Codeforces Global Round 7

A. Bad Ugly Numbers

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

You are given a integer n ($n > 0$). Find **any** integer s which satisfies these conditions, or report that there are no such numbers:

In the decimal representation of s:

- $s > 0$,
- s consists of n digits,
- no digit in s equals 0,
- s is not divisible by any of it's digits.

Input

The input consists of multiple test cases. The first line of the input contains a single integer t ($1 \le t \le 400$), the number of test cases. The next t lines each describe a test case.

Each test case contains one positive integer n ($1 \le n \le 10^5$).

It is guaranteed that the sum of n for all test cases does not exceed 10^5.

Output

For each test case, print an integer s which satisfies the conditions described above, or "-1" (without quotes), if no such number exists. If there are multiple possible solutions for s, print **any** solution.

Example

input

```
4
1
2
3
4
```

output

```
-1
57
239
6789
```

Note

In the first test case, there are no possible solutions for s consisting of one digit, because any such solution is divisible by itself.

For the second test case, the possible solutions are: 23, 27, 29, 34, 37, 38, 43, 46, 47, 49, 53, 54, 56, 57, 58, 59, 67, 68, 69, 73, 74, 76, 78, 79, 83, 86, 87, 89, 94, 97, and 98.

For the third test case, one possible solution is 239 because 239 is not divisible by 2, 3 or 9 and has three digits (none of which equals zero).

B. Maximums

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

Alicia has an array, $a_1, a_2, \ldots, a_n$, of non-negative integers. For each $1 \le i \le n$, she has found a non-negative integer $x_i = max(0, a_1, \ldots, a_{i-1})$. Note that for $i = 1$, $x_i = 0$.

For example, if Alicia had the array $a = \{0, 1, 2, 0, 3\}$, then $x = \{0, 0, 1, 2, 2\}$.

Then, she calculated an array, $b_1, b_2, \ldots, b_n$: $b_i = a_i - x_i$.

For example, if Alicia had the array $a = \{0, 1, 2, 0, 3\}$, $b = \{0 - 0, 1 - 0, 2 - 1, 0 - 2, 3 - 2\} = \{0, 1, 1, -2, 1\}$.

Alicia gives you the values $b_1, b_2, \ldots, b_n$ and asks you to restore the values $a_1, a_2, \ldots, a_n$. Can you help her solve the problem?

Input

The first line contains one integer n ($3 \le n \le 200\,000$) – the number of elements in Alicia's array.

The next line contains n integers, $b_1, b_2, \ldots, b_n$ ($-10^9 \le b_i \le 10^9$).

It is guaranteed that for the given array b there is a solution $a_1, a_2, \ldots, a_n$, for all elements of which the following is true: $0 \le a_i \le 10^9$.

Output

Print n integers, $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 10^9$), such that if you calculate x according to the statement, b_1 will be equal to $a_1 - x_1$, b_2 will be equal to $a_2 - x_2$, ..., and b_n will be equal to $a_n - x_n$.

It is guaranteed that there exists at least one solution for the given tests. It can be shown that the solution is unique.

Examples

input

```
5
0 1 1 -2 1
```

output

```
0 1 2 0 3
```

input

```
3
1000 999999000 -1000000000
```

output

```
1000 1000000000 0
```

input

```
5
2 1 2 2 3
```

output

```
2 3 5 7 10
```

Note

The first test was described in the problem statement.

In the second test, if Alicia had an array $a = \{1000, 1000000000, 0\}$, then $x = \{0, 1000, 1000000000\}$ and $b = \{1000 - 0, 1000000000 - 1000, 0 - 1000000000\} = \{1000, 999999000, -1000000000\}$.

C. Permutation Partitions

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

You are given a permutation $p_1, p_2, \ldots, p_n$ of integers from 1 to n and an integer k, such that $1 \le k \le n$. A permutation means that every number from 1 to n is contained in p exactly once.

Let's consider all partitions of this permutation into k disjoint segments. Formally, a partition is a set of segments $\{[l_1, r_1], [l_2, r_2], \ldots, [l_k, r_k]\}$, such that:

- $1 \le l_i \le r_i \le n$ for all $1 \le i \le k$;
- For all $1 \le j \le n$ there exists **exactly one** segment $[l_i, r_i]$, such that $l_i \le j \le r_i$.

