**Array**

Given **N** planets numbered from **1 to N**. A skrull at planet **1**, jumps from planet to planet in multiples of Aj (Aj, 2Aj, 3Aj).  
j is specific to the skrull. Whenever a skrull reaches a planet, it makes a small colony there. You have to find out how many planets, from **1 to N**, are left not colonised after all**K** skrulls have reached the end. Each skrull has its own jump factor denoted by **Aj**, and each skrull starts at planet number 1.

**Input:**  
The first line consists of a integer**T** denoting the number of testcases. **T** test cases follow. Each test case consists of two lines of input. The first line consists of two integers: **N,** which denotes the number of planets; and **K,**which denotes the number of skrulls. Second line of each test case consists of **K** space separated integers denoting the jumblip factor of skrulls.

**Output:**  
For each testcase, in a new line, print a single integer denoting the number of planets which  are left not colonized.

**Constraints:**  
1 <= T <= 100  
1 <= N <= 107  
1 <= K <= 107  
1 <= Aj <= 100

**If the Input is:**

7

6 5 8 4 7 10 9

**What is the output?**

**Stacks queues**

There are a number of plants in Asgard. Each of these plants has been treated with some amount of pesticide. After each day, if any plant has more pesticide than the plant on its left, being weaker than the left one, it dies.

You are given the initial values of the pesticide in each of the plants. Print the number of days after which no plant dies, i.e. the time after which there are no plants with more pesticide content than the

plant to their left.

For example, pesticide levels . Using a -indexed array, day  plants  and  die leaving . On day , plant  of the current array dies leaving . As there is no plant with a higher concentration of pesticide than the one to its left, plants stop dying after day .

**Function Description**  
Complete the function *poisonousPlants* in the editor below. It must return an integer representing the number of days until plants no longer die from pesticide.

poisonousPlants has the following parameter(s):

* *p*: an array of integers representing pesticide levels in each plant

**Input Format**

The first line contains an integer , the size of the array .  
The next line contains  space-separated integers .

**If the Input is:**

7

6 5 8 4 7 10 9

**What is the Output?**

**Trees**

Of course, Titan has the shape of a [tree](https://en.wikipedia.org/wiki/Tree_(graph_theory)) — there are **N** servers and **N-1** bidirectional channels connecting pairs of servers in such a way that there is exactly one path between each pair of servers.

Each Titan citizen lives on exactly one server at any point in time. Each channel connecting two servers has a value called *blip*. Each citizen can move to a different server through channels; moving through a channel with blip **2w** takes **w** nanoseconds. Initially, there are **ci** citizens living on the **i**-th server (for each 1 ≤ **i** ≤ **N**). Recently, Thanos has created an [Intitial Currency O](https://en.wikipedia.org/wiki/Initial_coin_offering)ffering  for Titancoin — the cryptocurrency of the future in Titan. Everyone in Titan should receive exactly 1 Titancoin. How to organize this process? Titancoins can only be received on certain servers containing cryptoexchanges. Unfortunately, Thanos can only create cryptoexchanges on two servers, since they consume a lot of electric power. The citizens want to receive Titancoin as fast as possible; therefore, each citizen chooses the closest server with a cryptoexchange and moves to that server. Receiving Titancoin at a cryptoexchange takes zero time. Thanos wants to minimize the sum of the times each citizen needs to spend moving to receive Titancoin. The two servers containing cryptoexchanges can be chosen arbitrarily. Compute the minimum total time it takes Titan's citizens to receive Titancoin!

**Input**

* The first line of the input contains a single integer **T** denoting the number of test cases. The description of **T** test cases follows.
* The first line of each test case contains a single integer **N**.
* The second line contains **N** space-separated integers **c1, c2, ..., cN**.
* **N-1** lines follow. Each of these lines contains three space-separated integers **u**, **v** and **w** denoting a channel with blip **2w** nanoseconds connecting servers **u** and **v**.

**Output**

For each test case, print a single line containing one integer — the minimal sum of times it takes each citizen to receive Titancoin if the two cryptoexchanges are placed optimally.

**Constraints**

* 2 ≤ **N** ≤ 50000
* 1 ≤ **ci** ≤ 50000 for each valid **i**
* 1 ≤ **w** ≤ 50000
* 1 ≤ **u**, **v** ≤ **N**
* the channels describe a tree
* 1 ≤ sum of **N** over all test cases ≤ 250000

**If the Input is :**

2

4

10 25 15 40

1 2 1

2 3 3

1 4 4

7

24786 24640 17641 19735 8667 15790 5948

3 7 17108

7 6 4875

7 1 9317

1 2 8597

3 4 28200

1 5 22026

**What is the sum of the digits in the first line in the output?**

**Graph**

A group of connected 1's forms a spacecraft. The task is to complete the method **findSpacecrafts()** which returns the **number of spacecrafts** present. The function takes three arguments the first is the boolean matrix **A** and then the next two arguments are **N** and **M** denoting the size(N\*M) of the matrix A .

**Input:**  
The first line of input will be the number of testcases **T**, then T test cases follow. The first line of each testcase contains two space separated integers N and M. Then in the next line are the NxM inputs of the matrix A separated by space .

**Output:**  
For each testcase in a new line, print the number of spacecrafts present.

**Constraints:**  
1 <= T <= 100  
1 <= N, M <= 50  
0 <= A[i][j] <= 1

**If the Input is:**  
2  
3 3  
1 1 0 0 0 1 1 0 1  
4 4  
1 1 0 0 0 0 1 0 0 0 0 1 0 1 0 0

**Which digit occurs two times in the Output?**

**Dynamic**

Iron man is man of equality. He needs your help to divide his “powers” evenly amongst all his followers. By doing this, Iron man can create equality amongst his followers and he'll be called a true “poweri”.

Iron man has **N** powers, and **K** followers. Each power is given a numerical value which shows its intensity.

Your task is to determine whether it is possible to allocate all the powers to followers in such a way that the sum of intensities of the powers allocated to each follower is equal. **Note :**A power can be allocated to only one of the followers.

**Input**

The first line of the input contains an integer **T**, denoting the number of test cases. Then **T** test cases follow. The first line of each case contains two integers **N** and **K**, with **N** denoting the number of powers and **K** denoting the number of followers. In the next line are **N** space separated integers denoting the intensities of each power.

**Output**

For each test case, output "yes" if it is possible to divide his powers equally amongst his followers; otherwise output "no" (without quotes).

**Constraints**

* **1** ≤ **T** ≤ **10**
* **1** ≤ **N** ≤ **21**
* **1** ≤ **K** ≤ **8**
* Subtask #1 (20 points) : **0** ≤ **intensity of power** ≤ **10^5**
* Subtask #2 (80 points) : **0** ≤ **intensity of power** ≤ **10^10**

**If the Input is :**

2

5 3

1 2 4 5 6

5 3

1 2 4 5 7

Yes

no