|  |  |
| --- | --- |
| [A](https://codeforces.com/group/Vk6pDIfre6/contest/320698/problem/A) | [Капитализация слова](https://codeforces.com/group/Vk6pDIfre6/contest/320698/problem/A) |

**Решение С++**

#include <iostream>

#include <string>

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

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

string s**;**

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

s**[**0**]** **=** toupper**(**s**[**0**]);**

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

**}**

**Решение Python**

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

**print(**s**[**0**].**upper**()+**s**[**1**:])**

|  |  |  |
| --- | --- | --- |
| [B](https://codeforces.com/group/Vk6pDIfre6/contest/320698/problem/B) |  | [Петя и строки](https://codeforces.com/group/Vk6pDIfre6/contest/320698/problem/B) |

В данной задаче сначала нужно было привести обе строки к одинаковому регистру (например, сделать все буквы маленькими), а затем просто сравнить их лексикографически, пройдя слева направо по символам.

**Решение С++**

#include <iostream>

#include <string>

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

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

string s**;**

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

s**[**0**]** **=** toupper**(**s**[**0**]);**

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

**}**

**Решение Python**

a **=** input**().**lower**()**

b **=** input**().**lower**()**

**print(**1 **if** a **>** b **else** **-**1 **if** a **<** b **else** 0**)**

|  |  |
| --- | --- |
| [C](https://codeforces.com/group/Vk6pDIfre6/contest/320698/problem/C) | [Перестановка строки](https://codeforces.com/group/Vk6pDIfre6/contest/320698/problem/C) |

Нужно проверить, что обе строки состоят из одинакового набора символов.

Проще всего отсортировать символы в каждом строке и сравнить строки.

**Решение С++**

#include <iostream>

#include <string>

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

int main**()**

**{**

string str**,**str1**;**

cin **>>** str**>>**str1**;**

sort**(**str**.**begin**(),** str**.**end**());**

sort**(**str1**.**begin**(),** str1**.**end**());**

**if** **(**str **==** str1**)**

cout **<<** "YES"**;**

**else** cout **<<** "NO"**;**

**}**

**Решение Python**

**print(**"YES" **if** sorted**(**input**())** **==** sorted**(**input**())** **else** "NO"**)**

|  |  |
| --- | --- |
| [D](https://codeforces.com/group/Vk6pDIfre6/contest/320698/problem/D) | [Контрольная сумма](https://codeforces.com/group/Vk6pDIfre6/contest/320698/problem/D) |

Можно явно решить задачу с помощью сортировки или структуры данных словарь (map)

Но есть хитрое решение за O(N). Воспользуемся побитовым оператором xor. Если мы применим xor к одинаковым числам, то они превратятся в ноль. А ноль с любым числом дает это число. И порядок действий не важен. Оператор xor чаще всего обозначается как ^

**Решение С++**

#include <iostream>

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

int main**()**

**{**

int n**,** a**;**

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

int b **=** 0**;**

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

cin **>>** a**;**

b **=** b **^** a**;**

**}**

cout **<<** b**;**

**}**

**Решение Python**

**from** functools **import** reduce

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

**print(**reduce**(lambda** x**,** y**:** x **^** y**,** lst**))**

|  |  |
| --- | --- |
| [E](https://codeforces.com/group/Vk6pDIfre6/contest/320698/problem/E) | [Ложные новости](https://codeforces.com/group/Vk6pDIfre6/contest/320698/problem/E) |

Воспользуемся методом двух указателей: пройдем циклом по строке одновременно сдвигая второй указатель в строке “heidi”. Если дошли до конца то ответ YES. O(N)

**Решение С++**

#include <iostream>

#include <string>

#include <algorithm>

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

int main**(){**

string s**,**p**=**"heidi"**;**

int j**=**0**;**

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

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

**if(**s**[**i**]==**p**[**j**]){**

j**++;**

**if(**j**==**5**)** **break;**

**}**

**}**

cout**<<(**j**==**5 **?** "YES" **:** "NO"**);**

**}**

**Решение Python**

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

j **=** 0

ss **=** 'heidi#'

**for** i **in** s**:**

**if** i **==** ss**[**j**]:**

j **+=** 1

**print** **(**'YES' **if** j **==** 5 **else** 'NO'**)**

|  |  |
| --- | --- |
| [F](https://codeforces.com/group/Vk6pDIfre6/contest/320698/problem/F) | [Порядок 2](https://codeforces.com/group/Vk6pDIfre6/contest/320698/problem/F) |

В данной задаче может показаться, что достаточно отсортировать числа как строки по возрастанию. Но это неверно, рассмотрим такой тест:

2

2 21

В данном случае 221 больше, чем 212.

Решение:

Пусть у нас есть массив 3 12 52 1

Склеим все в одну строку любым способом 312521.