Two partitions are different if there exists a segment that lies in one partition but not the other.

Let's calculate *the partition value*, defined as $\sum\limits_{i=1}^{k} \max\limits_{l_i \le j \le r_i} p_j$, for all possible partitions of the permutation into k disjoint segments. Find the maximum possible partition value over all such partitions, and the number of partitions with this value. As the second value can be very large, you should find its remainder when divided by $998\,244\,353$.

Input

The first line contains two integers, n and k ($1 \le k \le n \le 200\,000$) — the size of the given permutation and the number of segments in a partition.

The second line contains n different integers $p_1, p_2, \ldots, p_n$ ($1 \le p_i \le n$) — the given permutation.

Output

Print two integers — the maximum possible partition value over all partitions of the permutation into k disjoint segments and the number of such partitions for which the partition value is equal to the maximum possible value, modulo $998\,244\,353$.

Please note that you should only find **the second** value modulo $998\,244\,353$.

Examples

input

```
3 2
2 1 3
```

output

```
5 2
```

input

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

output

```
15 1
```

input

```
7 3
2 7 3 1 5 4 6
```

output

```
18 6
```

Note

In the first test, for $k = 2$, there exists only two valid partitions: $\{[1, 1], [2, 3]\}$ and $\{[1, 2], [3, 3]\}$. For each partition, the partition value is equal to $2 + 3 = 5$. So, the maximum possible value is 5 and the number of partitions is 2.

In the third test, for $k = 3$, the partitions with the maximum possible partition value are $\{[1, 2], [3, 5], [6, 7]\}$, $\{[1, 3], [4, 5], [6, 7]\}$, $\{[1, 4], [5, 5], [6, 7]\}$, $\{[1, 2], [3, 6], [7, 7]\}$, $\{[1, 3], [4, 6], [7, 7]\}$, $\{[1, 4], [5, 6], [7, 7]\}$. For all of them, the partition value is equal to $7 + 5 + 6 = 18$.

The partition $\{[1, 2], [3, 4], [5, 7]\}$, however, has the partition value $7 + 3 + 6 = 16$. This is not the maximum possible value, so we don't count it.

D1. Prefix-Suffix Palindrome (Easy version)

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

This is the easy version of the problem. The difference is the constraint on the sum of lengths of strings and the number of test cases. You can make hacks only if you solve all versions of this task.

You are given a string s, consisting of lowercase English letters. Find the longest string, t, which satisfies the following conditions:

- The length of t does not exceed the length of s.
- t is a palindrome.
- There exists two strings a and b (possibly empty), such that $t = a + b$ ("$+$" represents concatenation), and a is prefix of s while b is suffix of s.

Input

The input consists of multiple test cases. The first line contains a single integer t ($1 \leq t \leq 1000$), the number of test cases. The next t lines each describe a test case.

Each test case is a non-empty string s, consisting of lowercase English letters.

It is guaranteed that the sum of lengths of strings over all test cases does not exceed 5000.

Output

For each test case, print the longest string which satisfies the conditions described above. If there exists multiple possible solutions, print any of them.

Example

input

```
5
a
abcdfdcecba
abbaxyzyx
codeforces
acbba
```

output

```
a
abcdfdcba
xyzyx
c
abba
```

Note

In the first test, the string $s =$"a" satisfies all conditions.

In the second test, the string "abcdfdcba" satisfies all conditions, because:

- Its length is 9, which does not exceed the length of the string s, which equals 11.
- It is a palindrome.
- "abcdfdcba" $=$ "abcdfdc" $+$ "ba", and "abcdfdc" is a prefix of s while "ba" is a suffix of s.

It can be proven that there does not exist a longer string which satisfies the conditions.

In the fourth test, the string "c" is correct, because "c" $=$ "c" $+$ "" and a or b can be empty. The other possible solution for this test is "s".

