R. KARTHIKA
953622104045

MINI PROJECT

Aim:

To determine the graphs, Probability Theory and Cakes. The goal might be to calculate the probability of a person starting from a particular location reaching another location in some specific location.

Program Statement:

chef is viewing a simulation of creating a tree with N vertices (numbered 1 through N) which calculates mysterious coefficients for all the edges. initially, there is a graph without edges, but there is a set S containing N-1 edges(numbered 1 through N-1) which should get added to the graph one by one.

In this process, N-1 times, the following steps are performed:
An edge e is chosen uniformly randomly from the current set S.
This edge is added to the graph and removed from S.
The coefficient of e is set to the number of vertices in the connected component created by adding this edge.
Chef is wondering about the expected values of coefficients of the edges. Unfortunately, he does not have much time for such matter since he needs to prepare a superb cake for Chefina's birthday tomorrow.
Help chef and for each edge, calculate the expected value of its coefficients modulo 998,244,353. We can prove that each coefficient is a rational number; for a coefficient in the form P/Q where P and Q are positive integers and Q is coprime with 998,244,353, you should compute p. Q^-1 modulo 998,244,353 where Q^-1 is the multiplicative inverse of Q modulo 998,244,353.
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 following N-1 lines describe the edges in S. For each valid i, the i-th of these lines contains two space separated integers u and v denoting that the i-th edge connects vertices u and v.**

Output:

R. KARTHIKA
953622104045

For each test case, print a single line containing N-1 integers. for each valid i, the i-th of these integers should be R. Q^-1 modulo 998,244,353 for the coefficient of the i-th edge.

Constraints:

- **2<=T<=10,000**
- **2<=N<=50,000**
- **1<=u,v<=N**
- **the edges in S form a tree**
- **the sum of N over all test cases does not exceed 200,000**

Program Code:

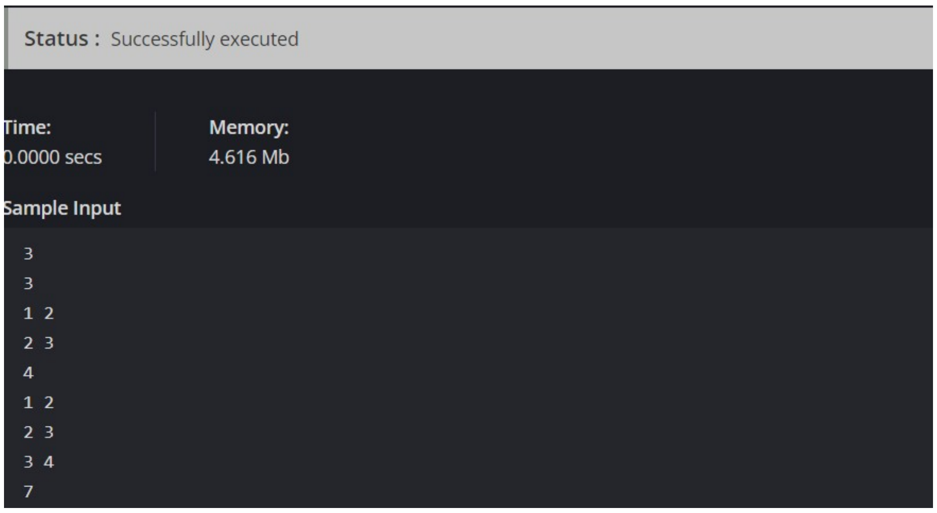
```
#include <stdlib.h>
#define MOD 998244353
int *parent, *componentSize;
int findSet(int v)
{
   if (v == parent[v]) return v;
   return parent[v] = findSet(parent[v]);
}
void unionSets(int a, int b)
{
   a = findSet(a);
   b = findSet(b);
   if (a != b)
  {
      if (componentSize[a] < componentSize[b])
     {
         int temp = a;
         a = b;
         b = temp;
      }
      parent[b] = a;
```

```
        componentSize[a] += componentSize[b];
    }
}

int main()
{
    int T;
    scanf("%d", &T);
   while (T--)
   {
        int N;
        scanf("%d", &N);
        parent = (int *)malloc((N + 1) * sizeof(int));
        componentSize = (int *)malloc((N + 1) * sizeof(int));

        for (int i = 1; i <= N; ++i) {
            parent[i] = i;
            componentSize[i] = 1;
        }
        int *result = (int *)malloc((N - 1) * sizeof(int));
        int edges[N - 1][2];
        for (int i = 0; i < N - 1; ++i)
       {
            scanf("%d %d", &edges[i][0], &edges[i][1]);
        }
        for (int i = N - 2; i >= 0; --i)
       {
            int u = edges[i][0];
            int v = edges[i][1];
            int sizeU = componentSize[findSet(u)];
            int sizeV = componentSize[findSet(v)];
            result[i] = (sizeU * 1LL * sizeV) % MOD;
            unionSets(u, v);
        }
        for (int i = 0; i < N - 1; ++i)
        {
            int inverse = 1;
            for (int j = 1; j <= result[i]; ++j)
            {
                inverse = (inverse * 1LL * j) % MOD;
            }
            int modInverse = 1;
          int exponent = MOD - 2;
          while (exponent > 0)
          {
                if (exponent % 2 == 1)
                {
                    modInverse = (modInverse * 1LL * inverse) % MOD;
                }
                inverse = (inverse * 1LL * inverse) % MOD;
                exponent /= 2;
            }


            printf("%d ", modInverse);
        }
        printf("\n");
        free(parent);
        free(componentSize);
        free(result);
    }
        return 0;
}
```

