}**

**Решение Python**

n **=** int**(**input**())**

flag **=** 1

**while** n **>** 1**:**

**if** n **%** 2**:**

flag **=** 0

**break**

n **=** n **//** 2

**if** flag**:**

**print(**"YES"**)**

**else:**

**print(**"NO"**)**

|  |  |
| --- | --- |
| [D](https://codeforces.com/group/Vk6pDIfre6/contest/315142/problem/D) | [Монеты](https://codeforces.com/group/Vk6pDIfre6/contest/315142/problem/D) |

Обратите внимание, что использование монет максимальной стоимости всегда будет оптимальным. Следовательно, мы можем использовать floor(S / n) монеты достоинством n. Теперь, если Smodn не равно 0, то нам нужно использовать еще одну монету стоимостью S mod n. Следовательно, наш ответ можно записать как ceil (S / n)

**Решение C++**

#include <bits/stdc++.h>

**using** **namespace** std**;**

#define IOS ios::sync\_with\_stdio(0); cin.tie(0); cout.tie(0);

#define endl "\n"

#define int long long

const int N**=**1e5**+**5**;**

int32\_t main**()**

**{**

IOS**;**

int n**,** s**;**

cin**>>**n**>>**s**;**

int ans**=(**s**+**n**-**1**)/**n**;**

cout**<<**ans**;**

**return** 0**;**

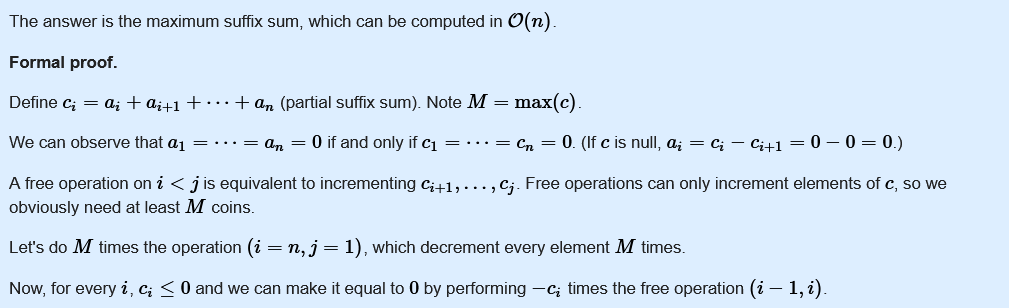
**}**

**Решение Python**

n**,**s**=**map**(**int**,**input**().**split**())**

**print(-(-**s**//**n**))**

|  |  |
| --- | --- |
| [E](https://codeforces.com/group/Vk6pDIfre6/contest/315142/problem/E) | [Уничтожение массива](https://codeforces.com/group/Vk6pDIfre6/contest/315142/problem/E) |



**Решение C++**

#include <bits/stdc++.h>

**using** **namespace** std**;**

int main**()** **{**

ios**::**sync\_with\_stdio**(false),** cin**.**tie**(**0**);**

int t**;** cin **>>** t**;**

**while** **(**t**--)** **{**

int n**;**

cin **>>** n**;**

long long cur **=** 0**;**

**for** **(**int i **=** 0**;** i **<** n**;** **++**i**)** **{**

long long x**;** cin **>>** x**;**

cur **=** max**(**0LL**,** cur **+** x**);**

**}**

cout **<<** cur **<<** "\n"**;**

**}**

**}**

**Решение Python**

**for** \_ **in** range**(**int**(**input**())):**

n**,**s**=**int**(**input**()),**0

**for** i **in** map**(**int**,**input**().**split**()):**

s**=**max**(**0**,**s**+**i**)**

**print(**s**)**

|  |  |
| --- | --- |
| [F](https://codeforces.com/group/Vk6pDIfre6/contest/315142/problem/F) | [Восстановление массива](https://codeforces.com/group/Vk6pDIfre6/contest/315142/problem/F) |

Изначально сделаем три запроса на сумму чисел *a*1 + *a*2 = *c*1, *a*1 + *a*3 = *с*2 и *a*2 + *a*3 = *c*3.

После этого мы получаем систему из трех уравнений с тремя неизвестными *a*1, *a*2, *a*3. После простых вычислений получим, что *a*3 = (*c*3 - *c*1 + *c*2) / 2. После этого легко находятся *a*1 и *a*2. Теперь мы знаем значения *a*1, *a*2, *a*3, потратив на это 3 запроса.

Затем для всех *i* от 4 до *n* нужно сделать запрос на сумму *a*1 + *ai*. Если очередная сумма равна *ci*, то *ai* = *ci* - *a*1 (напомним, что мы уже знаем значение *a*1).

Таким образом можно восстановить весь массив, потратив на это ровно *n* запросов.

