**Routine Library**

By 1em0n Team

March 29, 2019

Part 1 几何

（留坑）

Part 2 数论

2.1 分解质因数

**2.2 同余方程（扩欧）**

2.3 欧拉函数

2.4 简易素数表

2.5 中国剩余定理

2.6 快速傅里叶变换（FFT）

2.7 线性规划

2.8 高斯消元

**2.9 佩兰方程**

Part 3 字符串

**3.1 Trie**

3.2 KMP

3.3 后缀树

3.4 后缀数组

**3.5 后缀自动机**

3.6 回文树

**3.7 马拉车**

Part 4 图论

**4.1 二分图匹配**

4.2 最大团

**4.3 网络流**

**4.4 强联通分量/Tarjan**

4.5 2-sat

**4.6 割点**

4.7 割边

**4.8 dsu**

**4.9 树链剖分**

**4.10 Floyd最小环**

**4.11 最小生成树**

**4.12 点分治**

**4.13 LCA（倍增/ST/树剖）**

Part 5 数据结构

5.1 Splay

**5.2 树状数组**

**5.3 线段树**

**5.4 并查集带权**

**5.5单调栈——求全flag最大子矩阵**

**5.6 ST表**

Part 6 其它

6.1 高精度

**6.2 数位DP**

**6.3 CDQ分治（二维偏序）**

**6.4 三分**

**6.5 矩阵快速幂**

2.2扩欧

ll exgcd(ll a,ll b,ll &x,ll &y)

{

if(b==0)

{

x=1;

y=0;

return a;

}

ll g=exgcd(b,a%b,x,y),t;

t=x;

x=y;

y=t-(a/b)\*y;

return g;

}

2.9佩兰方程

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

ll a[20000];

bool pell\_minimum\_solution(ll n,ll &x0,ll &y0){//求PELL方程最小整数解

ll m=(ll)sqrt((double)n);//M是N的平方根向下取整

if(m\*m==n)return false;//当n是完全平方数则佩尔方程无解（不讨论正负一，零）

//下面是把N用连分数形式存，B,C,TMP（即AI）的递推见解释以22为例参考

int i=0; //连分数的数位

a[i++]=m; //A0位整数部分m=4

ll b=m,c=1; //B=4即整数部分位,C=1即求RN时的分母

double sq=sqrt(n);//SQ是N的高精度根,相当于r0

double tmp;//tp在下面的循环就是rn

do{

c=(n-b\*b)/c;

tmp=(sq+b)/c;

a[i++]=(ll)(floor(tmp));

b=a[i-1]\*c-b;

//printf("%lld %lld %lld\n",a[i-1],b,c);

}while(a[i-1]!=2\*a[0]);//当有一位等于整数两倍就结束

//下面就是要把连分数形式化成分子分母的形式，求PQ两个值

ll p=1,q=0;

for(int j=i-2;j>=0;j--){

ll t=p;

p=q+p\*a[j];

q=t;

//printf("a[%d]=%lld %lld %lld\n",j,a[j],p,q);

}

if((i-1)%2==0){x0=p;y0=q;}//如果I是奇数，X0与Y0都是0

else{x0=2\*p\*p+1;y0=2\*p\*q;}//如果I是偶数，X0是两倍P方+1，y0是两倍PQ

return true;

}

int main(){

ll n,x,y;

while(~scanf("%lld",&n)){

if(pell\_minimum\_solution(n,x,y)){//输入N求X,Y的值，有解就输出

printf("x=%lld y=%lld\t",x,y);//x,y

printf("%lld^2-%lld\*%lld^2=1\t",x,n,y);//x^2-n\*y^2=1

printf("%lld-%lld=1\n",x\*x,n\*y\*y);//(x^2)-(n\*y^2)=1

}

}

return 0;

}

3.1 Trie

int ch[N\*L][27],size=0,val[N\*L];

void insert(char s[],int n,int id){

int u=0;

for(int i=1;i<=n;i++){

int v=s[i]-'a';

if(!ch[u][v]) ch[u][v]=++size;

u=ch[u][v];

}

val[u]=id;//printf("ins %d %d\n",u,id);

}

3.5后缀自动机

#include<iostream>

#include<cstdio>

#include<cstring>

#include<algorithm>

#include<queue>

#include<cmath>

#include<map>

#define fi first

#define se second

#define pii pair<int,int>

#define inf 1000000005

#define pb push\_back

#define maxn 100005

using namespace std;

typedef long long int ll;

const int N=3e6+10;

int now=1,cnt=1,cntedge=0;

ll ans=0;

char s[3000005];

struct Suffix\_Automaton

{

int mp[N][30];

int fa[N];

int p,np,q,nq;

int mx[N],siz[N],v[N],nxt[N],head[N];