D2. Prefix-Suffix Palindrome (Hard version)

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

This is the hard version of the problem. The difference is the constraint on the sum of lengths of strings and the number of test cases. You can make hacks only if you solve all versions of this task.

You are given a string s, consisting of lowercase English letters. Find the longest string, t, which satisfies the following conditions:

- The length of t does not exceed the length of s.
- t is a palindrome.
- There exists two strings a and b (possibly empty), such that $t = a + b$ ("$+$" represents concatenation), and a is prefix of s while b is suffix of s.

Input

The input consists of multiple test cases. The first line contains a single integer t ($1 \leq t \leq 10^5$), the number of test cases. The next t lines each describe a test case.

Each test case is a non-empty string s, consisting of lowercase English letters.

It is guaranteed that the sum of lengths of strings over all test cases does not exceed 10^6.

Output

For each test case, print the longest string which satisfies the conditions described above. If there exists multiple possible solutions, print any of them.

Example

input

```
5
a
abcdfdcecba
abbaxyzyx
codeforces
acbba
```

output

```
a
abcdfdcba
xyzyx
c
abba
```

Note

In the first test, the string $s =$"a" satisfies all conditions.

In the second test, the string "abcdfdcba" satisfies all conditions, because:

- Its length is 9, which does not exceed the length of the string s, which equals 11.
- It is a palindrome.
- "abcdfdcba" $=$ "abcdfdc" $+$ "ba", and "abcdfdc" is a prefix of s while "ba" is a suffix of s.

It can be proven that there does not exist a longer string which satisfies the conditions.

In the fourth test, the string "c" is correct, because "c" $=$ "c" $+$ "" and a or b can be empty. The other possible solution for this test is "s".

E. Bombs

time limit per test: 3 seconds
memory limit per test: 256 megabytes

input: standard input
output: standard output

You are given a permutation, $p_1, p_2, \ldots, p_n$.

Imagine that some positions of the permutation contain bombs, such that there exists at least one position without a bomb.

For some fixed configuration of bombs, consider the following process. Initially, there is an empty set, A.

For each i from 1 to n:

- Add p_i to A.
- If the i-th position contains a bomb, remove the largest element in A.

After the process is completed, A will be non-empty. The *cost of the configuration of bombs* equals the largest element in A.

You are given another permutation, $q_1, q_2, \ldots, q_n$.

For each $1 \le i \le n$, find the cost of a configuration of bombs such that there exists a bomb in positions $q_1, q_2, \ldots, q_{i-1}$.

For example, for $i = 1$, you need to find the cost of a configuration without bombs, and for $i = n$, you need to find the cost of a configuration with bombs in positions $q_1, q_2, \ldots, q_{n-1}$.

Input

The first line contains a single integer, n ($2 \le n \le 300\,000$).

The second line contains n distinct integers $p_1, p_2, \ldots, p_n$ $(1 \le p_i \le n)$.

The third line contains n distinct integers $q_1, q_2, \ldots, q_n$ $(1 \le q_i \le n)$.

Output

Print n space-separated integers, such that the i-th of them equals the cost of a configuration of bombs in positions $q_1, q_2, \ldots, q_{i-1}$.

Examples

input

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

output

```
3 2 1
```

input

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

output

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

Note

In the first test:

- If there are no bombs, A is equal to $\{1, 2, 3\}$ at the end of the process, so the cost of the configuration is 3.
- If there is one bomb in position 1, A is equal to $\{1, 2\}$ at the end of the process, so the cost of the configuration is 2;
- If there are two bombs in positions 1 and 2, A is equal to $\{1\}$ at the end of the process, so the cost of the configuration is 1.

In the second test:

Let's consider the process for $i = 4$. There are three bombs on positions $q_1 = 5$, $q_2 = 2$, and $q_3 = 1$.

At the beginning, $A = \{\}$.