**Решение C++**

#include<bits/stdc++.h>

**using** **namespace** std**;**

int a**[**5002**];**

int main**()**

**{**

ios**::**sync\_with\_stdio**(false);**

int n**,** i**;**

cin **>>** n**;**

**for(**i **=** 2**;** i **<=** n**;** i**++)**

**{**

cout **<<** "? 1 " **<<** i **<<** endl**;**

cin **>>** a**[**i**];**

**}**

cout **<<** "? 2 3" **<<** endl**;**

cin **>>** a**[**1**];**

cout **<<** "! " **<<** **(**a**[**2**]** **+** a**[**3**]** **-** a**[**1**])** **/** 2**;**

**for(**i **=** 2**;** i **<=** n**;** i**++)**

cout **<<** " " **<<** a**[**i**]** **-** **(**a**[**2**]** **+** a**[**3**]** **-** a**[**1**])** **/** 2**;**

**return** 0**;**

**}**

**Решение Python**

n **=** int**(**input**())**

**print(**'? 1 2'**)**

x **=** int**(**input**())**

**print(**'? 2 3'**)**

y **=** int**(**input**())**

**print(**'? 1 3'**)**

z **=** int**(**input**())**

b **=** **(**x **+** y **-** z**)** **//** 2

a **=** **[**x **-** b**,** b**,** y **-** b**]**

**for** i **in** range**(**3**,** n**):**

**print(**f'? {i} {i + 1}'**)**

a**.**append**(**int**(**input**())** **-** a**[-**1**])**

**print(**'!'**,** **\***a**)**

|  |  |
| --- | --- |
| [G](https://codeforces.com/group/Vk6pDIfre6/contest/315142/problem/G) | [Идеальная клавиатура](https://codeforces.com/group/Vk6pDIfre6/contest/315142/problem/G) |

Задача может быть решена при помощи жадного алгоритма. Будем поддерживать клавиатуру с буквами, которые уже встречались, и текущую позицию на ней.

Если очередная буква строки уже есть на клавиатуре, то она должна быть соседней с текущей, иначе ответа не существует.

Если такой буквы еще не было, то мы можем добавить ее в соседнюю свободную позицию. Если свободной клетки нет, то ответа не существует.

В конце необходимо добавить буквы, которых не было в строке *s*

.

**Решение C++**

#include <bits/stdc++.h>

**using** **namespace** std**;**

#define sz(a) int((a).size())

#define all(a) (a).begin(), (a).end()

#define forn(i, n) for (int i = 0; i < int(n); ++i)

void solve**()** **{**

string s**;**

cin **>>** s**;**

vector**<**bool**>** used**(**26**);**

used**[**s**[**0**]** **-** 'a'**]** **=** **true;**

string t**(**1**,** s**[**0**]);**

int pos **=** 0**;**

**for** **(**int i **=** 1**;** i **<** sz**(**s**);** i**++)** **{**

**if** **(**used**[**s**[**i**]** **-** 'a'**])** **{**

**if** **(**pos **>** 0 **&&** t**[**pos **-** 1**]** **==** s**[**i**])** **{**

pos**--;**

**}** **else** **if** **(**pos **+** 1 **<** sz**(**t**)** **&&** t**[**pos **+** 1**]** **==** s**[**i**])** **{**

pos**++;**

**}** **else** **{**

cout **<<** "NO" **<<** endl**;**

**return;**

**}**

**}** **else** **{**

**if** **(**pos **==** 0**)** **{**

t **=** s**[**i**]** **+** t**;**

**}** **else** **if** **(**pos **==** sz**(**t**)** **-** 1**)** **{**

t **+=** s**[**i**];**

pos**++;**

**}** **else** **{**

cout **<<** "NO" **<<** endl**;**

**return;**

**}**

**}**

used**[**s**[**i**]** **-** 'a'**]** **=** **true;**

**}**

forn**(**i**,** 26**)** **if** **(!**used**[**i**])**

t **+=** char**(**i **+** 'a'**);**

cout **<<** "YES" **<<** endl **<<** t **<<** endl**;**

**}**

int main**()** **{**

int tc**;**

cin **>>** tc**;**

forn**(**i**,** tc**)** solve**();**

**}**

**Решение Python**

**from** collections **import** defaultdict

al **=** **[**chr**(**ord**(**'a'**)+**i**)** **for** i **in** range**(**26**)]**

t **=** int**(**input**())**

**for** \_ **in** range**(**t**):**

a **=** input**()**

d **=** defaultdict**(lambda:** set**())**

**for** u**,** v **in** zip**(**a**,** a**[**1**:]):**

d**[**u**].**add**(**v**)**

d**[**v**].**add**(**u**)**

s **=** **None**

x **=** **False**

**for** i**,** v **in** d**.**items**():**

**if** len**(**v**)** **>** 2**:**

x **=** **True**

**break**

**elif** len**(**v**)** **==** 1**:**

s **=** i

**if** len**(**d**)** **==** 0**:**

**print(**"YES"**)**

**print(**""**.**join**(**al**))**

**elif** x **or** s **is** **None:**

**print(**"NO"**)**

**else:**

z **=** **[**s**,** d**[**s**].**pop**()]**

**while** **True:**

p**,** q **=** z**[-**1**],** z**[-**2**]**

d**[**p**].**discard**(**q**)**

**if** d**[**p**]:**

z**.**append**(**d**[**p**].**pop**())**

**else:**

**break**

**print(**"YES"**)**

**print(**""**.**join**(**z**)** **+** ""**.**join**(**b **for** b **in** al **if** b **not** **in** z**))**

|  |  |
| --- | --- |
| [H](https://codeforces.com/group/Vk6pDIfre6/contest/315142/problem/H) | [Алёна и дерево](https://codeforces.com/group/Vk6pDIfre6/contest/315142/problem/H) |

