Birthday Assignment

locked

by ma5termind

Editorial by ma5termind

Explanation

Disclaimer: This was the hardest problem of this contest and i noticed it's little bit hard for me to explain. I will try my best, please bear with me.

By now, you might have got the problem is all about finding the number of topological sortings in the randomly directed tree T.

Let randomly root the tree T at any node say 1. Lets calculate a $DP_(i, j)$ denoting the number of topological sorts of subtree rooted at i^{th} node such that i^{th} node is placed at j^{th} position. Assume that there are X outgoing edges to the children from i^{th} and Y incoming edges. Note that all the X children has to be placed on the left and all the Y children has to be placed on the right side of i^{th} in a valid topological sort.

Now lets calculate another $dp_(i, j)$ for those X nodes. Similiarly, we will do for Y nodes. Here $dp(i, j)$ denotes the number of ways of filling j spaces using topological sorts of subtrees rooted at $X_1, X_2, \ldots X_i$ such that $X_1, X_2, X_3, \ldots X_i$ is placed in j spaces. Note that to compute this dp, we can use the DP already calculated at these children nodes.

$$dp(i, j) = \sum_{1 \le k \le |S_i|} \left(nCr(j, k) \times dp(i-1, j-k) \times \sum_{1 \le l \le k} DP(i, l) \right)$$

Note that this $\sum_{1 \le l \le k} DP(i, l)$ is just a prefix sum and can be precomputed.

Now, for X nodes basically you have computed dp_j, number of filling up j spaces using subtrees of X nodes such that $X_1, X_2, \ldots . X_x$ is included. Lets call this dp as $left[]$ and similary call the dp for Y nodes as $right[]$.

$$DP_{(i,j)} = \sum_{1 \le k < j} nCr(j-1, k) \times left_k \times nCr(SZ_i - j, SZ_X - k) \times right_{SZ_i - j - (SZ_x - k)}$$

Where SZ_i denotes the size of subtree rooted at i^{th} node & SZ_X denotes the sum of sizes of subtrees rooted at all those X nodes.

Please have a look at author's solution for better understanding.

Set by ma5termind

Problem Setter's code :