Explanation:

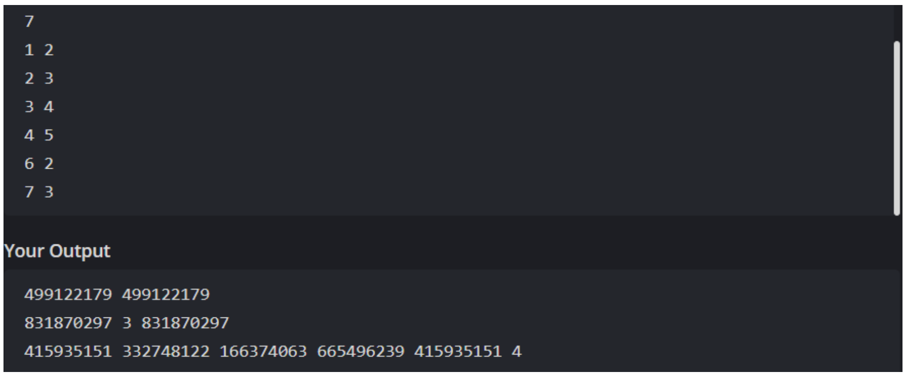
The code is designed to process multiple test cases, each representing a different graph. The code defines a ‘MOD’ with the value 998244353. This function is used to perform operations modulo this value. The code declares two global integer arrays ‘parent and componentsize’. It allocates memory for an array result of size ‘N-1’ which will store the results of calculations for each edge in the graph. It also reads the edges of the graph into a 2D array edges of size 'N-1' by 2 and each row represents an edge connecting two nodes.

Output:



```
Status : Successfully executed

Time:            Memory:
0.0000 secs      4.616 Mb

Sample Input
3
3
1 2
2 3
4
1 2
2 3
3 4
7
```

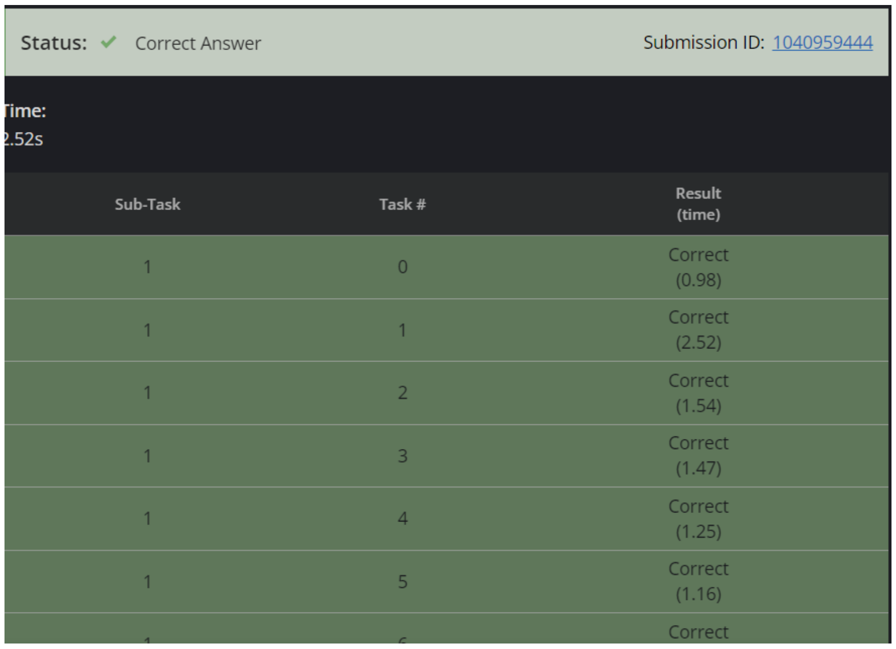
Fig 1.1: Run Command



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

Your Output
499122179 499122179
831870297 3 831870297
415935151 332748122 166374063 665496239 415935151 4
```

Fig 1.2 : Run Command

R. KARTHIKA
953622104045


Status: ✓ Correct Answer   Submission ID: 1040959444

ime:
.52s

Sub-Task	Task #	Result (time)
1	0	Correct (0.98)
1	1	Correct (2.52)
1	2	Correct (1.54)
1	3	Correct (1.47)
1	4	Correct (1.25)
1	5	Correct (1.16)
1	6	Correct

Conclusion:

Thus the program using graphs, probability theory and cakes were executed successfully.