Codeforces Round #607 (Div. 2)

A. Suffix Three

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

We just discovered a new data structure in our research group: a **suffix three**!

It's very useful for natural language processing. Given three languages and three suffixes, a suffix three can determine which language a sentence is written in.

It's super simple, 100% accurate, and doesn't involve advanced machine learning algorithms.

Let us tell you how it works.

- If a sentence ends with "`po`" the language is Filipino.
- If a sentence ends with "`desu`" or "`masu`" the language is Japanese.
- If a sentence ends with "`mnida`" the language is Korean.

Given this, we need you to implement a suffix three that can differentiate Filipino, Japanese, and Korean.

Oh, did I say three suffixes? I meant four.

Input

The first line of input contains a single integer t $(1 \le t \le 30)$ denoting the number of test cases. The next lines contain descriptions of the test cases.

Each test case consists of a single line containing a single string denoting the sentence. Spaces are represented as underscores (the symbol "_") for ease of reading. The sentence has at least 1 and at most 1000 characters, and consists only of lowercase English letters and underscores. The sentence has no leading or trailing underscores and no two consecutive underscores. It is guaranteed that the sentence ends with one of the four suffixes mentioned above.

Output

For each test case, print a single line containing either "`FILIPINO`", "`JAPANESE`", or "`KOREAN`" (all in uppercase, without quotes), depending on the detected language.

Example

input

```
8
kamusta_po
genki_desu
ohayou_gozaimasu
annyeong_hashimnida
hajime_no_ippo
bensamu_no_sentou_houhou_ga_okama_kenpo
ang_halaman_doon_ay_sarisari_singkamasu
si_roy_mustang_ay_namamasu
```

output

```
FILIPINO
JAPANESE
JAPANESE
KOREAN
FILIPINO
FILIPINO
JAPANESE
JAPANESE
```

Note

The first sentence ends with "`po`", so it is written in Filipino.

The second and third sentences end with "`desu`" and "`masu`", so they are written in Japanese.

The fourth sentence ends with "`mnida`", so it is written in Korean.

B. Azamon Web Services

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Your friend Jeff Zebos has been trying to run his new online company, but it's not going very well. He's not getting a lot of sales on his website which he decided to call **Azamon**. His big problem, you think, is that he's not ranking high enough on the search engines. If only he could rename his products to have better names than his competitors, then he'll be at the top of the search results and will be a millionaire.

After doing some research, you find out that search engines only sort their results lexicographically. If your friend could rename his products to lexicographically smaller strings than his competitor's, then he'll be at the top of the rankings!

To make your strategy less obvious to his competitors, you decide to swap no more than two letters of the product names.

Please help Jeff to find improved names for his products that are lexicographically smaller than his competitor's!

Given the string s representing Jeff's product name and the string c representing his competitor's product name, find a way to swap **at most one pair** of characters in s (that is, find two distinct indices i and j and swap s_i and s_j) such that the resulting new name becomes strictly lexicographically smaller than c, or determine that it is impossible.

Note: String a is strictly lexicographically smaller than string b if and only if one of the following holds:

- a is a proper prefix of b, that is, a is a prefix of b such that $a \neq b$;
- There exists an integer $1 \leq i \leq \min(|a|, |b|)$ such that $a_i < b_i$ and $a_j = b_j$ for $1 \leq j < i$.

Input

The first line of input contains a single integer t ($1 \leq t \leq 1500$) denoting the number of test cases. The next lines contain descriptions of the test cases.

Each test case consists of a single line containing two space-separated strings s and c ($2 \leq |s| \leq 5000, 1 \leq |c| \leq 5000$). The strings s and c consists of uppercase English letters.

It is guaranteed that the sum of $|s|$ in the input is at most 5000 and the sum of the $|c|$ in the input is at most 5000.

Output

For each test case, output a single line containing a single string, which is either

- the new name which is obtained after swapping no more than one pair of characters that is strictly lexicographically smaller than c. In case there are many possible such strings, you can output any of them;
- three dashes (the string "---" without quotes) if it is impossible.

