1120 No Change

Though it might be hard to imagine, the inhabitants of a small country Additivia do not know of such thing as change, which probably has to do with them not knowing subtraction either. When they buy something, they always need to have the exact amount of addollars, their currency. The only other option, but not a really attractive one, is over-paying.

Professor Adem, one of the Additivian mathematicians came up with an algorithm for keeping a balanced portfolio. The idea is the following. Suppose you have more coins of value v_1 than coins of value v_2. In this case you should try to spend at least as many coins of value v_1 as those of value v_2 on any buy you make. Of course spending too many v_1 coins is not a good idea either, but to make the algorithm simpler professor Adem decided to ignore the problem. The algorithm became an instant hit and professor Adem is now designing a kind of "electronic portfolio" with built-in Adem's algorithm. All he needs now is a software for these machines, that will decide whether a given amount of addollars can be paid using a given set of coins according to the rules of Adem's algorithm. Needless to say, you are his chosen programmer for the task.

Write a program that reads the description of a set of coins and an amount of addollars to be paid, and determines whether you can pay that amount according to Professor Adem's rules.

Input

The input begins with a single positive integer on a line by itself indicating the number of the cases following, each of them as described below. This line is followed by a blank line, and there is also a blank line between two consecutive inputs.

The input starts with the amount of addollars to be paid x, where $1 \leq x \leq 100000$. The number of different coin values k follows, where $1 \leq k \leq 5$. The values of the coins $v_1, \ldots, v_k$ follow, where $1 \leq v_i \leq 10000$.

Notice that the order among coin values is significant: you need to spend at least as many coins of value v_1 as coins of value v_2, at least as many coins of value v_2 as those of value v_3, and so on. You may assume that you have a sufficiently large number of coins of each value.

Output

For each test case, the output must follow the description below. The outputs of two consecutive cases will be separated by a blank line.

Your program should output either a single word '`YES`', if the given amount can be paid according to the rules, or a single word '`NO`' otherwise.

Sample Input

```
1

13 3 9 2 1
```

Sample Output

```
NO
```