- Operation 1: Add $p_1 = 2$ to A, so A is equal to $\{2\}$. There exists a bomb in position 1, so we should delete the largest element from A. A is equal to $\{\}$.
- Operation 2: Add $p_2 = 3$ to A, so A is equal to $\{3\}$. There exists a bomb in position 2, so we should delete the largest element from A. A is equal to $\{\}$.
- Operation 3: Add $p_3 = 6$ to A, so A is equal to $\{6\}$. There is no bomb in position 3, so we do nothing.
- Operation 4: Add $p_4 = 1$ to A, so A is equal to $\{1, 6\}$. There is no bomb in position 4, so we do nothing.
- Operation 5: Add $p_5 = 5$ to A, so A is equal to $\{1, 5, 6\}$. There exists a bomb in position 5, so we delete the largest element from A. Now, A is equal to $\{1, 5\}$.
- Operation 6: Add $p_6 = 4$ to A, so A is equal to $\{1, 4, 5\}$. There is no bomb in position 6, so we do nothing.

In the end, we have $A = \{1, 4, 5\}$, so the cost of the configuration is equal to 5.

F1. Wise Men (Easy Version)

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

This is the easy version of the problem. The difference is constraints on the number of wise men and the time limit. You can make hacks only if all versions of this task are solved.

n wise men live in a beautiful city. Some of them know each other.

For each of the $n!$ possible permutations $p_1, p_2, \ldots, p_n$ of the wise men, let's generate a binary string of length $n-1$: for each $1 \leq i < n$ set $s_i = 1$ if p_i and p_{i+1} know each other, and $s_i = 0$ otherwise.

For all possible 2^{n-1} binary strings, find the number of permutations that produce this binary string.

Input

The first line of input contains one integer n $(2 \leq n \leq 14)$ — the number of wise men in the city.

The next n lines contain a binary string of length n each, such that the j-th character of the i-th string is equal to '1' if wise man i knows wise man j, and equals '0' otherwise.

It is guaranteed that if the i-th man knows the j-th man, then the j-th man knows i-th man and no man knows himself.

Output

Print 2^{n-1} space-separated integers. For each $0 \leq x < 2^{n-1}$:

- Let's consider a string s of length $n-1$, such that $s_i = \lfloor \frac{x}{2^{i-1}} \rfloor \bmod 2$ for all $1 \leq i \leq n-1$.
- The $(x+1)$-th number should be equal to the required answer for s.

Examples

input

```
3
011
101
110
```

output

```
0 0 0 6
```

input

```
4
0101
1000
0001
1010
```

output

```
2 2 6 2 2 6 2 2
```

Note

In the first test, each wise man knows each other, so every permutation will produce the string 11.

In the second test:

- If $p = \{1, 2, 3, 4\}$, the produced string is 101, because wise men 1 and 2 know each other, 2 and 3 don't know each other, and 3 and 4 know each other;
- If $p = \{4, 1, 2, 3\}$, the produced string is 110, because wise men 1 and 4 know each other, 1 and 2 know each other and 2, and 3 don't know each other;
- If $p = \{1, 3, 2, 4\}$, the produced string is 000, because wise men 1 and 3 don't know each other, 3 and 2 don't know each other, and 2 and 4 don't know each other.

F2. Wise Men (Hard Version)

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

This is the hard version of the problem. The difference is constraints on the number of wise men and the time limit. You can make hacks only if all versions of this task are solved.

n wise men live in a beautiful city. Some of them know each other.

For each of the $n!$ possible permutations $p_1, p_2, \ldots, p_n$ of the wise men, let's generate a binary string of length $n-1$: for each $1 \leq i < n$ set $s_i = 1$ if p_i and p_{i+1} know each other, and $s_i = 0$ otherwise.

For all possible 2^{n-1} binary strings, find the number of permutations that produce this binary string.

Input

The first line of input contains one integer n $(2 \le n \le 18)$ — the number of wise men in the city.

The next n lines contain a binary string of length n each, such that the j-th character of the i-th string is equal to '1' if wise man i knows wise man j, and equals '0' otherwise.

It is guaranteed that if the i-th man knows the j-th man, then the j-th man knows i-th man and no man knows himself.

Output

Print 2^{n-1} space-separated integers. For each $0 \le x < 2^{n-1}$:

- Let's consider a string s of length $n-1$, such that $s_i = \lfloor \frac{x}{2^{i-1}} \rfloor \bmod 2$ for all $1 \le i \le n-1$.
- The $(x+1)$-th number should be equal to the required answer for s.