Example

input

```
3
AZAMON APPLE
AZAMON AAAAAAAAAAALIBABA
APPLE BANANA
```

output

```
AMAZON
---
APPLE
```

Note

In the first test case, it is possible to swap the second and the fourth letters of the string and the resulting string "AMAZON" is lexicographically smaller than "APPLE".

It is impossible to improve the product's name in the second test case and satisfy all conditions.

In the third test case, it is possible not to swap a pair of characters. The name "APPLE" is lexicographically smaller than "BANANA". Note that there are other valid answers, e.g., "APPEL".

C. Cut and Paste

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

We start with a string s consisting only of the digits 1, 2, or 3. The length of s is denoted by $|s|$. For each i from 1 to $|s|$, the i-th character of s is denoted by s_i.

There is one cursor. The cursor's location ℓ is denoted by an integer in $\{0, \ldots, |s|\}$, with the following meaning:

- If $\ell = 0$, then the cursor is located before the first character of s.
- If $\ell = |s|$, then the cursor is located right after the last character of s.
- If $0 < \ell < |s|$, then the cursor is located between s_ℓ and $s_{\ell+1}$.

We denote by s_{left} the string to the left of the cursor and s_{right} the string to the right of the cursor.

We also have a string c, which we call our clipboard, which starts out as empty. There are three types of actions:

- **The Move action**. Move the cursor one step to the right. This increments ℓ once.
- **The Cut action**. Set $c \leftarrow s_{\text{right}}$, then set $s \leftarrow s_{\text{left}}$.
- **The Paste action**. Append the value of c to the end of the string s. Note that this doesn't modify c.

The cursor initially starts at $\ell = 0$. Then, we perform the following procedure:

1. Perform the Move action once.
2. Perform the Cut action once.
3. Perform the Paste action s_ℓ times.
4. If $\ell = x$, stop. Otherwise, return to step 1.

You're given the initial string s and the integer x. What is the length of s when the procedure stops? Since this value may be very large, only find it modulo $10^9 + 7$.

It is guaranteed that $\ell \le |s|$ at any time.

Input

The first line of input contains a single integer t ($1 \le t \le 1000$) denoting the number of test cases. The next lines contain descriptions of the test cases.

The first line of each test case contains a single integer x ($1 \le x \le 10^6$). The second line of each test case consists of the initial string s ($1 \le |s| \le 500$). It is guaranteed, that s consists of the characters "1", "2", "3".

It is guaranteed that the sum of x in a single file is at most 10^6. It is guaranteed that in each test case before the procedure will stop it will be true that $\ell \le |s|$ at any time.

Output

For each test case, output a single line containing a single integer denoting the answer for that test case modulo $10^9 + 7$.

Example

input

```
4
5
231
7
2323
6
333
24
133321333
```

output

```
25
1438
1101
686531475
```

Note

Let's illustrate what happens with the first test case. Initially, we have $s = 231$. Initially, $\ell = 0$ and $c = \varepsilon$ (the empty string). The following things happen if we follow the procedure above:

- Step 1, *Move once:* we get $\ell = 1$.
- Step 2, *Cut once:* we get $s = 2$ and $c = 31$.
- Step 3, *Paste $s_\ell = 2$ times:* we get $s = 23131$.
- Step 4: $\ell = 1 \neq x = 5$, so we return to step 1.

- Step 1, *Move once:* we get $\ell = 2$.
- Step 2, *Cut once:* we get $s = 23$ and $c = 131$.
- Step 3, *Paste $s_\ell = 3$ times:* we get $s = 23131131131$.
- Step 4: $\ell = 2 \neq x = 5$, so we return to step 1.

- Step 1, *Move once:* we get $\ell = 3$.
- Step 2, *Cut once:* we get $s = 231$ and $c = 31131131$.
- Step 3, *Paste $s_\ell = 1$ time:* we get $s = 23131131131$.
- Step 4: $\ell = 3 \neq x = 5$, so we return to step 1.