Statistics

Difficulty:	Hard
Time Complexity:	O(N^2) Required
Knowledge:	Dynamic programming on trees, Combinatronics
Publish Date:	Aug 01 2018

```
#include <stdio.h>
#include <cassert>
#include <iostream>
#include <vector>
#include <cmath>
#include <algorithm>
#include <memory.h>
#include <map>
#include <set>
#include <queue>
#include <list>
#include <sstream>
#include <cstring>
using namespace std ;

#define ft first
#define sd second
#define pb push_back
#define all(x) x.begin(),x.end()

#define ll long long int
#define vi vector<int>
#define vii vector<pair<int,int> >
#define pii pair<int,int>
#define plii pair<pair<ll, int>, int>
#define piii pair<pii, int>
#define viii vector<pair<pii, int> >
#define vl vector<ll>
#define vll vector<pair<ll,ll> >
#define pll pair<ll,ll>
#define pli pair<ll,int>
#define mp make_pair
#define ms(x, v) memset(x, v, sizeof x)

#define sc1(x) scanf("%d",&x)
#define sc2(x,y) scanf("%d%d",&x,&y)
#define sc3(x,y,z) scanf("%d%d%d",&x,&y,&z)

#define scll1(x) scanf("%lld",&x)
#define scll2(x,y) scanf("%lld%lld",&x,&y)
#define scll3(x,y,z) scanf("%lld%lld%lld",&x,&y,&z)

#define pr1(x) printf("%d\n",x)
#define pr2(x,y) printf("%d %d\n",x,y)
#define pr3(x,y,z) printf("%d %d %d\n",x,y,z)

#define prll1(x) printf("%lld\n",x)
#define prll2(x,y) printf("%lld %lld\n",x,y)
#define prll3(x,y,z) printf("%lld %lld %lld\n",x,y,z)

#define pr_vec(v) for(int i=0;i<v.size();i++) cout << v[i] << " " ;

#define f_in(st) freopen(st,"r",stdin)
#define f_out(st) freopen(st,"w",stdout)

#define fr(i, a, b) for(i=a; i<=b; i++)
#define fb(i, a, b) for(i=a; i>=b; i--)
#define ASST(x, l, r) assert( x <= r && x >= l )

const int mod = 1e9 + 7;

int ADD(int a, int b, int m = mod) {
    int s = a;
    s += b;
    if( s >= m )
      s -= m;
    return s;
}

int MUL(int a, int b, int m = mod) {
    return (1LL * a * b % m);
}
```

```
int power(int a, int b, int m = mod) {
    int res = 1;
    while( b ) {
        if( b & 1 ) {
            res = 1LL * res * a % m;
        }
        a = 1LL * a * a % m;
        b /= 2;
    }
    return res;
}

ll nC2(ll x) {
    return ( x * ( x - 1 ) / 2 );
}

const int maxn = 5 * 1000 + 10;

vi adj[maxn];
map<pii, bool> dir;
int dp[maxn][maxn], sz[maxn], NcR[maxn][maxn];

int nCr(int n, int r) {
    if(r > n) return 0;
    if(n == 0 || r == 0) return 1;
    int &ret = NcR[n][r];
    if(ret != -1) return ret;
    ret = 0;
    ret = nCr(n-1, r-1) + nCr(n-1, r);
    if( ret >= mod ) ret -= mod;
    return ret;
}

void dfs(int u, int p = -1) {
    sz[u] = 1;
    int total_left = 0, total_right = 0;
    for( auto it: adj[u] ) {
        if(it != p) {
            dfs(it, u);
            sz[u] += sz[it];
            if(dir[mp(u, it)]) {
                total_left += sz[it];
                int i; fr(i, 1, sz[it]) {
                    dp[it][i] += dp[it][i-1];
                    if(dp[it][i] >= mod) dp[it][i] -= mod;
                }
            } else {
                total_right += sz[it];
                int i; fb(i, sz[it]-1, 1) {
                    dp[it][i] += dp[it][i+1];
                    if(dp[it][i] >= mod) dp[it][i] -= mod;
                }
            }
        }
    }
    if(sz[u] == 1) { dp[u][1] = 1;}
    else {
        int left[2][total_left+1], right[2][total_right+1];
        ms(left, 0); ms(right, 0);
        int left_p = 0, left_n = 1, right_p = 0, right_n = 1;