Дальше будем много раз проходиться по массиву и пытаться обменять местами 2 числа. Если итоговая строка улучшается, то меняем. По сути, пузырьковая сортировка.

Еще короче: нужно проверять как выгоднее склеить a+b или b+a.

Можно добавить такое сравнение в сортировку (как компаратор)

**Решение С++**

#include <iostream>

#include <vector>

#include <string>

#include <algorithm>

using namespace std;

bool cmp(string a, string b) {

return a + b < b + a;

}

string mas[111];

int main()

{

int n;

cin >> n;

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

cin >> mas[i];

sort(mas, mas + n, cmp);

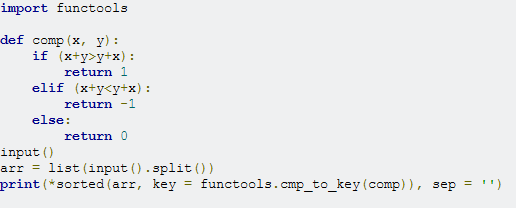
for (int i = 0; i < n; i++)

cout << mas[i];

return 0;

}

**Решение Python**



|  |  |
| --- | --- |
| [G](https://codeforces.com/group/Vk6pDIfre6/contest/320698/problem/G) | [Хеш строки](https://codeforces.com/group/Vk6pDIfre6/contest/320698/problem/G) |

Задача на реализацию

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <algorithm>

#include <iostream>

#include <sstream>

#include <string>

using namespace std;

uint32\_t calcHash(string s, uint32\_t p) {

uint32\_t res = 0;

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

res = res \* p + s[i];

return res;

}

int main() {

uint32\_t p;

cin >> p;

string s;

cin >> s;

cout << calcHash(s, p);

return 0;

}

|  |  |
| --- | --- |
| [H](https://codeforces.com/group/Vk6pDIfre6/contest/320698/problem/H) | [Hack hash](https://codeforces.com/group/Vk6pDIfre6/contest/320698/problem/H) |

Проще всего было воспользоваться парадоксом дней рождения. Если мы сгенерируем миллион случайных строк, то среди них почти со 100% вероятностью будет две с одинаковыми хешами.

#include <iostream>

#include <cstdio>

#include <cmath>

#include <string>

#include <algorithm>

#include <vector>

#include <map>

#include <sstream>

using namespace std;

#pragma comment(linker, "/STACK:20000000")

uint32\_t calcHash(string s, uint32\_t p) {

uint32\_t res = 0;

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

res = res \* p + s[i];

return res;

}

map <uint32\_t, string> mp;

void solve()

{

uint32\_t p;

cin >> p;

while (true)

{

string s = "";

int len = rand() % 30 + 1;

for (int i = 0; i < len; i++)

{

char c = 'a' + rand() % 26;

s += c;

}

uint32\_t h = calcHash(s, p);

if ((mp.find(h) != mp.end()) && mp[h] != s)

{

cout << s << endl;

cout << mp[h] << endl;

return;

}

mp[h] = s;

}

}

int main()

{

solve();

return 0;

}

Либо можно воспользоваться генератором теста, описанным здесь (для модулей равных степеней двойки)

<https://codeforces.com/blog/entry/4898?locale=ru>

|  |  |
| --- | --- |
| [I](https://codeforces.com/group/Vk6pDIfre6/contest/320698/problem/I) | [Кольцо](https://codeforces.com/group/Vk6pDIfre6/contest/320698/problem/I) |

Полиномиальный хеш.

abcb

a\*p^3 + b\*p^2 + c\*p^1 + b = ((a\*p + b)\*p + c)\*p + b

, где p достаточно большое простое число, например 3137

Считаем long long хеш с переполнением.

Как изменяется хеш при обмене двух элементов, например a и c

Вычитаем a\*p^3 и c\*p^1

Прибавляем с\*p^3 и a\*p^1

Как определять, эквивалентна ли текущая строка исходной.

Она должна быть подстрокой удвоенной исходной строки

abcbabcb

Посчитаем хеши для всех подстрок длины 4: abcb bcba cbab babc abcb

Это можно сделать за линейное время.

На каждом шаге запроса будем смотреть, есть ли у нас такой хеш.

Это можно сделать тремя способами

1. set или map. Это самый надежный. Но более медленный и требовательный к памяти. Сложность O(N\*logN)
2. Использовать обычный массив. Отсортировать и искать бинарным поиском.

Сложность O(N\*logN). Работает быстрее сета и требует минимум памяти.

1. Использовать свою хеш-таблицу. Самый быстрый способ. Сложность O(N). Требователен к памяти. Размер массива лучше выбирать в 10 раз больше количества возможных хешей. У хеша берем например 20 младших бит. И смотрим в таблице есть ли на этом месте требуемый хеш. Если место занято, то переходим на следующую позицию. Так идем пока не дойдем до пустого места.