Examples

input

```
3
011
101
110
```

output

```
0 0 0 6
```

input

```
4
0101
1000
0001
1010
```

output

```
2 2 6 2 2 6 2 2
```

Note

In the first test, each wise man knows each other, so every permutation will produce the string 11.

In the second test:

- If $p = \{1, 2, 3, 4\}$, the produced string is 101, because wise men 1 and 2 know each other, 2 and 3 don't know each other, and 3 and 4 know each other;
- If $p = \{4, 1, 2, 3\}$, the produced string is 110, because wise men 1 and 4 know each other, 1 and 2 know each other and 2, and 3 don't know each other;
- If $p = \{1, 3, 2, 4\}$, the produced string is 000, because wise men 1 and 3 don't know each other, 3 and 2 don't know each other, and 2 and 4 don't know each other.

G. Spiderweb Trees

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

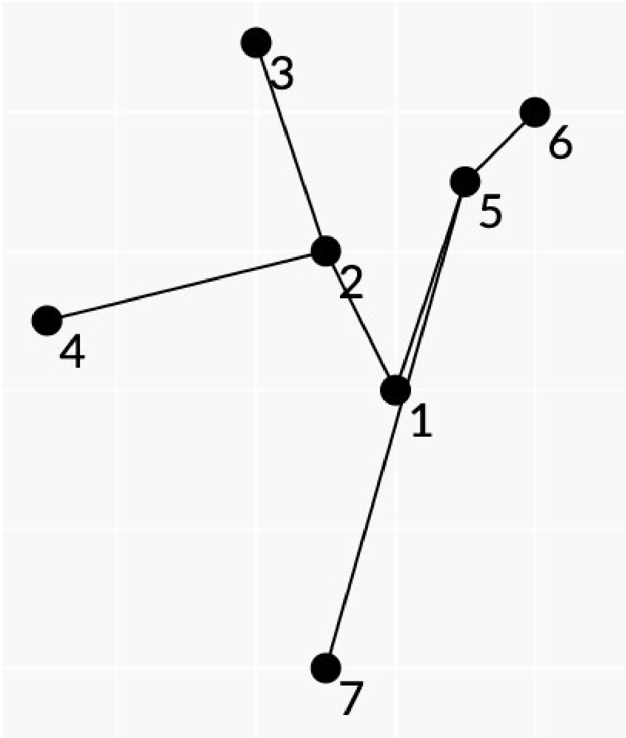
Let's call a graph with n vertices, each of which has it's own point $A_i = (x_i, y_i)$ with integer coordinates, a *planar tree* if:

- All points $A_1, A_2, \ldots, A_n$ are different and no three points lie on the same line.
- The graph is a tree, i.e. there are exactly $n-1$ edges there exists a path between any pair of vertices.
- For all pairs of edges (s_1, f_1) and (s_2, f_2), such that $s_1 \neq s_2$, $s_1 \neq f_2$, $f_1 \neq s_2$, and $f_1 \neq f_2$, the segments $A_{s_1}A_{f_1}$ and $A_{s_2}A_{f_2}$ don't intersect.

Imagine a planar tree with n vertices. Consider the convex hull of points $A_1, A_2, \ldots, A_n$. Let's call this tree a *spiderweb tree* if for all $1 \le i \le n$ the following statements are true:

- All leaves (vertices with degree ≤ 1) of the tree lie on the border of the convex hull.
- All vertices on the border of the convex hull are leaves.

An example of a spiderweb tree:



The points A_3, A_6, A_7, A_4 lie on the convex hull and the leaf vertices of the tree are $3, 6, 7, 4$.

Refer to the notes for more examples.

Let's call the subset $S \subset \{1, 2, \ldots, n\}$ of vertices a *subtree* of the tree if for all pairs of vertices in S, there exists a path that contains only vertices from S. Note that any subtree of the planar tree is a planar tree.