- Step 1, *Move once:* we get $\ell = 4$.
- Step 2, *Cut once:* we get $s = 2313$ and $c = 1131131$.
- Step 3, *Paste $s_\ell = 3$ times:* we get $s = 2313113113111311311131131$.
- Step 4: $\ell = 4 \neq x = 5$, so we return to step 1.

- Step 1, *Move once:* we get $\ell = 5$.
- Step 2, *Cut once:* we get $s = 23131$ and $c = 13113111311311131131$.
- Step 3, *Paste $s_\ell = 1$ times:* we get $s = 2313113113111311311131131$.
- Step 4: $\ell = 5 = x$, so we stop.

At the end of the procedure, s has length 25.

D. Beingawesomeism

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

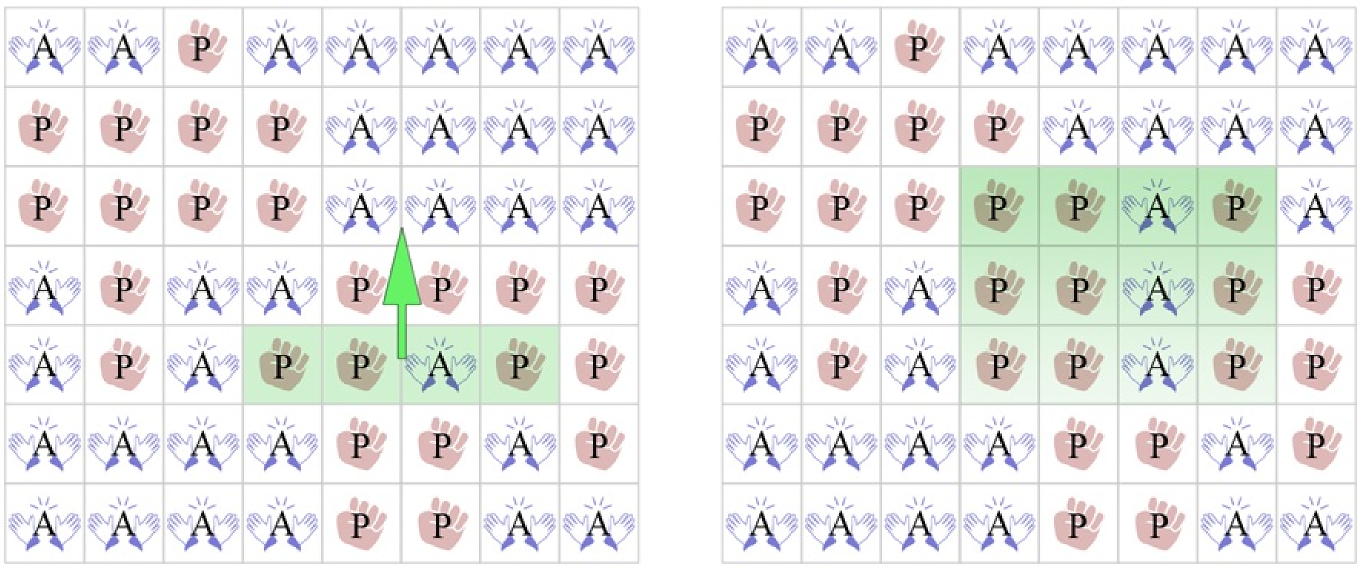
You are an all-powerful being and you have created a rectangular world. In fact, your world is so bland that it could be represented by a $r \times c$ grid. Each cell on the grid represents a country. Each country has a dominant religion. There are only two religions in your world. One of the religions is called Beingawesomeism, who do good for the sake of being good. The other religion is called Pushingittoofarism, who do murders for the sake of being bad.