//Suffix\_Automaton(){np=nq=1;}

void insert(int x)

{

p=now,np=++cnt;

mx[np]=mx[now]+1;now=np;siz[np]=1;

while(!mp[p][x]&&p)mp[p][x]=np,p=fa[p];

if(!p)fa[np]=1;

else

{

q=mp[p][x];

if(mx[q]==mx[p]+1)fa[np]=q;

else

{

nq=++cnt;

mx[nq]=mx[p]+1;

memcpy(mp[nq],mp[q],sizeof(mp[q]));

fa[nq]=fa[q];fa[q]=fa[np]=nq;

while(mp[p][x]==q)mp[p][x]=nq,p=fa[p];

}

}

}

void addedge(int u,int t)

{

cntedge++;

v[cntedge]=t;

nxt[cntedge]=head[u];

head[u]=cntedge;

}

void build()

{

for(int i=2;i<=nq;i++)addedge(fa[i],i);

}

void dfs(int u)

{

for(int i=head[u];i;i=nxt[i])dfs(v[i]),siz[u]+=siz[v[i]];

if(siz[u]!=1)ans=max(ans,(ll)siz[u]\*mx[u]);

}

}sam;

int main()

{

scanf("%s",s);

for(int i=0;s[i];i++)sam.insert(s[i]-'a'+1);

sam.build();sam.dfs(1);

printf("%lld\n",ans);

return 0;

}

3.7马拉车

int Manacher(){

Len[0] = 0;

int sum = 0;

mx = 0;

for(int i=1;i<len;i++){

if(i < mx) Len[i] = Min(mx - i, Len[2 \* id - i]);

else Len[i] = 1;

while(str[i - Len[i]]== str[i + Len[i]]) Len[i]++;

if(Len[i] + i > mx){

mx = Len[i] + i;

id = i;

sum = Max(sum, Len[i]);

}

}

return (sum - 1);

}

4.1二分图

4.1.1二分图最大匹配

const int N=1005;

inline int read(){

char c=getchar();int x=0,f=1;

while(c<'0'||c>'9'){if(c=='-')f=-1;c=getchar();}

while(c>='0'&&c<='9'){x=x\*10+c-'0';c=getchar();}

return x\*f;

}

int n,m,s,u,v;

struct edge{

int v,ne;

}e[N\*N<<1];

int h[N],cnt=0;

inline void ins(int u,int v){

cnt++;

e[cnt].v=v;e[cnt].ne=h[u];h[u]=cnt;

}

int vis[N],le[N];

bool find(int u){

for(int i=h[u];i;i=e[i].ne){

int v=e[i].v;

if(!vis[v]){

vis[v]=1;

if(!le[v]||find(le[v])){

le[v]=u;

return true;

}

}

}

return false;

}

int ans=0;

void hungary(){

for(int i=1;i<=n;i++){

memset(vis,0,sizeof(vis));

if(find(i)) ans++;

}

}

int main(){

n=read();m=read();int t=read();

for(int i=1;i<=t;i++){u=read();v=read();if(v>m)continue;ins(u,v);}

ans=0;

hungary();

printf("%d\n",ans);

}

4.1.2 二分图染色

bool color(int u,int c){

col[u]=c;

for(int i=h[u];i;i=e[i].ne){

int v=e[i].v;

if(col[u]==col[v]) return false;

if(!col[v]&&!color(v,3-c)) return false;

}

return true;

}

4.3 网络流

4.3.1 Dinic

#define inf 200000000

#define maxm 100005

#define maxn 10005

#define md 1000000007

using namespace std;

typedef long long int ll;

int n,m,s,t;

int head[maxn],cur[maxn];

struct node

{

int to,next,w;

}edge[2\*maxm];

int cnt=0;

void addedge(int x,int y,int z)

{

edge[cnt].to=y;

edge[cnt].w=z;

edge[cnt].next=head[x];

head[x]=cnt;

cnt++;

}

int dep[maxn];

bool bfs()

{

queue<int>q;while(!q.empty())q.pop();

memset(dep,0,sizeof(dep));

dep[s]=1;

q.push(s);

while(!q.empty())

{

int u=q.front();q.pop();

for(int i=head[u];i!=-1;i=edge[i].next)

{

int v=edge[i].to;

if(!dep[v]&&edge[i].w>0)

{

dep[v]=dep[u]+1;

q.push(v);

}

}

}

if(dep[t])return 1;

return 0;

}

int dfs(int u,int dist)

{

if(u==t)return dist;

//int flow=0;

for(int& i=cur[u];i!=-1;i=edge[i].next)

{

int v=edge[i].to;

if(dep[v]==dep[u]+1&&edge[i].w>0)

{

int di=dfs(v,min(dist,edge[i].w));

if(di>0)

{

edge[i].w-=di;

edge[i^1].w+=di;

return di;