        left[left_p][0] = right[right_p][0] = 1;
        int left_count = total_left;
        int right_count = total_right;
        total_left = total_right = 0;
        for( auto it: adj[u] ) {
            if(it != p) {
                if(dir[mp(u, it)]) {
                    total_left += sz[it];
                    int i, j;
                    fr(i, 0, left_count) {
                        if(left[left_p][i]) {
                            fr(j, 1, sz[it]) {
                                int v = 1;
                                v = 1LL * nCr(i+j, j) * dp[it][j] % mod;
                                v = 1LL * v * left[left_p][i] % mod;
                                v = 1LL * v * nCr(total_left - (i+j), sz[it
] - j) % mod;
                                left[left_n][i+j] += v;
                                if(left[left_n][i+j] >= mod) left[left_n][i
+j] -= mod;
                            }
                            left[left_p][i] = 0;
                        }
                    }
                    swap(left_p, left_n);
                } else {
                    total_right += sz[it];
                    int i, j; fr(i, 0, right_count) {
                        if(right[right_p][i]) {
                            fr(j, 1, sz[it]) {
                                int v = 1;
                                v = 1LL * nCr(i+j, j) * dp[it][sz[it] - j +
 1] % mod;
                                v = 1LL * v * right[right_p][i] % mod;
                                v = 1LL * v * nCr(total_right - i - j, sz[i
t] - j) % mod;
                                right[right_n][i+j] += v;
                                if(right[right_n][i+j] >= mod) right[right_
n][i+j] -= mod;
                            }
                            right[right_p][i] = 0;
                        }
                    }
                    swap(right_p, right_n);
                }
            }
        }
        total_left = left_count;
        total_right = right_count;
        int i, j;
        fr(i, 1, sz[u]) {
            dp[u][i] = 0;
            fr(j, 0, min(i-1, total_left)) {
                int x = total_left - j;
                int v = left[left_p][j] % mod;
                v = 1LL * v * nCr(i-1, j) % mod;
                v = 1LL * v * nCr(sz[u] - i, x) % mod;
                if(sz[u] - i - x >= 0 && sz[u] - i - x <= total_right)
                    v = 1LL * v * right[right_p][sz[u] - i - x] % mod;
                else
                    v = 0;
                dp[u][i] += v;
                if(dp[u][i] >= mod) dp[u][i] -= mod;
            }
        }
    }
}

void solve() {
    int n, m; cin >> n >> m;
    int i, j;
    fr(i, 1, n-1) {
        int x, y;
        cin >> x >> y;
        dir[mp(x, y)] = 1;
        dir[mp(y, x)] = 0;
        adj[x].pb( y );
        adj[y].pb( x );
    }
    int ans = 0, mult = 1;
    dfs(1);
    fr(i, 1, n) {
        // cout << dp[1][i] << " ";
        mult = 1LL * mult * m % mod; m --;
        mult = 1LL * mult * power(i, mod-2) % mod;
        ans += dp[1][i]; ans %= mod;
    }
    // cout << "\n";
    cout << 1LL * ans * mult % mod << "\n";
    fr(i, 1, n) {
            adj[i].clear();
            fr(j, 1, n) dp[i][j] = 0;
        }
        dir.clear();
}
int main() {
    ms(NcR, -1);
    int t;  cin >> t;
    while( t-- ) solve();
    return 0;
}
```

