## A: Connection among Cities

You are given a list of cities. Each direct connection between two cities has its transportation cost (an integer bigger than 0). The goal is to find the paths of minimum cost between pairs of cities. Assume that the cost of each path (which is the sum of costs of all direct connections belonging to this path) is at most 200000. The name of a city is a string containing characters a,...,z and is at most 10 characters long.

**Input:**

s [the number of tests <= 10]
n [the number of cities <= 10000]
NAME [city name]
p [the number of neighbours of city NAME]
nr cost [nr - index of a city connected to NAME (the index of the first city is 1)]
[cost - the transportation cost]
r [the number of paths to find <= 100]
NAME1 NAME2 [NAME1 - source, NAME2 - destination]
[empty line separating the tests]

**Output:**

cost [the minimum transportation cost from city NAME1 to city NAME2 (one per line)]

| Sample Input: | Sample Output: |
|---|---|
| 1<br>4<br>gdansk<br>2<br>2 1<br>3 3<br>bydgoszcz<br>3<br>1 1<br>3 1<br>4 4<br>torun<br>3<br>1 3<br>2 1<br>4 1<br>warszawa<br>2<br>2 4 | 3<br>2 |

$O(V+E\log V)$

| | |
|---|---|
| 3 1<br>2<br>gdansk warszawa<br>bydgoszcz warszawa | |

## Problem B: Palindrome

You are given an array a consisting of n integers.Your task is to determine if a has some subsequence of length at least 3 that is a palindrome.

Recall that an array b is called a subsequence of the array a if b can be obtained by removing some (possibly, zero) elements from a (not necessarily consecutive) without changing the order of remaining elements. For example, [2], [1,2,1,3] and [2,3] are subsequences of [1,2,1,3], but [1,1,2] and [4] are not.Also, recall that a palindrome is an array that reads the same backward as forward. In other words, the array a of length n is the palindrome if $a_i = a_{n-i-1}$ for all i from 1 to n. For example, arrays [1234], [1,2,1], [1,3,2,2,3,1] and [10,100,10] are palindromes, but arrays [1,2] and [1,2,3,1] are not.You have to answer t independent test cases.

**Input:**

The first line of the input contains one integer t ($1 \le t \le 100$) — the number of test cases.

Next 2t lines describe test cases. The first line of the test case contains one integer n ($3 \le n \le 5000$) — the length of a. The second line of the test case contains n integers $a_1, a_2, \ldots a_n$ ($1 \le a_n \le n$), where $a_i$ is the i-th element of a.

It is guaranteed that the sum of n over all test cases does not exceed 5000 ($\Sigma n \le 5000$).

**Output**

For each test case, print the answer — "YES" (without quotes) if a has some subsequence of length at least 3 that is a palindrome and "NO" otherwise.

| Sample Input | Sample Output |
|---|---|
| 5<br>3<br>1 2 1 ( YES<br>5<br>1 2 2 3 2<br>3<br>1 1 2<br>4<br>1 2 2 1 (YES)<br>10<br>1 1 2 2 3 3 4 4 5 5 | YES<br>YES<br>NO<br>YES<br>NO |

5 X 10^6

## Question-1

A function abcd() is defined. Which algorithm is implemented in the function abcd()? What kind of problems can be solved by the function?

```
#define pii pair<int,int>
int fx[]={1,-1,0,0};
int fy[]={0,0,1,-1};
int cell[100][100];
int d[100][100],vis[100][100];
int row,col;
void abcd(int sx,int sy)
{
	memset(vis,0,sizeof( vis));
	vis[sx][sy]=1;
	queue<pii>q;
	q.push(pii(sx,sy));
	while(!q.empty())
	{
		pii top=q.front(); q.pop();
		for(int k=0;k<4;k++)
		{
			int tx=top.uu+fx[k];
			int ty=top.vv+fy[k];
			if(tx>=0 and tx<row and ty>=0 and ty<col and cell[tx][ty]!=-1 and vis[tx][ty]==0)
			{
				vis[tx][ty]=1;
				d[tx][ty]=d[top.uu][top.vv]+1;
				q.push(pii(tx,ty));
			}
		}
	}
}
```

## Question-2

Prepare three datasets for the best case, average case and the worst case of Dijkstra Algorithm. Explain why the datasets are belonging the best case, average case and the worst case.

## Question-3

Adilbek was assigned to a special project. For Adilbek it means that he has n days to run a special program and provide its results. But there is a problem: the program needs to run for d days to calculate the results.

Fortunately, Adilbek can optimize the program. If he spends x (x is a non-negative integer) days optimizing the program, he will make the program run in $\lceil \frac{d}{x+1} \rceil$ days ($\lceil a \rceil$ is the ceiling function: $\lceil 2.4 \rceil = 3$, $\lceil 2 \rceil = 2$). The program cannot be run and optimized simultaneously, so the total number of days he will spend is equal to $x + \lceil \frac{d}{x+1} \rceil$.

Will Adilbek be able to provide the generated results in no more than n days? You can cooperate Adilbek. In that case your task is to provide a solution or algorithm. You should not write any program for Adilbek.

**Input**

The first line contains a single integer T ($1 \le T \le 50$) — the number of test cases. The next T lines contain test cases – one per line. Each line contains two integers n and d ($1 \le n \le 10^9$, $1 \le d \le 10^9$) — the number of days before the deadline and the number of days the program runs.

**Output**

Print T answers — one per test case. For each test case print YES (case insensitive) if Adilbek can fit in n days or NO (case insensitive) otherwise.

| Sample Input | Sample Output |
|---|---|
| 3<br>1 1<br>4 5<br>5 11 | YES<br>YES<br>NO |

## Question 1

The basic convex hull algorithm is fairly interesting; however, you may have noticed that you can not draw the hull until all of the points have been specified. Is there any interactive way to add points to the hull and redraw it while maintaining an optimal time complexity?

## Question 2

Dmitry has a string s, consisting of lowercase Latin letters. Dmitry decided to remove two consecutive characters from the string s and you are wondering how many different strings can be obtained after such an operation.

For example, Dmitry has a string "aaabcc". You can get the following different strings: "abcc"(by deleting the first two or second and third characters), "aacc"(by deleting the third and fourth characters),"aaac"(by deleting the fourth and the fifth character) and "aaab" (by deleting the last two).

**Input**

The first line of input data contains a single integer t ($1 \le t \le 10^4$) — number of test cases. The descriptions of the test cases follow. The first line of the description of each test case contains an integer n ($3 \le n \le 2 \cdot 10^5$). The second line of the description of each test case contains a string s of length n consisting of lowercase Latin letters. It is guaranteed that the sum of n for all test cases does not exceed $2 \cdot 10^5$.

**Output**

For each test case print one integer — the number of distinct strings that can be obtained by removing two consecutive letters. Write an algorithm to solve the problem.

| Sample Input | Sample Output |
|---|---|
| 7<br>6<br>aaabcc<br>10<br>aaaaaaaaaa<br>6<br>abcdef<br>7<br>abacaba<br>6<br>cccfff | 4<br>1<br>5<br>3<br>3<br>3<br>1 |

| | |
|---|---|
| 4<br>abba<br>5<br>ababa | |

## Question 3

Can two different patterns in a text generate the same hash value in Rabin Karp algorithm? How can you prevent generating the same hash value from two different patterns?