Oh, and you are actually not really all-powerful. You just have one power, which you can use infinitely many times! Your power involves *missionary groups*. When a missionary group of a certain country, say a, passes by another country b, they change the dominant religion of country b to the dominant religion of country a.

In particular, a single use of your power is this:

- You choose a horizontal $1 \times x$ subgrid or a vertical $x \times 1$ subgrid. That value of x is up to you;
- You choose a direction d. If you chose a horizontal subgrid, your choices will either be NORTH or SOUTH. If you choose a vertical subgrid, your choices will either be EAST or WEST;
- You choose the number s of steps;
- You command each country in the subgrid to send a missionary group that will travel s steps towards direction d. In each step, they will visit (and in effect convert the dominant religion of) all s countries they pass through, as detailed above.
- The parameters x, d, s must be chosen in such a way that any of the missionary groups won't leave the grid.

The following image illustrates one possible single usage of your power. Here, A represents a country with dominant religion Beingawesomeism and P represents a country with dominant religion Pushingittoofarism. Here, we've chosen a 1×4 subgrid, the direction NORTH, and $s = 2$ steps.



You are a being which believes in free will, for the most part. However, you just really want to stop receiving murders that are attributed to your name. Hence, you decide to use your powers and try to make Beingawesomeism the dominant religion in every country.

What is the minimum number of usages of your power needed to convert everyone to Beingawesomeism?

With god, nothing is impossible. But maybe you're not god? If it is impossible to make Beingawesomeism the dominant religion in all countries, you must also admit your mortality and say so.

Input

The first line of input contains a single integer t ($1 \le t \le 2 \cdot 10^4$) denoting the number of test cases.

The first line of each test case contains two space-separated integers r and c denoting the dimensions of the grid ($1 \le r, c \le 60$). The next r lines each contains c characters describing the dominant religions in the countries. In particular, the j-th character in the i-th line describes the dominant religion in the country at the cell with row i and column j, where:

- "A" means that the dominant religion is Beingawesomeism;
- "P" means that the dominant religion is Pushingittoofarism.

It is guaranteed that the grid will only contain "A" or "P" characters. It is guaranteed that the sum of the $r \cdot c$ in a single file is at most $3 \cdot 10^6$.

Output

For each test case, output a single line containing the minimum number of usages of your power needed to convert everyone to

Beingawesomeism, or the string "MORTAL" (without quotes) if it is impossible to do so.

Example

input

```
4
7 8
AAPAAAAA
PPPPAAAA
PPPPAAAA
APAAPPPP
APAPPAPP
AAAAPPAP
AAAAPPAA
6 5
AAAAA
AAAAA
AAPAA
AAPAP
AAAPP
AAAPP
4 4
PPPP
PPPP
PPPP
PPPP
3 4
PPPP
PAAP
PPPP
```

