

D. Diameter Counting

题意

求所有 n 标号树的直径和。

题解1

考虑求树的直径的过程，可以先删去所有叶子结点，得到一棵新树，称为一次操作。然后再不断对新树进行操作，知道最后剩下一个或两个结点。

此时如果只剩下一个结点，则树的直径为操作次数 $\times 2$ ；如果剩下两个结点，则树的直径为操作次数 $\times 2 + 1$ 。

考虑通过逆算法反向构建树。设 $f(i,j)$ 表示有 j 个叶子结点的 i 标号树个数，假设上一步操作删除了 $k(k \geq j)$ 个叶子。

于是问题等价于给这 k 个叶子找一个父结点，使得原来的 j 个叶子结点至少有一个儿子。

同时对于这 $i+k$ 个结点，标号是任意的，对所有 $i+k$ 标号树而言，删去 k 叶子结点得到的树的标号方式实际上有 $\binom{i+k}{i} f(i,j)$ 种。

设 $g(i,j,k)$ 表示长度为 k 且每个位置有 i 种可取值且特定的 j 个值至少出现一次的序列个数，于是有

$$f(i+k,k) = g(i,j,k) \binom{i+k}{i}$$

接下来考虑求 $g(i,j,k)$ 。可以考虑序列前 $k-1$ 位，如果此时 j 个特定值都出现了至少一次，则第 k 位可以任取，于是有

$$g(i,j,k) = i \times g(i,j,k-1)$$

如果前 $k-1$ 位只有 $j-1$ 个特殊值出现了至少一次，则显然前 $k-1$ 位的取值只有 $i-1$ 种，同时要从 j 个特殊值中确定一个放在第 k 位，有

$$g(i,j,k) = j \times g(i-1,j-1,k-1)$$

最后设 $h(i,j)$ 表示有 j 个叶子的 i 标号树的直径之和。同样假设上一步操作删除了 $k(k \geq j)$ 个叶子。

考虑原有的树的直径和操作带来的直径 $+2$ 的新贡献，于是有

$$h(i+k,k) = g(i,j,k) \binom{i+k}{i} (h(i,j) + 2f(i,j))$$

另外所有 $h(i+k,k) = 2f(i,j)g(i,j,k) \binom{i+k}{i}$ 也可以等价于 $h(i,j) = 2f(i,j)$ 。时空复杂度 $O(n^3)$ 。

```
const int MAXN=505;
int mod;
int quick_pow(int n,int k){
```

```
int ans=1;
while(k){
    if(k&1)ans=1LL*ans*n%mod;
    n=1LL*n*n%mod;
    k>>=1;
}
return ans;
}
int frac[MAXN],invf[MAXN];
int C(int n,int m){
    return 1LL*frac[n]*invf[m]%mod*invf[n-m]%mod;
}
int f[MAXN][MAXN],g[MAXN][MAXN][MAXN],h[MAXN][MAXN];
int main()
{
    int n=read_int();
    mod=read_int();
    frac[0]=1;
    _for(i,1,MAXN)frac[i]=1LL*frac[i-1]*i%mod;
    invf[MAXN-1]=quick_pow(frac[MAXN-1],mod-2);
    for(int i=MAXN-1;i;i--)
    invf[i-1]=1LL*invf[i]*i%mod;
    g[0][0][0]=1;
    _rep(i,1,n){
        g[i][0][0]=1;
        _rep(k,1,n)
        g[i][0][k]=1LL*g[i][0][k-1]*i%mod;
        _rep(j,1,i)_rep(k,j,n)
        g[i][j][k]=(1LL*g[i][j][k-1]*i+1LL*g[i-1][j-1][k-1]*j)%mod;
    }
    f[1][1]=f[2][2]=1;
    _rep(i,1,n)_rep(j,1,i)_rep(k,max(j,2),n-i)
    f[i+k][k]=(f[i+k][k]+1LL*f[i][j]*g[i][j][k]%mod*C(i+k,i))%mod;
    h[2][2]=1;
    _rep(i,3,n)_rep(j,1,i)
    h[i][j]=2LL*f[i][j]%mod;
    _rep(i,1,n)_rep(j,1,i)_rep(k,max(j,2),n-i)
    h[i+k][k]=(h[i+k][k]+1LL*h[i][j]*g[i][j][k]%mod*C(i+k,i))%mod;
    int ans=0;
    _rep(i,1,n)
    ans=(ans+h[n][i])%mod;
    enter(ans);
    return 0;
}
```