}

}

}

return 0;

}

int dinic()

{

int ans=0;

while(bfs())

{

for(int i=1;i<=n;i++)cur[i]=head[i];

while(int d=dfs(s,inf))ans+=d;

}

return ans;

}

int main()

{

memset(head,-1,sizeof(head));

scanf("%d%d%d%d",&n,&m,&s,&t);

for(int i=1;i<=m;i++)

{

int x,y,z;

scanf("%d%d%d",&x,&y,&z);

addedge(x,y,z);

addedge(y,x,0);

}

printf("%d\n",dinic());

return 0;

}

4.3.2费用流

#define inf 200000000

#define maxm 100005

#define maxn 10005

#define md 1000000007

using namespace std;

typedef long long int ll;

int n,m,s,t;

int head[maxn],cur[maxn];

struct node

{

int to,next,w;

}edge[2\*maxm];

int cnt=0;

void addedge(int x,int y,int z)

{

edge[cnt].to=y;

edge[cnt].w=z;

edge[cnt].next=head[x];

head[x]=cnt;

cnt++;

}

int dep[maxn];

bool bfs()

{

queue<int>q;while(!q.empty())q.pop();

memset(dep,0,sizeof(dep));

dep[s]=1;

q.push(s);

while(!q.empty())

{

int u=q.front();q.pop();

for(int i=head[u];i!=-1;i=edge[i].next)

{

int v=edge[i].to;

if(!dep[v]&&edge[i].w>0)

{

dep[v]=dep[u]+1;

q.push(v);

}

}

}

if(dep[t])return 1;

return 0;

}

int dfs(int u,int dist)

{

if(u==t)return dist;

//int flow=0;

for(int& i=cur[u];i!=-1;i=edge[i].next)

{

int v=edge[i].to;

if(dep[v]==dep[u]+1&&edge[i].w>0)

{

int di=dfs(v,min(dist,edge[i].w));

if(di>0)

{

edge[i].w-=di;

edge[i^1].w+=di;

return di;

}

}

}

return 0;

}

int dinic()

{

int ans=0;

while(bfs())

{

for(int i=1;i<=n;i++)cur[i]=head[i];

while(int d=dfs(s,inf))ans+=d;

}

return ans;

}

int main()

{

memset(head,-1,sizeof(head));

scanf("%d%d%d%d",&n,&m,&s,&t);

for(int i=1;i<=m;i++)

{

int x,y,z;

scanf("%d%d%d",&x,&y,&z);

addedge(x,y,z);

addedge(y,x,0);

}

printf("%d\n",dinic());

return 0;

}

4.4 强联通分量/Tarjan

const int N=1e4+5,M=5e4+5;

typedef long long ll;

inline int read(){

char c=getchar();int x=0,f=1;

while(c<'0'||c>'9'){if(c=='-')f=-1;c=getchar();}

while(c>='0'&&c<='9'){x=x\*10+c-'0';c=getchar();}

return x\*f;

}

int n,m,u,v;

struct edge{

int v,ne;

}e[M];

int h[N],cnt=0;

inline void ins(int u,int v){

cnt++;

e[cnt].v=v;e[cnt].ne=h[u];h[u]=cnt;

}

int dfn[N],belong[N],low[N],dfc,scc,st[N],top;

int size[N];

void dfs(int u){

dfn[u]=low[u]=++dfc;

st[++top]=u;

for(int i=h[u];i;i=e[i].ne){

int v=e[i].v;

if(!dfn[v]){

dfs(v);

low[u]=min(low[u],low[v]);

}else if(!belong[v])

low[u]=min(low[u],dfn[v]);

}

if(low[u]==dfn[u]){

scc++;

while(true){

int x=st[top--];

belong[x]=scc;

size[scc]++;

if(x==u) break;

}

}

}

int outd[N],ind[N],ans;

void point(){

for(int u=1;u<=n;u++)

for(int i=h[u];i;i=e[i].ne){

int v=e[i].v;

if(belong[u]!=belong[v]) outd[belong[u]]++,ind[belong[v]]++;

}

}

int main(){

n=read();m=read();

for(int i=1;i<=m;i++){u=read();v=read();ins(u,v);}

for(int i=1;i<=n;i++) if(!dfn[i]) dfs(i);

point();

for(int i=1;i<=scc;i++){

if(outd[i]==0){

if(ans){ans=0;break;}

else ans=size[i];

}

}

printf("%d",ans);

}

4.6割点

const int N=1e5+5,M=1e5+5,INF=1e9+5;

inline int read(){

char c=getchar();int x=0,f=1;

while(c<'0'||c>'9'){if(c=='-')f=-1;c=getchar();}

while(c>='0'&&c<='9'){x=x\*10+c-'0';c=getchar();}

return x;