Будем делать dfs. Пусть мы сейчас стоим в вершине *u*. Пусть *v* — это какой-то предок вершины *u*. Тогда *dist*(*v*, *u*) = *dist*(1, *u*) - *dist*(1, *v*). Если *dist*(*v*, *u*) > *au*, то вершина *u* заставляет вершину *v* грустить. Так что необходимо удалить все поддерево вершины *u*. Соответственно, в dfs можно поддерживать минимум среди *dist*(1, *v*), где *v* — это предок *u*(вершина, в которой мы сейчас стоим). И если разность *dist*(1, *u*) и этого минимума больше *au*, то удаляем *au* вместе со всем поддеревом.

**Решение C++**

#include <bits/stdc++.h>

**using** **namespace** std**;**

#define x first

#define y second

int n**;**

int a**[**100009**];**

vector**<**pair**<**int**,**int**>** **>** G**[**100009**];**

int dfs**(**int u**,**int fa**,**int q**)**

**{**

**if(**a**[**u**]<**q**)return** 0**;**

**if(**q**<**0**)**q**=**0**;**

int ans**=**1**;**

**for(**int i**=**0**;**i**<**G**[**u**].**size**();++**i**)**

**{**

pair**<**int**,**int**>** **&**e**=**G**[**u**][**i**];**

**if(**e**.**x**==**fa**)continue;**

ans**+=**dfs**(**e**.**x**,**u**,**q**+**e**.**y**);**

**}**

**return** ans**;**

**}**

int main**()**

**{**

scanf**(**"%d"**,&**n**);**

**for(**int i**=**1**;**i**<=**n**;++**i**)**scanf**(**"%d"**,&**a**[**i**]);**

**for(**int i**=**2**;**i**<=**n**;++**i**)**

**{**

int u**,**c**;**

scanf**(**"%d%d"**,&**u**,&**c**);**

G**[**i**].**push\_back**(**make\_pair**(**u**,**c**));**

G**[**u**].**push\_back**(**make\_pair**(**i**,**c**));**

**}**

printf**(**"%d\n"**,**n**-**dfs**(**1**,**0**,**0**));**

**return** 0**;**

**}**

**Решение Python**

**from** sys **import** stdin

input**=lambda** **:** stdin**.**readline**().**strip**()**

**from** math **import** ceil**,**sqrt**,**factorial**,**gcd

**from** collections **import** deque

**def** dfs**(**x**):**

stack**=[**x**]**

**while** stack**:**

x**=**stack**.**pop**()**

**for** i **in** graph**[**x**]:**

graph**[**i**].**remove**(**x**)**

stack**.**append**(**i**)**

**def** dfs\_simple**(**x**):**

stack**=[**x**]**

t**=**0

**while** stack**:**

x**=**stack**.**pop**()**

t**+=**1

**for** i **in** graph**[**x**]:**

stack**.**append**(**i**)**

**return** t

n**=**int**(**input**())**

l**=**list**(**map**(**int**,**input**().**split**()))**

graph**={**i**:**set**()** **for** i **in** range**(**n**)}**

d**={}**

**for** i **in** range**(**1**,**n**):**

a**,**b**=**map**(**int**,**input**().**split**())**

a**-=**1

graph**[**a**].**add**(**i**)**

graph**[**i**].**add**(**a**)**

d**[(**a**,**i**)]=**b

dfs**(**0**)**

z**=[]**

stack**=[[**0**,**0**]]**

# print(graph)

**while** stack**:**

x**=**stack**.**pop**()**

**if** x**[**1**]>**l**[**x**[**0**]]:**

z**.**append**(**x**[**0**])**

**else:**

**for** i **in** graph**[**x**[**0**]]:**

**if** **(**i**,**x**[**0**])** **in** d**:**

e**=**d**[(**i**,**x**[**0**])]**

**else:**

e**=**d**[(**x**[**0**],**i**)]**

**if** e**>=**0**:**

**if** x**[**1**]>=**0**:**

t**=**x**[**1**]+**e

**else:**

t**=**e

**else:**

**if** x**[**1**]>=**0**:**

t**=**x**[**1**]+**e

**else:**

t**=**e

stack**.**append**([**i**,**t**])**

# print(z)

count**=**0

**for** i **in** z**:**

count**+=**dfs\_simple**(**i**)**

**print(**count**)**