|  |  |
| --- | --- |
| [A](https://codeforces.com/group/Vk6pDIfre6/contest/316095/problem/A) | [Studying Algorithms](https://codeforces.com/group/Vk6pDIfre6/contest/316095/problem/A) |

Жадное решение. Выбираем алгоритмы по мере увеличения времени изучения.

**Решение C++**

#include<bits/stdc++.h>

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

int main**(){**

int n**,**x**;**

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

int a**[**n**];**

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

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

**}**

sort**(**a**,**a**+**n**);**

int i**=**0**;**

int ans**=**0**;**

**while(**i**<**n**){**

x**-=**a**[**i**];**

**if(**x**>=**0**)**ans**++;**

i**++;**

**}**

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

**}**

**Решение Python**

**from** sys **import** exit

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

c **=** 0

a **=** **[**int**(**i**)** **for** i **in** input**().**split**()]**

a**.**sort**()**

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

c **+=** a**[**i**]**

**if** c **>** x**:**

**print(**i**)**

exit**()**

**print(**n**)**

|  |  |
| --- | --- |
| [B](https://codeforces.com/group/Vk6pDIfre6/contest/316095/problem/B) | [Еще одна игра со строкой](https://codeforces.com/group/Vk6pDIfre6/contest/316095/problem/B) |

После некоторых наблюдений мы видим, что игроки всегда должны выбирать наиболее значимую букву для изменения, потому что она больше всего координирует лексикографический порядок последней строки. Следовательно, Алиса выберет все нечетные индексы, в то время как Боб выберет все четные индексы, а затем Алиса изменит все буквы, которые она выберет, на минимально возможные буквы, а Боб изменит все буквы, которые он выберет, на максимально возможные буквы. То есть Алиса изменит буквы на «а», если исходная буква не «а», и на «б» в противном случае; Боб изменит буквы на «z», если исходная буква не «z», и на «y» в противном случае. Асимптотика O (n).

**Решение C++**

#include <bits/stdc++.h>

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

void solve**()**

**{**

string s**;**

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

**for** **(**int i **=** 0**;** i **<** s**.**size**();** **++**i**)**

**{**

**if** **(**i **%** 2 **==** 0**)**

**{**

s**[**i**]** **=** s**[**i**]** **==** 'a' **?** 'b' **:** 'a'**;**

**}**

**else**

**{**

s**[**i**]** **=** s**[**i**]** **==** 'z' **?** 'y' **:** 'z'**;**

**}**

**}**

cout **<<** s **<<** endl**;**

**}**

int main**()**

**{**

int tests**;**

cin **>>** tests**;**

**while** **(**tests**--)** solve**();**

**return** 0**;**

**}**

**Решение Python**

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

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

s**=**input**()**

s2**=**''

**for** j **in** range**(**len**(**s**)):**

**if** j**%**2**==**0**:**

**if** s**[**j**]==**'a'**:**s2**+=**'b'

**else:**s2**+=**'a'

**else:**

**if** s**[**j**]==**'z'**:**s2**+=**'y'

**else:**s2**+=**'z'

**print(**s2**)**

|  |  |
| --- | --- |
| [C](https://codeforces.com/group/Vk6pDIfre6/contest/316095/problem/C) | [Космическая навигация](https://codeforces.com/group/Vk6pDIfre6/contest/316095/problem/C) |
|  |  |

Подсказка 1. Вы можете думать об этой задаче как о двух независимых одномерных вопросах (один - вверх и вниз, а другой - влево и вправо) вместо одного двухмерного вопроса.

Подсказка 2: каков интервал позиций, которые вы можете достичь для каждой одномерной части, и посмотрите, находится ли конечная точка в этом интервале.

Подсказка 3: интервал вверх и вниз равен [-Счетчик D, Счетчик U], а интервал левого и правого - [-Счетчик L, Счетчик R].

**Решение C++**

#include <bits/stdc++.h>

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

int main**(){**

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

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

int x**,**y**;**cin**>>**x**>>**y**;**

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

int u**=**0**,**d**=**0**,**l**=**0**,**r**=**0**;**

**for(**int i**=**0**;**i**<**s**.**length**();**i**++){**

**if(**s**[**i**]==**'U'**)**u**++;**

**else** **if(**s**[**i**]==**'R'**)**r**++;**

**else** **if(**s**[**i**]==**'D'**)**d**++;**

**else** l**++;**

**}**

**if(**x **>** 0 **&&** r **>=** x **)**x **=** 0**;**

**if(**x **<** 0 **&&** l **>=** **-**x **)**x **=** 0**;**

**if(**y **>** 0 **&&** u **>=** y **)**y **=** 0**;**

**if(**y **<** 0 **&&** d **>=** **-**y **)**y **=** 0**;**

cout**<<((!**x **&&** **!**y**)?**"YES"**:**"NO"**)<<**endl**;**

**}**

**}**

**Решение Python**

**import** collections

tc **=** int**(**input**())**

**for** ti **in** range**(**tc**):**

x**,** y **=** **[**int**(**i**)** **for** i **in** input**().**split**()]**

str **=** collections**.**Counter**(**input**())**

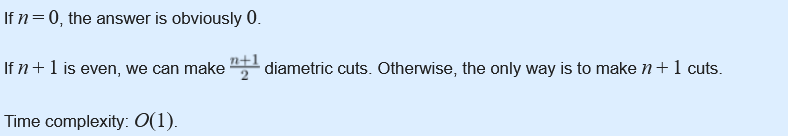
**if** **-**str**[**"D"**]** **<=** y **<=** str**[**"U"**]** **and** **-**str**[**"L"**]** **<=** x **<=** str**[**"R"**]:**