Tested by bayleef

Problem Tester's code :

```
using System;
using System.IO;
using System.Collections.Generic;

namespace CSharpParser
{
    public class Solution : SolutionBase
    {
        private static readonly int[,] c = new int[1001, 1001];
        private static void Rec(int i, int pr, List<int>[] left, List<int>
[] right, int[][] dp)
        {
            const int mod = 1000000007;
            var dl = new int[1];
            dl[0] = 1;
            foreach (var j in left[i])
            {
                if (j == pr) continue;
                Rec(j, i, left, right, dp);
                var temp = new int[dl.Length + dp[j].Length];
                var sdp = 0;
                for (var k = 0; k < dp[j].Length; k++)
                {
                    sdp = (sdp + dp[j][k]) % mod;
                    for (var l = 0; l < dl.Length; l++)
                        temp[k + l + 1] = (int)((temp[k + l + 1] + (long)dl
[l] * sdp % mod * c[l + k + 1, l] % mod * c[dp[j].Length - k - 1 + dl.Lengt
h - l - 1, dl.Length - l - 1]) % mod);
                }
                dl = temp;
            }

            var dr = new int[1];
            dr[0] = 1;
            foreach (var j in right[i])
            {
                if (j == pr) continue;
                Rec(j, i, left, right, dp);
                var temp = new int[dr.Length + dp[j].Length];
                var sdp = 0;
                for (var k = 0; k < dp[j].Length; k++)
                {
                    sdp = (sdp + dp[j][dp[j].Length - 1 - k]) % mod;
                    for (var l = 0; l < dr.Length; l++)
                        temp[k + l + 1] = (int)((temp[k + l + 1] + (long)dr
[l] * sdp % mod * c[l + k + 1, l] % mod * c[dp[j].Length - k - 1 + dr.Lengt
h - l - 1, dr.Length - l - 1]) % mod);
                }
                dr = temp;
            }

            dp[i] = new int[dl.Length + dr.Length - 1];
            for (var ll = 0; ll < dl.Length; ll++)
                for (var rr = 0; rr < dr.Length; rr++)
                {
                    var lr = dl.Length - 1 - ll;
                    var rl = dr.Length - 1 - rr;
                    dp[i][ll + rl] = (int)((dp[i][ll + rl] + (long)dl[ll] *
 dr[rr] % mod * c[ll + rl, ll] % mod * c[lr + rr, rr]) % mod);
                }
        }

        protected override void Solve()
        {
            const int mod = 1000000007;
            for (var i = 0; i <= 1000; i++)
            {
                c[i, 0] = 1;
                for (var j = 1; j <= i; j++)
                    c[i, j] = (c[i - 1, j] + c[i - 1, j - 1]) % mod;
            }
            var o = new int[1001];
            o[1] = 1;
            for (var i = 2; i < o.Length; i++)
                o[i] = (int)((mod - mod / i) * (long)o[mod % i] % mod);
            Next(out int T);
            if(T<1 || T>5) throw new Exception();
            while (T-- > 0)
            {
                Next(out int n);
                Next(out int m);
                if(n<1 || n>1000)throw new Exception();
                if(m<1 || m>1000000000)throw new Exception();
                var left = new List<int>[n];
                var right = new List<int>[n];
                left.Fill(temp => new List<int>());
                right.Fill(temp => new List<int>());
                for (var k = 1; k < n; k++)
                {
                    Next(out int i);
                    Next(out int j);
                    --i;
                    --j;
                    left[i].Add(j);
                    right[j].Add(i);
                }
                var dp = new int[n][];
                Rec(0, -1, left, right, dp);
                var ans = 0;
                for (var i = 0; i < dp[0].Length; i++)
                    ans = (ans + dp[0][i]) % mod;
                for (var i = 0; i < n; i++)
                    ans = (int)((long)ans * (m - i) % mod * o[i + 1] % mod
);
                PrintLine(ans);
            }
        }
    }

    public static class Algorithm
    {
        private static readonly Random Rnd = new Random();

        public static void Swap<T>(ref T a, ref T b)
        {
            var temp = a;
            a = b;
            b = temp;
        }

        public static T Max<T>(params T[] a)
        {
            var ans = a[0];
            var comp = Comparer<T>.Default;
            for (var i = 1; i < a.Length; i++) ans = comp.Compare(ans, a[i
]) >= 0 ? ans : a[i];
            return ans;
        }

        public static T Min<T>(params T[] a)
        {
            var ans = a[0];
            var comp = Comparer<T>.Default;
            for (var i = 1; i < a.Length; i++) ans = comp.Compare(ans, a[i
]) <= 0 ? ans : a[i];
            return ans;
        }

        public static void RandomShuffle<T>(IList<T> a, int index, int leng
th)
        {
            if (index < 0 || length < 0) throw new ArgumentOutOfRangeExcept
ion();
            var last = index + length;
            if (last > a.Count) throw new ArgumentException();
            for (var i = index + 1; i < last; i++)
            {
                var j = Rnd.Next(index, i + 1);
                var t = a[i];
                a[i] = a[j];
                a[j] = t;
            }
        }

        public static void RandomShuffle<T>(IList<T> a)
        {
            RandomShuffle(a, 0, a.Count);
        }

        public static bool NextPermutation<T>(IList<T> a, int index, int le