#include <stdio.h>

#include <sstream>

#include <iostream>

#include <string>

#include <algorithm>

#include <vector>

#include <list>

#include <iomanip>

#include <map>

#include <set>

#include <cmath>

#include <queue>

#include <cassert>

#include <string.h>

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

#pragma comment(linker, "/STACK:20000000")

**typedef** vector**<**int**>** vi**;**

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

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

string problem\_name **=** "ring"**;**

void init**(){**

freopen**((**problem\_name**+**".in"**).**c\_str**(),**"rt"**,**stdin**);**

freopen**((**problem\_name**+**".out"**).**c\_str**(),**"wt"**,**stdout**);**

**}**

const int maxlen**=**100000**;**

long long hval **=** 3137**;**

long long p**[**2**\***maxlen**+**115**];**

char st**[**2**\***maxlen**+**115**];**

long long hmas**[**2**\***maxlen**+**115**];**

int len**;**

long long calc**(**string **&**s**)**

**{**

long long res**=**0**;**

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

res**=**res**\***hval**+(**long long**)**s**[**i**];**

**return** res**;**

**}**

string s**;**

long long ht**[(**1**<<**20**)+**5**];**

void addhash**(**long long val**)**

**{**

long long p **=** val**&((**1**<<**20**)-**1**);**

**while** **(**ht**[**p**])**

**{**

p**++;**

**if** **(**p**>=(**1**<<**20**))** p **=** 0**;**

**}**

ht**[**p**]** **=** val**;**

**}**

int findhash**(**long long val**)**

**{**

long long p **=** val**&((**1**<<**20**)-**1**);**

**while** **(**ht**[**p**])**

**{**

**if** **(**ht**[**p**]==**val**)** **return** 1**;**

p**++;**

**if** **(**p**>=(**1**<<**20**))** p **=** 0**;**

**}**

**return** 0**;**

**}**

long long gethash**(**int l**,** int r**)** // получаем значение на отрезке от l до r

**{**

**return** hmas**[**r**]** **-** hmas**[**l**-**1**]\***p**[**r**-**l**+**1**];**

**}**

long long f**(**long long h**,**int a**,** int b**)** // обмен

**{**

h**-=**s**[**a**]\***p**[**len**-**a**-**1**];**

h**-=**s**[**b**]\***p**[**len**-**b**-**1**];**

h**+=**s**[**a**]\***p**[**len**-**b**-**1**];**

h**+=**s**[**b**]\***p**[**len**-**a**-**1**];**

**return** h**;**

**}**

int main**()**

**{**

init**();**

//srand(5);

p**[**0**]=**1**;**

**for** **(**int i**=**1**;**i**<**100100**;**i**++)**

p**[**i**]=**p**[**i**-**1**]\***hval**;**

gets**(**st**);**

len**=**strlen**(**st**);**

s **=** st**;**

s**+=**s**;**

memset**(**ht**,**0**,sizeof(**ht**));**

long long h**=**0**;**

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

h**=**h**\***hval **+** s**[**i**];**

addhash**(**h**);**

long long cur **=** h**;**

hmas**[**0**]=**s**[**0**];**

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

hmas**[**i**]=**hmas**[**i**-**1**]\***hval**+**s**[**i**];**

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

addhash**(**gethash**(**i**-**len**+**1**,**i**));**

int n**;**

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

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

**{**

int a**,**b**;**

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

a**--;**

b**--;**

cur **=** f**(**cur**,**a**,**b**);**

swap**(**s**[**a**],**s**[**b**]);**

**if** **(**findhash**(**cur**))**

printf**(**"YES\n"**);** **else**

printf**(**"NO\n"**);**

**}**

**return** 0**;**

**}**

|  |  |
| --- | --- |
| [J](https://codeforces.com/group/Vk6pDIfre6/contest/320698/problem/J) | [Тест](https://codeforces.com/group/Vk6pDIfre6/contest/320698/problem/J) |

Можно решать через алгоритм **КМП** или **хешированием**.

