A

洛谷 P2858
区间 DP，用 dp[l][r]表示 1~l-1 与 r+1~n 均被取完后，剩下的美食所能获得的最大的美味值，可用记忆化搜索来写。

代码如下：

```
#include <iostream>
using namespace std;

int n,a[2100],dp[2100][2100];

int dfs(int l,int r)
{
    if(l>r) return 0;
    if(dp[l][r]) return dp[l][r];
    int d=n-(r-l+1)+1;
    int s1=dfs(l+1,r)+d*a[l];
    int s2=dfs(l,r-1)+d*a[r];
    dp[l][r]=max(s1,s2);
    return dp[l][r];
}

int main()
{
    cin>>n;
    for(int i=1;i<=n;i++) cin>>a[i];
    cout<<dfs(1,n)<<endl;
    return 0;
}
```

B

洛谷 P2015
树形 DP，用 dp[x][i]表示以 x 为根节点的子树中保留 i 条边所能获得的最大边权和，更新时暴力枚举 i 即可。

代码如下：

```
#include <iostream>
#include <cstdio>
using namespace std;

int n,q,f[110][110];
struct node {
```

```
    int x,y,d,next;
}a[210]; int len,last[210];

inline void ins(int x,int y,int d)
{
    ++len; a[len].x=x,a[len].y=y,a[len].d=d;
    a[len].next=last[x],last[x]=len;
}

void dfs(int x,int fa)
{
    for(int k=last[x];k;k=a[k].next) {
        int y=a[k].y;
        if(y==fa) continue;
        dfs(y,x);
        for(int i=q;i>0;i--) //暴力枚举子树中保留的边数
            for(int j=0;j+1<=i;j++) //暴力枚举孩子结点保留的边数
                f[x][i]=max(f[x][i],f[x][i-j-1]+f[y][j]+a[k].d);
    }
}

int main()
{
    scanf("%d%d",&n,&q);
    for(int i=1;i<n;i++) {
        int x,y,c; scanf("%d%d%d",&x,&y,&c);
        ins(x,y,c),ins(y,x,c);
    }
    dfs(1,0);
    printf("%d\n",f[1][q]);
    return 0;
}
```

C

洛谷 P8059

一只猴子用手去钩住另一只猴子，看作二者之间存在一条边相连接，猴子放手看作该边被删除，把被删除的边记录下来，没删的直接相连，所得到的就是最后的情况。再将删除的边按照相反的顺序重新连接，当连接完一条边后，使得某点与 1 号点相连通，那么这就是这只猴子的掉落时间，考虑用并查集完成。

代码如下：

```
#include <iostream>
using namespace std;
```

```
int
n,m,a[210000][3],bk[210000][2],num[410000],h[410000],fa[210000],ans[210000],nx
t[210000]; //nxt 数组表示相连的下一个点，在更新答案时会用到

int gr(int a)
{
    if(fa[a]==0) return a;
    return fa[a]=gr(fa[a]);
}

int dfs(int x)
{
    if(ans[x]!=-2) return ans[x];
    return ans[x]=dfs(nxt[x]);
} //往前得到一整个连通块中猴子的掉落时间

int main()
{
    cin>>n>>m;
    for(int i=1;i<=n;i++) cin>>a[i][1]>>a[i][2];
    for(int i=1;i<=n;i++) bk[i][1]=bk[i][2]=1;
    for(int i=0;i<m;i++) {
        cin>>num[i]>>h[i];
        bk[num[i]][h[i]]=0; //记录被删除的边
    }
    for(int i=1;i<=n;i++) {    //将未被删除的边连接
        if(bk[i][1]&&a[i][1]!=-1) {
            int x=gr(i),y=gr(a[i][1]);
            if(x!=y) {
                if(x==1) swap(x,y);
                fa[x]=y,nxt[x]=y;
            }
        }
        if(bk[i][2]&&a[i][2]!=-1) {
            int x=gr(i),y=gr(a[i][2]);
            if(x!=y) {
                if(x==1) swap(x,y);
                fa[x]=y,nxt[x]=y;
            }
        }
    }
    for(int i=1;i<=n;i++) {
        if(gr(i)==gr(1)) ans[i]=-1; //最后仍然与 1 相连通的点
```

```
            else ans[i]=-2; //其余的点
        }
        for(int i=m-1;i>=0;i--) { //将删除的边按相反的顺序重新连接
            if(a[num[i]][h[i]]==-1) continue;
            int x=gr(num[i]),y=gr(a[num[i]][h[i]]);
            if(x!=y) {
                if(x==1) swap(x,y);
                if(y==1) ans[x]=i; //点 x 重新与 1 相连，得到他的掉落时间
                fa[x]=y,nxt[x]=y;
            }
        }
        for(int i=2;i<=n;i++) {
            ans[i]=dfs(i);
        }
        for(int i=1;i<=n;i++) cout<<ans[i]<<endl;
        return 0;
}
```

D

洛谷 P4281

令 a=LCA(x,y),b=LCA(x,z),c=LCA(y,z),由于两个人一起移动比一个人移动的花费多，故答案一定在 a,b,c 三点之中。通过观察可知，a,b,c 三点中一定至少有两点是相同的，且集合点在这一点的花费不低于在另一点的花费，则只需找到相同的两点，然后以另一点作为目标点即可。