ngth, Comparison<T> compare = null)
        {
            compare = compare ?? Comparer<T>.Default.Compare;
            if (index < 0 || length < 0) throw new ArgumentOutOfRangeExcept
ion();
            var last = index + length;
            if (last > a.Count) throw new ArgumentException();
            for (var i = last - 1; i > index; i--)
                if (compare(a[i], a[i - 1]) > 0)
                {
                    var j = i + 1;
                    for (; j < last; j++) if (compare(a[j], a[i - 1]) <= 0)
 break;
                    var t = a[i - 1];
                    a[i - 1] = a[j - 1];
                    a[j - 1] = t;
                    for (; i < last - 1; i++, last--)
                    {
                        t = a[i];
                        a[i] = a[last - 1];
                        a[last - 1] = t;
                    }
                    return true;
                }
            for (var i = index; i < last - 1; i++, last--)
            {
                var t = a[i];
                a[i] = a[last - 1];
                a[last - 1] = t;
            }
            return false;
        }

        public static bool NextPermutation<T>(IList<T> a, Comparison<T> com
pare = null)
        {
            return NextPermutation(a, 0, a.Count, compare);
        }

        public static bool PrevPermutation<T>(IList<T> a, int index, int le
ngth, Comparison<T> compare = null)
        {
            compare = compare ?? Comparer<T>.Default.Compare;
            if (index < 0 || length < 0) throw new ArgumentOutOfRangeExcept
ion();
```

```
            var last = index + length;
            if (last > a.Count) throw new ArgumentException();
            for (var i = last - 1; i > index; i--)
                if (compare(a[i], a[i - 1]) < 0)
                {
                    var j = i + 1;
                    for (; j < last; j++) if (compare(a[j], a[i - 1]) >= 0)
 break;
                    var t = a[i - 1];
                    a[i - 1] = a[j - 1];
                    a[j - 1] = t;
                    for (; i < last - 1; i++, last--)
                    {
                        t = a[i];
                        a[i] = a[last - 1];
                        a[last - 1] = t;
                    }
                    return true;
                }
            for (var i = index; i < last - 1; i++, last--)
            {
                var t = a[i];
                a[i] = a[last - 1];
                a[last - 1] = t;
            }
            return false;
        }

        public static bool PrevPermutation<T>(IList<T> a, Comparison<T> com
pare = null)
        {
            return PrevPermutation(a, 0, a.Count, compare);
        }

        public static int LowerBound<T>(IList<T> a, int index, int length,
T value, Comparison<T> compare = null)
        {
            compare = compare ?? Comparer<T>.Default.Compare;
            if (index < 0 || length < 0) throw new ArgumentOutOfRangeExcept
ion();
            if (index + length > a.Count) throw new ArgumentException();
            var ans = index;
            var last = index + length;
            var p2 = 1;
            while (p2 <= length) p2 *= 2;
            for (p2 /= 2; p2 > 0; p2 /= 2) if (ans + p2 <= last && compare(
a[ans + p2 - 1], value) < 0) ans += p2;
            return ans;
        }

        public static int LowerBound<T>(IList<T> a, T value, Comparison<T>
compare = null)
        {
            return LowerBound(a, 0, a.Count, value, compare);
        }

        public static int UpperBound<T>(IList<T> a, int index, int length,
T value, Comparison<T> compare = null)
        {
            compare = compare ?? Comparer<T>.Default.Compare;
            if (index < 0 || length < 0) throw new ArgumentOutOfRangeExcept
ion();
            if (index + length > a.Count) throw new ArgumentException();
            var ans = index;
            var last = index + length;
            var p2 = 1;
            while (p2 <= length) p2 *= 2;
            for (p2 /= 2; p2 > 0; p2 /= 2) if (ans + p2 <= last && compare(
a[ans + p2 - 1], value) <= 0) ans += p2;
            return ans;
        }

        public static int UpperBound<T>(IList<T> a, T value, Comparison<T>
compare = null)
        {
            return UpperBound(a, 0, a.Count, value, compare);
        }

        public static void Fill<T>(this IList<T> array, T value) where T :
struct
        {
            for (var i = 0; i < array.Count; i++)
                array[i] = value;
        }

        public static void Fill<T>(this IList<T> array, Func<int, T> func)
        {
            for (var i = 0; i < array.Count; i++)
                array[i] = func(i);
        }
    }

    public class InStream : IDisposable
    {
        protected readonly TextReader InputStream;
        private string[] _tokens;
        private int _pointer;

        private InStream(TextReader inputStream)
        {
            InputStream = inputStream;
        }

        public static InStream FromString(string str)
        {
            return new InStream(new StringReader(str));
        }

        public static InStream FromFile(string str)
        {
            return new InStream(new StreamReader(str));
        }

        public static InStream FromConsole()
        {
            return new InStream(Console.In);
        }

        public string NextLine()
        {
            try
            {
                return InputStream.ReadLine();