You are given a planar tree with n vertexes. Let's call a partition of the set $\{1, 2, \ldots, n\}$ into non-empty subsets $A_1, A_2, \ldots, A_k$ (i.e. $A_i \cap A_j = \emptyset$ for all $1 \leq i < j \leq k$ and $A_1 \cup A_2 \cup \ldots \cup A_k = \{1, 2, \ldots, n\}$) good if for all $1 \leq i \leq k$, the subtree A_i is a spiderweb tree. Two partitions are different if there exists some set that lies in one parition, but not the other.

Find the number of good partitions. Since this number can be very large, find it modulo $998\,244\,353$.

Input

The first line contains an integer n ($1 \leq n \leq 100$) — the number of vertices in the tree.

The next n lines each contain two integers x_i, y_i ($-10^9 \leq x_i, y_i \leq 10^9$) — the coordinates of i-th vertex, A_i.

The next $n-1$ lines contain two integers s, f ($1 \leq s, f \leq n$) — the edges (s, f) of the tree.

It is guaranteed that all given points are different and that no three of them lie at the same line. Additionally, it is guaranteed that the given edges and coordinates of the points describe a planar tree.

Output

Print one integer — the number of good partitions of vertices of the given planar tree, modulo $998\,244\,353$.

Examples

input

```
4
0 0
0 1
-1 -1
1 -1
1 2
1 3
1 4
```

output

```
5
```

input

```
5
3 2
0 -3
-5 -3
5 -5
4 5
4 2
4 1
5 2
2 3
```

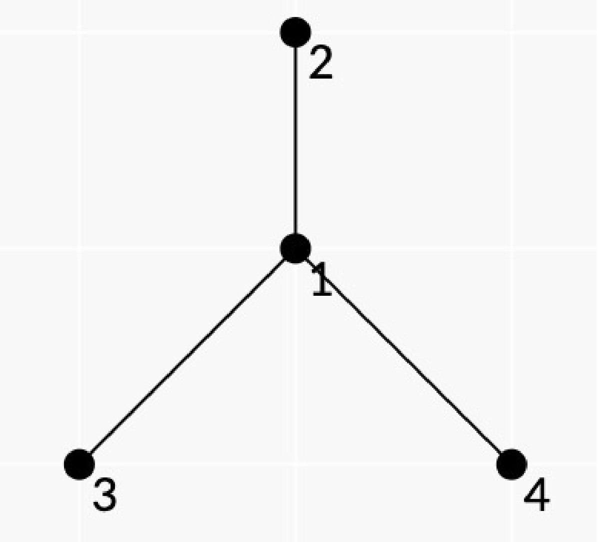
output

```
8
```

input
6 4 -5 0 1 -2 8 3 -10 0 -1 -4 -5 2 5 3 2 1 2 4 6 4 2
output
13

input
8 0 0 -1 2 -2 5 -5 1 1 3 0 3 2 4 -1 -4 1 2 3 2 5 6 4 2 1 5 5 7 5 8
output
36

Note



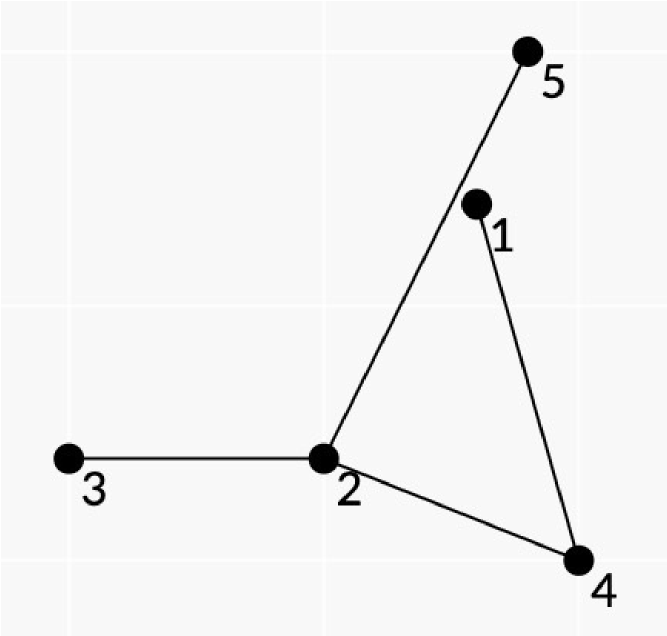
The picture for the first sample.