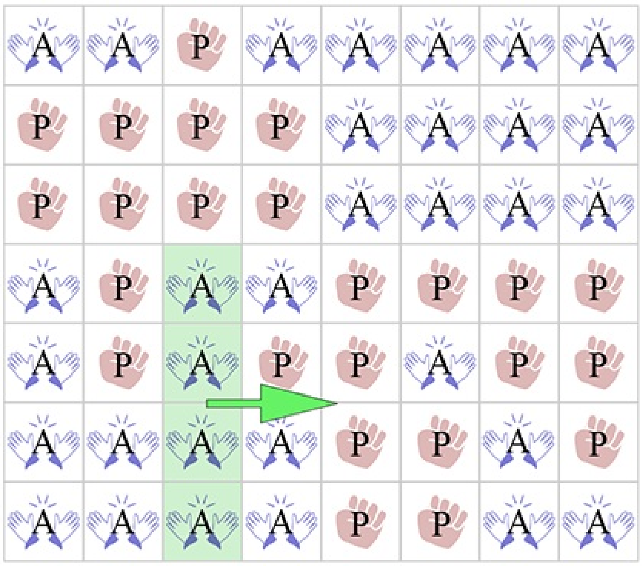
output

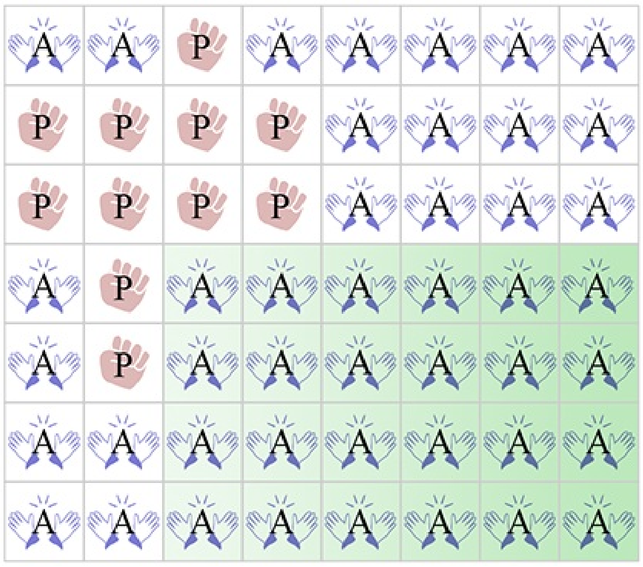
```
2
1
MORTAL
4
```

Note

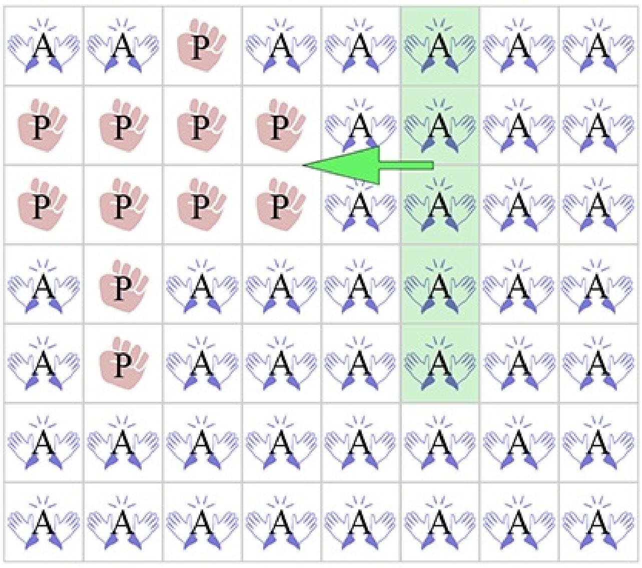
In the first test case, it can be done in two usages, as follows:

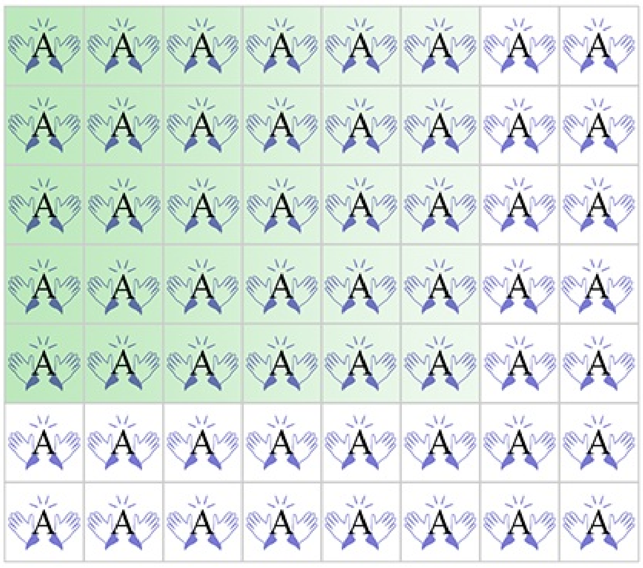
Usage 1:





Usage 2:





In the second test case, it can be done with just one usage of the power.

In the third test case, it is impossible to convert everyone to Beingawesomeism, so the answer is "`MORTAL`".

