Treino de iniciantes #11

A. Remove Smallest

1 second, 256 megabytes

You are given the array a consisting of n positive (greater than zero) integers.

In one move, you can choose two indices i and j ($i \neq j$) such that the absolute difference between a_i and a_j is no more than one ($|a_i - a_j| \leq 1$) and remove the smallest of these two elements. If two elements are equal, you can remove any of them (but exactly one).

Your task is to find if it is possible to obtain the array consisting of **only one element** using several (possibly, zero) such moves or not.

You have to answer t independent test cases.

Input

The first line of the input contains one integer t ($1 \leq t \leq 1000$) — the number of test cases. Then t test cases follow.

The first line of the test case contains one integer n ($1 \leq n \leq 50$) — the length of a. The second line of the test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 100$), where a_i is the i-th element of a.

Output

For each test case, print the answer: "YES" if it is possible to obtain the array consisting of **only one element** using several (possibly, zero) moves described in the problem statement, or "NO" otherwise.

input

```
5
3
1 2 2
4
5 5 5 5
3
1 2 4
4
1 3 4 4
1
100
```

output

```
YES
YES
NO
NO
YES
```

In the first test case of the example, we can perform the following sequence of moves:

- choose $i = 1$ and $j = 3$ and remove a_i (so a becomes $[2; 2]$);
- choose $i = 1$ and $j = 2$ and remove a_j (so a becomes $[2]$).

In the second test case of the example, we can choose any possible i and j any move and it doesn't matter which element we remove.

In the third test case of the example, there is no way to get rid of 2 and 4.

B. Two Arrays And Swaps

1 second, 256 megabytes

You are given two arrays a and b both consisting of n positive (greater than zero) integers. You are also given an integer k.

In one move, you can choose two indices i and j ($1 \leq i, j \leq n$) and swap a_i and b_j (i.e. a_i becomes b_j and vice versa). Note that i and j can be equal or different (in particular, swap a_2 with b_2 or swap a_3 and b_9 both are acceptable moves).

Your task is to find the **maximum** possible sum you can obtain in the array a if you can do no more than (i.e. at most) k such moves (swaps).

You have to answer t independent test cases.

Input

The first line of the input contains one integer t ($1 \leq t \leq 200$) — the number of test cases. Then t test cases follow.

The first line of the test case contains two integers n and k ($1 \leq n \leq 30; 0 \leq k \leq n$) — the number of elements in a and b and the maximum number of moves you can do. The second line of the test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 30$), where a_i is the i-th element of a. The third line of the test case contains n integers $b_1, b_2, \ldots, b_n$ ($1 \leq b_i \leq 30$), where b_i is the i-th element of b.

Output

For each test case, print the answer — the **maximum** possible sum you can obtain in the array a if you can do no more than (i.e. at most) k swaps.

input

```
5
2 1
1 2
3 4
5 5
5 5 6 6 5
1 2 5 4 3
5 3
1 2 3 4 5
10 9 10 10 9
4 0
2 2 4 3
2 4 2 3
4 4
1 2 2 1
4 4 5 4
```

output

```
6
27
39
11
17
```

In the first test case of the example, you can swap $a_1 = 1$ and $b_2 = 4$, so $a = [4, 2]$ and $b = [3, 1]$.

In the second test case of the example, you don't need to swap anything.

In the third test case of the example, you can swap $a_1 = 1$ and $b_1 = 10$, $a_3 = 3$ and $b_3 = 10$ and $a_2 = 2$ and $b_4 = 10$, so $a = [10, 10, 10, 4, 5]$ and $b = [1, 9, 3, 2, 9]$.

In the fourth test case of the example, you cannot swap anything.

In the fifth test case of the example, you can swap arrays a and b, so $a = [4, 4, 5, 4]$ and $b = [1, 2, 2, 1]$.

C. Honest Coach

2 seconds, 256 megabytes

There are n athletes in front of you. Athletes are numbered from 1 to n from left to right. You know the strength of each athlete — the athlete number i has the strength s_i.

You want to split all athletes into two teams. Each team must have at least one athlete, and each athlete must be exactly in one team.

You want the strongest athlete from the first team to differ as little as possible from the weakest athlete from the second team. Formally, you want to split the athletes into two teams A and B so that the value $|\max(A) - \min(B)|$ is as small as possible, where $\max(A)$ is the maximum strength of an athlete from team A, and $\min(B)$ is the minimum strength of an athlete from team B.

For example, if $n = 5$ and the strength of the athletes is $s = [3, 1, 2, 6, 4]$, then one of the possible split into teams is:

- first team: $A = [1, 2, 4]$,
- second team: $B = [3, 6]$.

In this case, the value $|\max(A) - \min(B)|$ will be equal to $|4 - 3| = 1$. This example illustrates one of the ways of optimal split into two teams.

Print the minimum value $|\max(A) - \min(B)|$.

Input
The first line contains an integer t ($1 \le t \le 1000$) — the number of test cases in the input. Then t test cases follow.

Each test case consists of two lines.

The first line contains positive integer n ($2 \le n \le 50$) — number of athletes.

The second line contains n positive integers $s_1, s_2, \ldots, s_n$ ($1 \le s_i \le 1000$), where s_i — is the strength of the i-th athlete. Please note that s values may not be distinct.