题解2

设 $f(i,j)$ 表示深度为 j 的 i 号标号树个数 $g(i,j)$ 表示深度不超过 j 的 i 号标号树个数。

对于一个直径为 $2d+1$ 的 n 号无根树，可以沿着直径的中心边切开，得到两个深度为 d 的有根树。

设 i 号点所在的有根树大小为 i ，考虑为他分配 $i-1$ 个编号，于是直径为 $2d+1$ 的 n 号无根树个数为

$$\sum_{i=1}^{n-1} f(i,d) f(n-i,d) \binom{n-1}{i-1}$$

对于一个直径为 $2d$ 的 n 号无根树，可以认为是根节点连接至少两个深度等于 $d-1$ 的有根树。

利用容斥，用深度为 d 的 n 号有根树个数减去根节点仅连接一个深度为 $d-1$ 的有根树个数的情况。

根节点仅连接一个深度为 $d-1$ 的有根树可以认为是由一个深度不超过 $d-1$ 的有根树根节点连接一个深度等于 $d-1$ 的有根树得到的。

于是直径为 $2d$ 的 n 号无根树个数为

$$f(n,d) - \sum_{i=1}^{n-1} f(i,d-1) g(n-i,d-1) \binom{n}{i}$$

接下来考虑计算 $f(i,j), g(i,j)$ ，难以直接计算，但显然有 $f(i,j) = g(i,j) - g(i,j-1)$ ，于是只需要计算 $g(i,j)$

对于深度不超过 j 的 i 号树个数，可以先分配一个编号给根节点，然后从与根节点相连的子树中找到编号最小的结点所在的子树。

设子树大小为 k ，对该子树，他深度不超过 $j-1$ ，显然有 $g(k,j-1)$ 种。另外需要从剩余 $i-2$ 个编号再分配 $k-1$ 个编号给子树。

对于余下的 $n-k$ 个点，深度仍然不超过 j ，但根节点编号已经分配，所以总数为 $\frac{g(i-k,j)}{i-k}$ ，于是有

$$g(i,j) = \sum_{k=1}^{i-1} \binom{i-2}{k-1} g(k,j-1) \frac{g(i-k,j)}{i-k}$$

时间复杂度 $O(n^3)$ ，空间复杂度 $O(n^2)$

```

const int MAXN=505;
int mod;
int quick_pow(int n,int k){
    int ans=1;
    while(k){
        if(k&1) ans=1LL*ans*n%mod;
        n=1LL*n*n%mod;
        k>>=1;
    }
    return ans;
}
int frac[MAXN], invf[MAXN], inv[MAXN];
int C(int n,int m){
    return 1LL*frac[n]*invf[m]%mod*invf[n-m]%mod;
}
int f[MAXN][MAXN], g[MAXN][MAXN];
int main()
{

```

```
int n=read_int();
mod=read_int();
frac[0]=1;
_for(i,1,MAXN) frac[i]=1LL*frac[i-1]*i%mod;
invf[MAXN-1]=quick_pow(frac[MAXN-1],mod-2);
for(int i=MAXN-1;i;i--){
    invf[i-1]=1LL*invf[i]*i%mod;
    inv[i]=1LL*invf[i]*frac[i-1]%mod;
}
g[1][0]=1;
_rep(i,1,n){
    _for(j,1,i)_for(k,1,i)
        g[i][j]=(g[i][j]+1LL*g[k][j-1]*g[i-
k][j]%mod*C(i-2,k-1)%mod*i%mod*inv[i-k])%mod;
    _rep(j,i,n)
        g[i][j]=g[i][i-1];
}
f[1][0]=1;
_rep(i,2,n)_for(j,1,i)
    f[i][j]=(g[i][j]-g[i][j-1])%mod;
int ans=0;
_for(i,1,n){
    int cnt=0,d=i/2;
    if(i&1){
        _for(j,1,n)
            cnt=(cnt+1LL*f[j][d]*f[n-j][d]%mod*C(n-1,j-1))%mod;
    }
    else{
        cnt=f[n][d];
        _for(j,1,n)
            cnt=(cnt-1LL*f[j][d-1]*g[n-j][d-1]%mod*C(n,j))%mod;
    }
    ans=(ans+1LL*cnt*i)%mod;
}
if(ans<0)ans+=mod;
enter(ans);
return 0;
}
```

From: <https://wiki.cvbbacm.com/> - CVBB ACM Team

Permanent link: https://wiki.cvbbacm.com/doku.php?id=2020-2021.teams.legal_string:%E7%BB%84%E9%98%9F%E8%AE%AD%E7%BB%83%E6%AF%94%E8%B5%9B%E8%AE%B0%E5%BD%95:%E7%BC%93%E5%86%B2%E5%8C%BA&rev=1629277690

Last update: 2021/08/18 17:08