In the first test case, all good partitions are:

1. $\{1\}$, $\{2\}$, $\{3\}$, $\{4\}$;
2. $\{1, 2\}$, $\{3\}$, $\{4\}$;
3. $\{1, 3\}$, $\{2\}$, $\{4\}$;
4. $\{1, 4\}$, $\{2\}$, $\{3\}$;
5. $\{1, 2, 3, 4\}$.

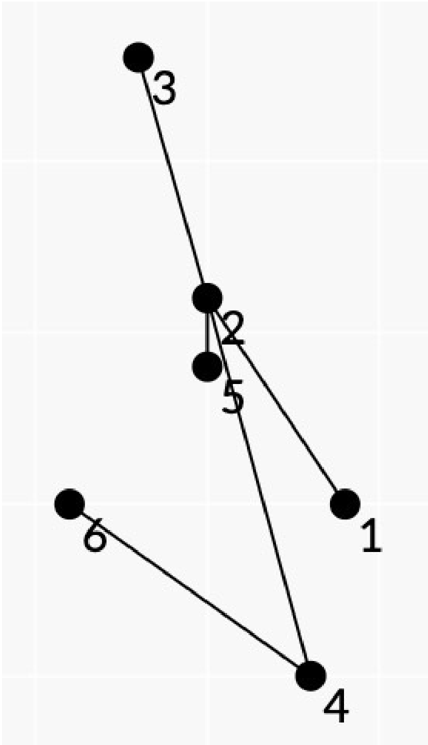
The partition $\{1, 2, 3\}$, $\{4\}$ isn't good, because the subtree $\{1, 2, 3\}$ isn't spiderweb tree, since the non-leaf vertex 1 lies on the convex hull.

The partition $\{2, 3, 4\}$, $\{1\}$ isn't good, because the subset $\{2, 3, 4\}$ isn't a subtree.

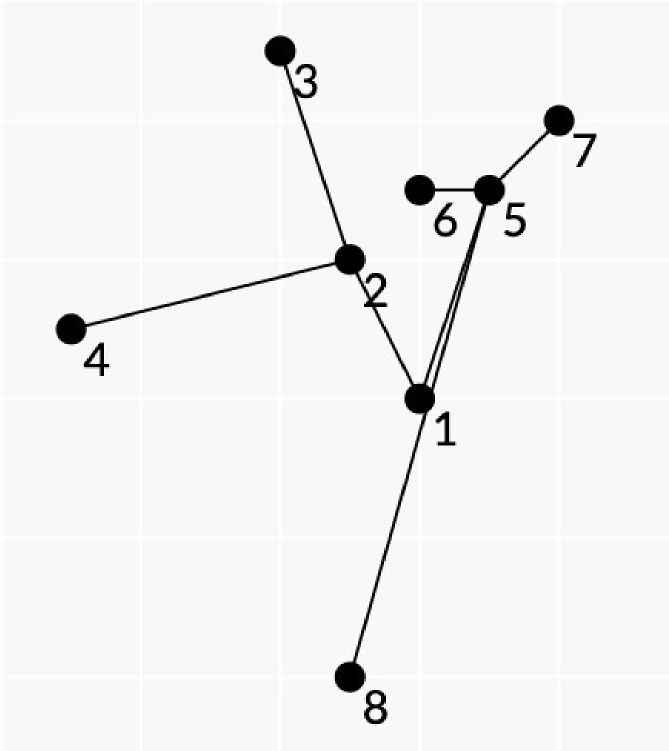


The picture for the second sample.

In the second test case, the given tree isn't a spiderweb tree, because the leaf vertex 1 doesn't lie on the convex hull. However, the subtree $\{2, 3, 4, 5\}$ is a spiderweb tree.



The picture for the third sample.



The picture for the fourth sample.

In the fourth test case, the partition $\{1, 2, 3, 4\}$, $\{5, 6, 7, 8\}$ is good because all subsets are spiderweb subtrees.