Output
For each test case print one integer — the minimum value of $|\max(A) - \min(B)|$ with the optimal split of all athletes into two teams. Each of the athletes must be a member of exactly one of the two teams.

input

```
5
5
3 1 2 6 4
6
2 1 3 2 4 3
4
7 9 3 1
2
1 1000
3
100 150 200
```

output

```
1
0
2
999
50
```

The first test case was explained in the statement. In the second test case, one of the optimal splits is $A = [2, 1]$, $B = [3, 2, 4, 3]$, so the answer is $|2 - 2| = 0$.

D. Nastya and an Array

1 second, 256 megabytes

Nastya owns too many arrays now, so she wants to delete the least important of them. However, she discovered that this array is magic! Nastya now knows that the array has the following properties:

- In one second we can add an arbitrary (possibly negative) integer to all elements of the array that are not equal to zero.
- When all elements of the array become equal to zero, the array explodes.

Nastya is always busy, so she wants to explode the array as fast as possible. Compute the minimum time in which the array can be exploded.

Input
The first line contains a single integer n ($1 \le n \le 10^5$) — the size of the array.

The second line contains n integers $a_1, a_2, ..., a_n$ ($-10^5 \le a_i \le 10^5$) — the elements of the array.

Output
Print a single integer — the minimum number of seconds needed to make all elements of the array equal to zero.

input

```
5
1 1 1 1 1
```

output

```
1
```

input

```
3
2 0 -1
```

output

```
2
```

input

```
4
5 -6 -5 1
```

output

```
4
```

In the first example you can add -1 to all non-zero elements in one second and make them equal to zero.

In the second example you can add -2 on the first second, then the array becomes equal to $[0, 0, -3]$. On the second second you can add 3 to the third (the only non-zero) element.

E. The New Year: Meeting Friends

1 second, 256 megabytes

There are three friend living on the straight line Ox in Lineland. The first friend lives at the point x_1, the second friend lives at the point x_2, and the third friend lives at the point x_3. They plan to celebrate the New Year together, so they need to meet at one point. What is the minimum total distance they have to travel in order to meet at some point and celebrate the New Year?

It's guaranteed that the optimal answer is always integer.

Input
The first line of the input contains three **distinct** integers x_1, x_2 and x_3 ($1 \le x_1, x_2, x_3 \le 100$) — the coordinates of the houses of the first, the second and the third friends respectively.

Output
Print one integer — the minimum total distance the friends need to travel in order to meet together.

input
7 1 4
output
6

input
30 20 10
output
20

In the first sample, friends should meet at the point 4. Thus, the first friend has to travel the distance of 3 (from the point 7 to the point 4), the second friend also has to travel the distance of 3 (from the point 1 to the point 4), while the third friend should not go anywhere because he lives at the point 4.

F. Special Elements

1 s., 64 MB

Pay attention to the non-standard memory limit in this problem.

Hint: Search about Prefix Sums.

In order to cut off efficient solutions from inefficient ones in this problem, the time limit is rather strict. Prefer to use compiled statically typed languages (e.g. C++). If you use Python, then submit solutions on PyPy. Try to write an efficient solution.

The array $a = [a_1, a_2, \ldots, a_n]$ $(1 \le a_i \le n)$ is given. Its element a_i is called special if there exists a pair of indices l and r $(1 \le l < r \le n)$ such that $a_i = a_l + a_{l+1} + \ldots + a_r$. In other words, an element is called special if it can be represented as the sum of **two or more consecutive elements** of an array (no matter if they are special or not).

Print the number of special elements of the given array a.

For example, if $n = 9$ and $a = [3, 1, 4, 1, 5, 9, 2, 6, 5]$, then the answer is 5:

- $a_3 = 4$ is a special element, since $a_3 = 4 = a_1 + a_2 = 3 + 1$;
- $a_5 = 5$ is a special element, since $a_5 = 5 = a_2 + a_3 = 1 + 4$;
- $a_6 = 9$ is a special element, since $a_6 = 9 = a_1 + a_2 + a_3 + a_4 = 3 + 1 + 4 + 1$;
- $a_8 = 6$ is a special element, since $a_8 = 6 = a_2 + a_3 + a_4 = 1 + 4 + 1$;
- $a_9 = 5$ is a special element, since $a_9 = 5 = a_2 + a_3 = 1 + 4$.

Please note that some of the elements of the array a may be equal — if several elements are equal and special, then all of them should be counted in the answer.

Input
The first line contains an integer t $(1 \le t \le 1000)$ — the number of test cases in the input. Then t test cases follow.

Each test case is given in two lines. The first line contains an integer n ($1 \le n \le 8000$) — the length of the array a. The second line contains integers $a_1, a_2, \ldots, a_n$ $(1 \le a_i \le n)$.

It is guaranteed that the sum of the values of n for all test cases in the input does not exceed 8000.

Output
Print t numbers — the number of special elements for each of the given arrays.

input
5 9 3 1 4 1 5 9 2 6 5 3 1 1 2 5 1 1 1 1 1 8 8 7 6 5 4 3 2 1 1 1
output
5 1 0 4 0