代码如下：

```
#include <iostream>
#include <cstdio>
using namespace std;

int n,m;
struct node {
    int x,y,next;
}a[1110000]; int len,last[510000];

inline void ins(int x,int y)
{
    len++; a[len].x=x,a[len].y=y;
    a[len].next=last[x]; last[x]=len;
}

struct trnode {
    int dep,par[30];
```

```
}tr[1110000];

void bt(int x,int fa)
{
	tr[x].dep=tr[fa].dep+1,tr[x].par[0]=fa;
	for(int i=1;(1<<i)<=tr[x].dep;i++)
		tr[x].par[i]=tr[tr[x].par[i-1]].par[i-1];
	for(int k=last[x];k;k=a[k].next) {
		int y=a[k].y;
		if(y!=fa) bt(y,x);
	}
}

inline int LCA(int x,int y)
{
	if(tr[x].dep<tr[y].dep) swap(x,y);
	for(int i=18;i>=0;i--)
		if(tr[x].dep-tr[y].dep>=(1<<i))
			x=tr[x].par[i];
	if(x==y) return x;
	for(int i=18;i>=0;i--) {
		if(tr[x].par[i]!=tr[y].par[i]&&tr[x].dep>=(1<<i)) {
			x=tr[x].par[i];
			y=tr[y].par[i];
		}
	}
	return tr[x].par[0];
}

int main()
{
	scanf("%d%d",&n,&m);
	for(int i=1;i<n;++i) {
		int x,y; scanf("%d%d",&x,&y);
		ins(x,y); ins(y,x);
	}
	bt(1,0);
	for(int i=1;i<=m;++i) {
		int x,y,z,k; scanf("%d%d%d",&x,&y,&z);
		int a=LCA(x,y),b=LCA(x,z),c=LCA(y,z);
		if(a==b) k=c;
		else if(a==c) k=b;
		else if(b==c) k=a;
		int ans=tr[x].dep+tr[y].dep+tr[z].dep-tr[a].dep-tr[b].dep-tr[c].dep;
```

```
            printf("%d %d\n",k,ans);
        }
        return 0;
}
```

E

Codeforces1843D

用 f[x]表示以 x 为根节点的子树中的叶子结点个数，那么生长在点 x 上的苹果的掉落方案数就是 f[x]，由乘法原理得，生长于 x、y 的两个苹果的掉落方案总数为 f[x]*f[y]。

代码如下：

```
#include <iostream>
#include <vector>
using namespace std;

int t,n,q,f[210000];
vector<int> a[210000];

void dfs(int x,int fa)
{
    for(int i=0;i<a[x].size();i++) {
        int y=a[x][i];
        if(y!=fa) { dfs(y,x); f[x]+=f[y]; }
    }
    if(f[x]==0) f[x]=1;
}

int main()
{
    cin>>t;
    while(t--) {
        cin>>n;
        for(int i=1;i<n;i++) {
            int x,y; cin>>x>>y;
            a[x].push_back(y),a[y].push_back(x);
        }
        for(int i=1;i<=n;i++) f[i]=0;
        dfs(1,0);
        cin>>q;
        while(q--) {
            int x,y; cin>>x>>y;
            cout<<1ll*f[x]*f[y]<<endl;
```

```
        }
        for(int i=1;i<=n;i++) a[i].clear();
    }
    return 0;
}
```

F

洛谷 P2023
乘法线段树模板题。

G

洛谷 P2357
模板题，线段树、树状数组均可。

H

洛谷 P8648
扫描线模板题，或者用二维前缀和也可以。

I

洛谷 P6216
manacher+kmp，对于 s1 中的回文子串，可以用 manacher 算法来求得；对于 s2 在 s1 中出现的位置，可以用 kmp 算法来求，进而用前缀和得出 s2 在 s1 的子串当中出现的次数。对于 manacher 所求的以某个位置为中心的最长回文子串，其内部同时存在多个以该位置为中心的回文子串，通过观察可知，可通过再次求前缀和的方式来求出 s2 在每个回文子串当中出现的总次数。

代码如下：

```
#include <iostream>
#include <cstring>
using namespace std;
#define LL long long

const LL M=(1ll<<32);
int n,m,len[6110000],mx,p,nxt[3110000];
char s1[3110000],s2[3110000],s[6110000];
LL ans,pre[3110000],sum[3110000];

int main()
{
    cin>>n>>m>>(s1+1)>>(s2+1);
```

```
    s[0]='#';
    for(int i=1;i<=n;i++) s[i*2-1]=s1[i],s[i*2]='#';
    for(int i=1;i<=2*n;i++) {
        if(i<mx) len[i]=min(mx-i,len[2*p-i]);
        else len[i]=1;
        while(i-len[i]>=0&&s[i+len[i]]==s[i-len[i]]) ++len[i];
        if(i+len[i]>mx) mx=i+len[i],p=i;
    } //manacher 求每个中心的最长回文子串
    nxt[1]=0;
    int j=0;
    for(int i=2;i<=m;i++) {
        j=nxt[i-1];
        while(j>0&&s2[i]!=s2[j+1]) j=nxt[j];
        if(s2[i]==s2[j+1]) nxt[i]=j+1;
        else nxt[i]=0;
    }
    j=0;
    for(int i=1;i<=n;i++) {
        while(j>0&&s1[i]!=s2[j+1]) j=nxt[j];
        if(s1[i]==s2[j+1]) ++j;
        if(j==m) pre[i-m+1]=1;
    } //kmp 求 s2 在 s1 当中出现的每个位置
    for(int i=1;i<=n;i++) pre[i]+=pre[i-1];
    for(int i=1;i<=n;i++) sum[i]=sum[i-1]+pre[i];
    for(int i=1;i<=n;i++) {
        int Len=len[i*2-1]-1;
        int l=i-Len/2,r=i+Len/2;
        if(Len<m) continue;
        ans=(ans+sum[r-m+1]-sum[i+m/2-m])%M;
        ans=((ans-(sum[i-m/2-1]-(l>=2?sum[l-2]:0)))%M+M)%M;
    }
    cout<<ans<<endl;
    return 0;
}
```