Mock INOI #6

A. Placement

2 seconds, 512 megabytes

You are the facilities manager of a new school, and you are now tasked with your most difficult challenge thus far - finding the right place to put a water fountain.

Your school has N locations (which can be any type of location, including classrooms, sports halls, libraries, and so on). These locations are numbered from 1 to N. Because funding is low, these locations are connected with $N-1$ bidirectional routes of length 1. It is obviously possible to travel between any two locations in the school.

K of the locations in the school are sports facilities. And therefore, your task is to place the water fountain in a location which is close to all these facilities. In particular, you want to find the number of locations for which the total sum of distances to all sports facilities is minimum. You also want to find this minimum sum of distances.

Input

The first line of input contains N, the number of locations in the school.

The following $N-1$ lines contain two space-separated integers, A and B. Each of these represent a route between A and B.

The next line of input contains K, the number of sports facilities.

The last line contains K space-separated integers, which denote the sports facilities.

Output

Output two space-separated integer, the number of locations which minimise the total sum of distances to all sports facilities, and this minimum sum.

Scoring

For all subtasks, $1 \leq N \leq 3 \times 10^5$ and $1 \leq K \leq N$. $1 \leq A, B \leq N$, and $A \neq B$.

- Subtask 1 [11 points]: $N = 1$
- Subtask 2 [9 points]: $K = 1$
- Subtask 3 [10 points]: $K \leq 2$
- Subtask 4 [16 points]: K is odd.
- Subtask 5 [13 points]: $N, K \leq 400$
- Subtask 6 [11 points]: $N, K \leq 2000$
- Subtask 7 [14 points]: $K = N$
- Subtask 8 [16 points]: No additional constraints.

input

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

output

```
1 3
```

input

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

output

```
2 4
```

input

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

output

```
1 0
```

For the first sample, the best location is 1, where the sum of distances to 2, 3 and 4 is 3. You can check that the sum of distances to all other nodes is larger, so this is the unique best position, so we output 1 for the number of best places, and 3 for the sum of distances to all sports nodes..

For the second sample, either 2 or 3 can be the best location. The sum of the distances from all sports nodes (since here, all nodes are special) to these nodes is 4.

For the final sample, 5 itself is the best possible location, with a total distance of 0 to all sports locations (just itself).

B. Polyhash

2.5 seconds, 512 megabytes

A hashing function is a function which takes a large piece of data as input, and then convert into a smaller piece of data. It is quite widely used in computer science. Servers often store hashes of passwords, and when a user enters their password, it is hashed and compared with the stored value in the server. This is safe as hash functions are typically incredibly difficult to inverse (it's easy to calculate the hash of data, but not retrieve the data from a hash), and therefore the original passwords are less likely to be discovered, even if the hashes are stolen.

Today, we're going to try and break a hash function which is commonly used in competitive programming, but is quite bad in practice (you will soon see why). Our version will only deal with hashing strings into integers.

This hashing technique is known as a polynomial hash. First, each character in the string is converted into an integer. In particular, a is given the value 1, b is given the value 2, and so on until z, which is given the value 26. Then, you are given some base p (which is prime), and you give the first character a place value of 1, the second character a place value of p, the third character a place value of p^2, and so on, until the last character which has a place value of p^{n-1} for a string of length n. Then, this is interpreted as a base-10 integer, modulo some other large prime m. This final integer is considered to be the polynomial hash. In fact, we have provided a linear-time implementation of the polynomial hash for you, which you can use for this problem.

Here is some C++ code to implement the above procedure:

```
int polyhash(string& s, int p, int m) {
        long long res = 0;
        long long place = 1;
        for (int i=0; i<s.size(); i++) {
                res += ((s[i] - 'a' + 1) * place) % m;
                res %= m;
                place *= p;
                place %= m;
        }
        return res;
}
```

And here is some Python code to implement the same procedure:

```
def polyhash(s, p, m):
    res = 0
    place = 1
    for i in s:
        res += ((ord(i)-ord('a')+1) * place) % m
        res %= m
        place *= p
        p %= m
    return res
```

Your task is quite simple. You're given the integers p and m, and an additional integer x, where $0 \le x < m$. You need to find a string S with length of between 1 and 10^5 (both inclusive), such that the value of $polyhash(S, p, m)$ is equal to x.

Input
The input contains three space-separated integers, p, m and x.

Output
Output a single string S of length between 1 and 10^5 inclusive, such that the value of $polyhash(S, p, m) = x$

Scoring
For all subtasks, p and m are prime numbers, with $26 < p < m \le 10^9 + 7$. Additionally, $0 \le x < m$.

- Subtask 1 [9 points]: $x = 1$
- Subtask 2 [13 points]: $1 \le x \le 26$
- Subtask 3 [15 points]: $m \le 100$
- Subtask 4 [22 points]: $m \le 2000$
- Subtask 5 [41 points]: No additional constraints.

input
31 10007 14
output
n

input
31 10007 159
output
de

input
107 1000000007 57832922
output
spjmkkkoor

You can verify for each of these cases that $polyhash(S, p, m) = x$

C. Stadium

4 seconds, 512 megabytes

N spectators have come to the Camp Nou stadium tonight to watch the final of the Champions League. As the Champions League is a major event, spectators from many teams have come to watch the final two teams play out 90 gruelling minutes to determine the best team in all of Europe (and realistically, the best team in the world).

You're the media director of today's event, and you notice that every spectator is wearing the colours of the team they support. You want to make the entire stadium look colourful, and for this reason, you want to divide the spectators into groups, and then ask each group to sit in one section. Obviously, the Camp Nou is a very large stadium, so there are enough sections available. The spectators are in a line outside the stadium, and each group needs to be a contiguous subsegment of that line.

However, you also know that people naturally want to sit with others who support the same team as them. A spectator will be unhappy if nobody else in their section supports the same team as them. You need to ensure that there are at most K unhappy people in each section.

How many different ways are there for you to assign spectators to groups, which satisfy these constraints? Since the answer may be large, output the answer modulo 998244353.

Input
The first line of input contains two space-separated integers, N and K. N is the number of total spectators, and K is the maximum number of unhappy people you can put in one section.

The second line of input contains N space-separated integers, the i-th of which represents the team the i-th spectator supports.

Output
Output a single integer, the number of ways to assign spectators to groups, modulo 998244353.

Scoring
The input is divided into multiple subtasks. For all subtasks, $1 \le N \le 99354$ and $1 \le K \le N$. All spectators support a team with number between 1 and N inclusive.

- Subtask 1 [11 points]: $K = N$
- Subtask 2 [3 points]: All array values are 1.
- Subtask 3 [7 points]: The array values are at most 2.
- Subtask 4 [19 points]: The array values are at most 10.
- Subtask 5 [13 points]: $N \le 500$
- Subtask 6 [12 points]: $N \le 3000$
- Subtask 7 [14 points]: $K = 1$
- Subtask 8 [21 points]: No additional constraints.

input
5 1 3 4 2 3 1
output
1

input
5 1 1 1 1 1 1
output
16

input
5 5 2 4 5 1 5
output
16

input
5 2 1 1 2 1 3
output
14

input
3 1 1 1 2
output
3