Java 1D Array (Part 2)

You are playing a game on your cell phone. You are given an array of length n, indexed from 0 to $n-1$. Each element of the array is either 0 or 1. You can only move to an index which contains 0. At first, you are at the 0^{th} position. In each move you can do one of the following things:

- Walk one step forward or backward.
- Make a jump of exactly length m forward.

That means you can move from position x to $x+1$, $x-1$ or $x+m$ in one move, but at least one of the following conditions must be true:

- The new position contains 0.
- The new position is greater than $n-1$.

You can't move backward from position 0. *If you move to any position greater than $n-1$, you win the game.*

Given the array and the length of the jump, you need to determine if it's possible to win the game or not.

Input Format

In the first line there will be an integer T denoting the number of test cases. Each test case will consist of two lines. The first line will contain two integers, n and m. On the second line there will be n space-separated integers, each of which is either 0 or 1.

Constraints:

- $1 \leq T \leq 5000$
- $2 \leq n \leq 100$
- $0 \leq m \leq 100$
- The first integer of the array is always 0.

Output Format

For each case output `YES` if it's possible to win the game, output `NO` otherwise.

Sample Input

```
4
5 3
0 0 0 0 0
6 5
0 0 0 1 1 1
6 3
0 0 1 1 1 0
3 1
0 1 0
```

Sample Output

```
YES
```

```
YES
NO
NO
```

Explanation

In the first case, you can just walk to reach the end of the array.
In the second case, you can walk to index 1 or 2 and jump from there. In the third case, jump length is too low, and you can't reach the end of the array. In the fourth case, jump length is 1, so it doesn't matter if you jump or walk, you can't reach the end.