            }
            catch (Exception)
            {
                return null;
            }
        }

        private string NextString()
        {
            try
            {
                while (_tokens == null || _pointer >= _tokens.Length)
                {
                    _tokens = NextLine().Split(new[] { ' ', '\t' }, StringS
plitOptions.RemoveEmptyEntries);
                    _pointer = 0;
                }
                return _tokens[_pointer++];
            }
            catch (Exception)
            {
                return null;
            }
        }

        public bool Next<T>(out T ans)
        {
            var str = NextString();
            if (str == null)
            {
                ans = default(T);
                return false;
            }
            ans = (T)Convert.ChangeType(str, typeof(T));
            return true;
        }

        public T[] NextArray<T>(int length)
        {
            var array = new T[length];
            for (var i = 0; i < length; i++)
                if (!Next(out array[i]))
                    return null;
            return array;
        }

        public T[,] NextArray<T>(int length, int width)
        {
            var array = new T[length, width];
            for (var i = 0; i < length; i++)
                for (var j = 0; j < width; j++)
                    if (!Next(out array[i, j]))
                        return null;
            return array;
        }

        public void Dispose()
        {
            InputStream.Close();
        }
    }

    public class OutStream : IDisposable
    {
        protected readonly TextWriter OutputStream;

        private OutStream(TextWriter outputStream)
        {
            OutputStream = outputStream;
        }

        public static OutStream FromString(System.Text.StringBuilder strB)
        {
            return new OutStream(new StringWriter(strB));
        }

        public static OutStream FromFile(string str)
        {
            return new OutStream(new StreamWriter(str));
        }

        public static OutStream FromConsole()
        {
            return new OutStream(Console.Out);
        }

        public void Print(string format, params object[] args)
        {
            OutputStream.Write(format, args);
        }

        public void PrintLine(string format, params object[] args)
        {
            Print(format, args);
            OutputStream.WriteLine();
        }

        public void PrintLine()
        {
            OutputStream.WriteLine();
        }
        public void Print<T>(T o)
        {
            OutputStream.Write(o);
        }

        public void PrintLine<T>(T o)
        {
            OutputStream.WriteLine(o);
        }

        public void PrintArray<T>(IList<T> a, string between = " ", string
after = "\n", bool printCount = false)
        {
            if (printCount)
                PrintLine(a.Count);
            for (var i = 0; i < a.Count; i++)
                Print("{0}{1}", a[i], i == a.Count - 1 ? after : between);
        }

        public void Dispose()
        {
            OutputStream.Close();
        }
    }

    public abstract class SolutionBase : IDisposable
    {
        private InStream _in;
        private OutStream _out;

        protected SolutionBase()
        {
            //System.Threading.Thread.CurrentThread.CurrentCulture = Syste
m.Globalization.CultureInfo.InvariantCulture;
            _in = InStream.FromConsole();
            _out = OutStream.FromConsole();
        }

        protected string NextLine()
        {
            return _in.NextLine();
        }

        protected bool Next<T>(out T ans)
        {
            return _in.Next(out ans);
        }

        protected T[] NextArray<T>(int length)
        {
            return _in.NextArray<T>(length);
        }

        protected T[,] NextArray<T>(int length, int width)
        {
            return _in.NextArray<T>(length, width);
        }

        protected void PrintArray<T>(IList<T> a, string between = " ", stri
ng after = "\n", bool printCount = false)
        {
            _out.PrintArray(a, between, after, printCount);
        }

        public void Print(string format, params object[] args)
        {
            _out.Print(format, args);
        }

        public void PrintLine(string format, params object[] args)
        {
            _out.PrintLine(format, args);
        }

        public void PrintLine()
        {
            _out.PrintLine();
        }

        public void Print<T>(T o)
        {
            _out.Print(o);
        }

        public void PrintLine<T>(T o)
        {
            _out.PrintLine(o);
        }

        public void Dispose()
        {
            _in.Dispose();
            _out.Dispose();
        }

        public void Freopen(string path, FileAccess access)
        {
            switch (access)
            {
                case FileAccess.Read:
                    _in.Dispose();
                    _in = InStream.FromFile(path);
                    break;
                case FileAccess.Write:
                    _out.Dispose();
                    _out = OutStream.FromFile(path);
                    break;
            }
        }

        protected abstract void Solve();

        public static void Main()
        {
            using (var p = new Solution()) p.Solve();
        }
    }
}
```