Решение хешированием.   
Пусть входные строки *s*0, *s*1, *s*2. Постараемся найти кратчайшую строку, которая содержит *s*0, *s*1, *s*2. Пройдём все порядки *s*0, *s*1, *s*2 и поищем длиннейшее перекрытие в конце строки *a* и начале строки *b*. Польный перебор конечно требует слишком много времени. С другой стороны, хеширование может решить за *O*(*n*) операции, где *n* = *min*(*len*(*a*), *len*(*b*)). Моя хеш-функция - полином *hash*(*x*0, *x*1, ..., *xn*) = *x*0 + *ax*1 + *a*2*x*2 + ... + *anxn*. Этот полином удобный в этой задаче потому что он имеет следующее свойство:  
Если известно *hash*(*xi*, ..., *xj*), тогда можно за *O*(1) посчитать следующие значения:

* *hash*(*xi*- 1, *xi*, ..., *xj*) = *xi*- 1 + *a* × *hash*(*xi*, ..., *xj*)
* *hash*(*xi*, ..., *xj*, *xj*+ 1) = *hash*(*xi*, ..., *xj*) + *aj*+ 1 -*i* × *xj*+ 1

Т.е., если известно значение хеш-функции какой-то подстроки, легко посчитать значение соседних подстрок. Для строк *a*, *b*, посчитаем значения хеш-функций подстрок в конце *a* и в начале *b*. Если они равные для подстрок размера *n*, тогда значит, что (может быть) есть дублирование *n* характеров в *a* и *b*.  
  
Поэтому, пройдём все порядки *s*0, *s*1, *s*2, и попробуем связать строки вместе. Осталась одна проблема --- если *si* подстрока *sj* и *i* ≠ *j*, тогда можно пропустить *si*. Используем хеширование быстро решить, *si* подстрока *sj* или нет.

**Решение С++**

#include<iostream>

#include<cstdio>

#include<cstring>

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

#define N 200020

#define NN 100010

char str**[**N**];**

char s**[**3**][**NN**];**

int p**[**N**];**

int kmp**(**char **\***s1**,**char **\***s2**)**

**{**

int n**=**strlen**(**s1**);**

int m**=**strlen**(**s2**);**

int j**=-**1**;**

p**[**0**]=-**1**;**

**for(**int k**=**1**;**k**<**m**;**k**++)**

**{**

**while(**j**>=**0**&&**s2**[**k**]!=**s2**[**j**+**1**])** j**=**p**[**j**];**

**if(**s2**[**k**]==**s2**[**j**+**1**])** j**++;**

p**[**k**]=**j**;**

**}**

j**=-**1**;**

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

**{**

**if(**j**==**m**-**1**)** **return** 0**;**

**while(**j**>=**0**&&**s1**[**k**]!=**s2**[**j**+**1**])** j**=**p**[**j**];**

**if(**s1**[**k**]==**s2**[**j**+**1**])** j**++;**

**}**

**return** **(**m**-**1**-**j**);**

**}**

int compair**(**char **\***s1**,**char **\***s2**,**char **\***s3**)**

**{**

int len**=**kmp**(**s1**,**s2**);**

str**[**0**]=**'\0'**;**

strcat**(**str**,**s1**);**

strcat**(**str**,**s2**+**strlen**(**s2**)-**len**);**

len**=**kmp**(**str**,**s3**);**

**return** strlen**(**str**)+**len**;**

**}**

int main**()**

**{**

**while(**scanf**(**"%s%s%s"**,**s**[**0**],**s**[**1**],**s**[**2**])!=**EOF**)**

**{**

int minn**;**

int l**;**

minn**=**compair**(**s**[**0**],**s**[**1**],**s**[**2**]);**

l**=**compair**(**s**[**0**],**s**[**2**],**s**[**1**]);**

**if(**l**<**minn**)** minn**=**l**;**

l**=**compair**(**s**[**1**],**s**[**2**],**s**[**0**]);**

**if(**l**<**minn**)** minn**=**l**;**

l**=**compair**(**s**[**1**],**s**[**0**],**s**[**2**]);**

**if(**l**<**minn**)** minn**=**l**;**

l**=**compair**(**s**[**2**],**s**[**0**],**s**[**1**]);**

**if(**l**<**minn**)** minn**=**l**;**

l**=**compair**(**s**[**2**],**s**[**1**],**s**[**0**]);**

**if(**l**<**minn**)** minn**=**l**;**

cout**<<**minn**<<**endl**;**

**}**

**}**

**Решение Python**

**def** p**(**a**,**b**):**

s**,**m**,**c**,**j**=**b**+**'#'**+**a**,**0**,**0**,**0**;**p**=[**0**]\***len**(**s**)**

**for** i **in** range**(**1**,**len**(**s**)):**

**while** j **and** s**[**i**]!=**s**[**j**]:** j**=**p**[**j**-**1**]**

**if** s**[**i**]==**s**[**j**]:** j**+=**1

p**[**i**]=**j

**if** j**==**len**(**b**):** **return** a

**return** a**[:**len**(**a**)-**p**[-**1**]]+**b

s**=[**input**()** **for** \_ **in** ' '**]**

**print(**min**(**len**(**p**(**s**[**x**[**0**]],**p**(**s**[**x**[**1**]],**s**[**x**[**2**]])))** **for** x **in** \_\_import\_\_**(**'itertools'**).**permutations**([**0**,**1**,**2**])))**