E. Jeremy Bearimy

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Welcome! Everything is fine.

You have arrived in The Medium Place, the place between The Good Place and The Bad Place. You are assigned a task that will either make people happier or torture them for eternity.

You have a list of k pairs of people who have arrived in a new inhabited neighborhood. You need to assign each of the $2k$ people into one of the $2k$ houses. Each person will be the resident of exactly one house, and each house will have exactly one resident.

Of course, in the neighborhood, it is possible to visit friends. There are $2k - 1$ roads, each of which connects two houses. It takes some time to traverse a road. We will specify the amount of time it takes in the input. The neighborhood is designed in such a way that from anyone's house, there is exactly one sequence of distinct roads you can take to any other house. In other words, the graph with the houses as vertices and the roads as edges is a tree.

The truth is, these k pairs of people are actually soulmates. We index them from 1 to k. We denote by $f(i)$ the amount of time it takes for the i-th pair of soulmates to go to each other's houses.

As we have said before, you will need to assign each of the $2k$ people into one of the $2k$ houses. You have two missions, one from the entities in The Good Place and one from the entities of The Bad Place. Here they are:

- The first mission, from The Good Place, is to assign the people into the houses such that the sum of $f(i)$ over all pairs i is minimized. Let's define this minimized sum as G. This makes sure that soulmates can easily and efficiently visit each other;
- The second mission, from The Bad Place, is to assign the people into the houses such that the sum of $f(i)$ over all pairs i is maximized. Let's define this maximized sum as B. This makes sure that soulmates will have a difficult time to visit each other.

What are the values of G and B?

Input

The first line of input contains a single integer t ($1 \le t \le 500$) denoting the number of test cases. The next lines contain descriptions of the test cases.

The first line of each test case contains a single integer k denoting the number of pairs of people ($1 \le k \le 10^5$). The next $2k - 1$ lines describe the roads; the i-th of them contains three space-separated integers a_i, b_i, t_i which means that the i-th road connects the a_i-th and b_i-th houses with a road that takes t_i units of time to traverse ($1 \le a_i, b_i \le 2k$, $a_i \neq b_i$, $1 \le t_i \le 10^6$). It is guaranteed that the given roads define a tree structure.

It is guaranteed that the sum of the k in a single file is at most $3 \cdot 10^5$.

Output

For each test case, output a single line containing two space-separated integers G and B.

Example

input

```
2
3
1 2 3
3 2 4
```

```
2 4 3
4 5 6
5 6 5
2
1 2 1
1 3 2
1 4 3
```

output

```
15 33
6 6
```

Note

For the sample test case, we have a minimum sum equal to $G = 15$. One way this can be achieved is with the following assignment:

- The first pair of people get assigned to houses 5 and 6, giving us $f(1) = 5$;
- The second pair of people get assigned to houses 1 and 4, giving us $f(2) = 6$;
- The third pair of people get assigned to houses 3 and 2, giving us $f(3) = 4$.

Note that the sum of the $f(i)$ is $5 + 6 + 4 = 15$.

We also have a maximum sum equal to $B = 33$. One way this can be achieved is with the following assignment:

- The first pair of people get assigned to houses 1 and 4, giving us $f(1) = 6$;
- The second pair of people get assigned to houses 6 and 2, giving us $f(2) = 14$;
- The third pair of people get assigned to houses 3 and 5, giving us $f(3) = 13$.

Note that the sum of the $f(i)$ is $6 + 14 + 13 = 33$.

F. Miss Punyverse

time limit per test: 4 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

The Oak has n nesting places, numbered with integers from 1 to n. Nesting place i is home to b_i bees and w_i wasps.