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

**else:**

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

|  |  |
| --- | --- |
| [D](https://codeforces.com/group/Vk6pDIfre6/contest/316095/problem/D) | [Пицца, пицца, пицца!!!](https://codeforces.com/group/Vk6pDIfre6/contest/316095/problem/D) |



**Решение C++**

#include <stdio.h>

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

long long n**;**

int main**()**

**{**

scanf**(**"%I64d"**,** **&**n**);**

n**++;**

**if** **(**n **==** 1**)** printf**(**"0"**);**

**else** **if** **(**n **%** 2 **==** 0**)**

printf**(**"%I64d"**,** n **/** 2**);**

**else** printf**(**"%I64d"**,** n**);**

**return** 0**;**

**}**

**Решение Python**

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

n**+=(**n**>**1**)**

**print(**n**//(**2**-(**n**%**2**)))**

|  |  |
| --- | --- |
| [E](https://codeforces.com/group/Vk6pDIfre6/contest/316095/problem/E) | [Алиса, Боб и шоколад](https://codeforces.com/group/Vk6pDIfre6/contest/316095/problem/E) |

Необходимо промоделировать описанную в условии игру. Имеем два указателя на начало и конец массива длин шоколадок, каждый раз смещаясь по первому, второму или обоим (в зависимости от длин шоколадок). Сместившись только по одному указателю, нужно уменьшить длину недоеденной шоколадки.

Единственная техническая трудность - правильно обработать последнюю шоколадку, можно ошибиться или зациклиться, если участники перейдут к ней одновременно.

**Решение C++**

#include<cstdio>

int n**,**a**[**100005**];**

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

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

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

scanf**(**"%d"**,&**a**[**i**]);**

a**[**i**]** **=** a**[**i**]+**a**[**i**-**1**];**

**}**

int i **=** 1**;**

**while(**a**[**i**]** **<=** a**[**n**]-**a**[**i**+**1**]** **&&** i **<** n**)** **{**

i**++;**

**}**

printf**(**"%d %d\n"**,**i**,**n**-**i**);**

**return** 0**;**

**}**

**Решение Python**

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

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

a **=** x **=** y **=** 0

b **=** n**-**1

**while** **(**a**<=**b**):**

**if(**x**<=**y**):**

x**+=**s**[**a**]**

a**+=**1

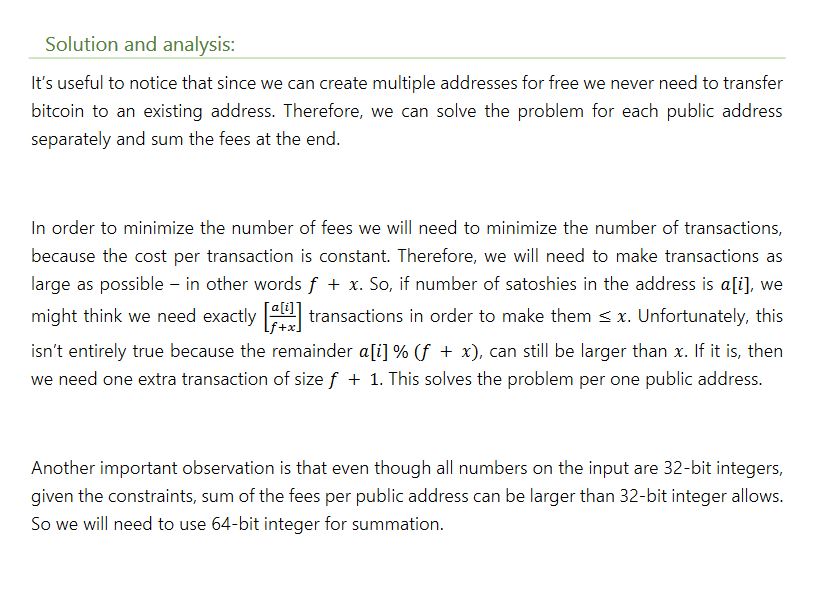
**else:**

y**+=**s**[**b**]**

b**-=**1

**print(**a**,** n**-**a**)**

|  |  |
| --- | --- |
| [F](https://codeforces.com/group/Vk6pDIfre6/contest/316095/problem/F) | [Дележ монет](https://codeforces.com/group/Vk6pDIfre6/contest/316095/problem/F) |

****

**Решение C++**

#include <iostream>

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

int a**[**200010**];**

int main**(){**

long long n**,**x**,**f**,**ans**=**0**;**

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

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

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

cin**>>**x**>>**f**;**

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

ans**+=(**a**[**i**]+**f**-**1**)/(**x**+**f**)\***f**;**

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

**}**

**Решение Python**

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

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

x**,** f **=** map**(**int**,** input**().**split**())**

s **=** 0

**for** y **in** a**:**

d **=** **(**y **-** x**)** **//** **(**x **+** f**)**

y **-=** d **\*** **(**x **+** f**)**

**if** y **>** x**:**

d **+=** 1

s **+=** d

**print(**s **\*** f**)**

|  |  |
| --- | --- |
| [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**)**