Some nesting places are connected by branches. We call two nesting places adjacent if there exists a branch between them. A simple path from nesting place x to y is given by a sequence $s_0, \ldots, s_p$ of distinct nesting places, where p is a non-negative integer, $s_0 = x$, $s_p = y$, and s_{i-1} and s_i are adjacent for each $i = 1, \ldots, p$. The branches of The Oak are set up in such a way that for any two pairs of nesting places x and y, there exists a unique simple path from x to y. Because of this, biologists and computer scientists agree that The Oak is in fact, a tree.

A village is a nonempty set V of nesting places such that for any two x and y in V, there exists a simple path from x to y whose intermediate nesting places all lie in V.

A set of villages $\mathcal{P}$ is called a partition if each of the n nesting places is contained in exactly one of the villages in $\mathcal{P}$. In other words, no two villages in $\mathcal{P}$ share any common nesting place, and altogether, they contain all n nesting places.

The Oak holds its annual Miss Punyverse beauty pageant. The two contestants this year are Ugly Wasp and Pretty Bee. The winner of the beauty pageant is determined by voting, which we will now explain. Suppose $\mathcal{P}$ is a partition of the nesting places into m villages $V_1, \ldots, V_m$. There is a local election in each village. Each of the insects in this village vote for their favorite contestant. If there are **strictly** more votes for Ugly Wasp than Pretty Bee, then Ugly Wasp is said to *win* in that village. Otherwise, Pretty Bee *wins*. Whoever wins in the most number of villages wins.

As it always goes with these pageants, bees always vote for the bee (which is Pretty Bee this year) and wasps always vote for the wasp (which is Ugly Wasp this year). Unlike their general elections, no one abstains from voting for Miss Punyverse as everyone takes it very seriously.

Mayor Waspacito, and his assistant Alexwasp, wants Ugly Wasp to win. He has the power to choose how to partition The Oak into exactly m villages. If he chooses the partition optimally, determine the maximum number of villages in which Ugly Wasp wins.

Input

The first line of input contains a single integer t ($1 \le t \le 100$) denoting the number of test cases. The next lines contain descriptions of the test cases.

The first line of each test case contains two space-separated integers n and m ($1 \le m \le n \le 3000$). The second line contains n space-separated integers $b_1, b_2, \ldots, b_n$ ($0 \le b_i \le 10^9$). The third line contains n space-separated integers $w_1, w_2, \ldots, w_n$ ($0 \le w_i \le 10^9$). The next $n - 1$ lines describe the pairs of adjacent nesting places. In particular, the i-th of them contains two space-separated integers x_i and y_i ($1 \le x_i, y_i \le n$, $x_i \ne y_i$) denoting the numbers of two adjacent nesting places. It is guaranteed that these pairs form a tree.

It is guaranteed that the sum of n in a single file is at most 10^5.

Output

For each test case, output a single line containing a single integer denoting the maximum number of villages in which Ugly Wasp wins, among all partitions of The Oak into m villages.

Example

input

```
2
4 3
10 160 70 50
70 111 111 0
1 2
2 3
3 4
2 1
143 420
214 349
2 1
```

output

```
2
0
```

Note

In the first test case, we need to partition the $n = 4$ nesting places into $m = 3$ villages. We can make Ugly Wasp win in 2 villages via the following partition: $\{\{1, 2\}, \{3\}, \{4\}\}$. In this partition,

- Ugly Wasp wins in village $\{1, 2\}$, garnering 181 votes as opposed to Pretty Bee's 170;
- Ugly Wasp also wins in village $\{3\}$, garnering 111 votes as opposed to Pretty Bee's 70;
- Ugly Wasp loses in the village $\{4\}$, garnering 0 votes as opposed to Pretty Bee's 50.

Thus, Ugly Wasp wins in 2 villages, and it can be shown that this is the maximum possible number.

In the second test case, we need to partition the $n = 2$ nesting places into $m = 1$ village. There is only one way to do this: $\{\{1, 2\}\}$. In this partition's sole village, Ugly Wasp gets 563 votes, and Pretty Bee also gets 563 votes. Ugly Wasp needs strictly more votes in order to win. Therefore, Ugly Wasp doesn't win in any village.