}

int n=0,m,u,v;

struct edge{

int v,ne;

}e[M<<1];

int h[N],cnt=0;

inline void ins(int u,int v){

cnt++;

e[cnt].v=v;e[cnt].ne=h[u];h[u]=cnt;

cnt++;

e[cnt].v=u;e[cnt].ne=h[v];h[v]=cnt;

}

int dfn[N],low[N],dfc=0,iscut[N];

void dfs(int u,int fa){

dfn[u]=low[u]=++dfc;

int child=0;

for(int i=h[u];i;i=e[i].ne){

int v=e[i].v;

if(!dfn[v]){

child++;

dfs(v,u);

low[u]=min(low[u],low[v]);

if(low[v]>=dfn[u]) iscut[u]=1;

}else if(v!=fa) low[u]=min(low[u],dfn[v]);

}

if(fa==0&&child==1) iscut[u]=0;

}

int main(){

n=read();m=read();

for(int i=1;i<=m;i++){u=read();v=read();ins(u,v);}

for(int i=1;i<=n;i++) if(!dfn[i]) dfs(i,0);

int ans=0;

for(int i=1;i<=n;i++) if(iscut[i]) ans++;

printf("%d\n",ans);

for(int i=1;i<=n;i++) if(iscut[i]) printf("%d ",i);

}

4.8 dsu

#include<iostream>

#include<cstring>

#include<algorithm>

#include<cstdio>

#include<cmath>

#define inf 2000000000

#define maxn 100005

using namespace std;

typedef long long int ll;

int n;

ll ans[maxn],sum=0;

int col[maxn],cnt[maxn],big[maxn],son[maxn];

int cntt=0,head[maxn],siz[maxn],sm=0;

struct node

{

int to,next;

}edge[2\*maxn];

void addedge(int x,int y)

{

cntt++;

edge[cntt].to=y;

edge[cntt].next=head[x];

head[x]=cntt;

}

void dfs(int now,int fa)

{

siz[now]=1;

for(int i=head[now];i;i=edge[i].next)

{

int v=edge[i].to;

if(v==fa)continue;

dfs(v,now);

siz[now]+=siz[v];

if(siz[v]>siz[son[now]])son[now]=v;

}

}

void add(int now,int fa,int val)

{

cnt[col[now]]+=val;

if(val>0&&cnt[col[now]]>=sm)

{

if(cnt[col[now]]>sm)sum=0,sm=cnt[col[now]];

sum+=col[now];

}

for(int i=head[now];i;i=edge[i].next)

{

int v=edge[i].to;

if(v!=fa&&!big[v])add(v,now,val);

}

}

void work(int now,int fa,int keep)

{

for(int i=head[now];i;i=edge[i].next)

{

int v=edge[i].to;

if(v==fa)continue;

if(v!=son[now])work(v,now,0);

}

if(son[now])work(son[now],now,1),big[son[now]]=1;

add(now,fa,1);

ans[now]=sum;

if(son[now])big[son[now]]=0;

if(keep==0)add(now,fa,-1),sm=sum=0;

}

int main()

{

scanf("%d",&n);

for(int i=1;i<=n;i++)scanf("%d",&col[i]);

for(int i=1;i<n;i++)

{

int x,y;

scanf("%d%d",&x,&y);

addedge(x,y);

addedge(y,x);

}

dfs(1,0);work(1,0,0);

for(int i=1;i<=n;i++)printf("%I64d ",ans[i]);

printf("\n");

return 0;

}

4.9树链剖分

#include<iostream>

#include<cstdio>

#include<cstdlib>

#include<cstring>

#include<cmath>

#include<algorithm>

#define LL long long

#define l(x) (x<<1)

#define r(x) ((x<<1)|1)

using namespace std;

struct EDG

{

int to,nex;

}es[1000010];

int n,m,r,p;

int first[100010];

int cnt;

void link(int a,int b)

{

es[++cnt].nex=first[a];

first[a]=cnt;

es[cnt].to=b;

}

LL a[1000100];

//树剖↓

LL real[1000010],sid[1000010],hson[1000010],fa[1000010],dep[1000010],size[1000010],top[1000010];

LL tot;

void dfs1(int x)

{

size[x]=1;

for(int i=first[x];i;i=es[i].nex)

{

int v=es[i].to;

if(v!=fa[x])

{

fa[v]=x;

dep[v]=dep[x]+1;

dfs1(v);

if(hson[x]==0||size[hson[x]]<size[v])

{

hson[x]=v;

}

size[x]+=size[v];

}

}

}

void dfs2(int x,int anc)

{

top[x]=anc;

sid[x]=++tot;

real[tot]=x;

if(hson[x]==0) return;

dfs2(hson[x],anc);

for(int i=first[x];i;i=es[i].nex)

{

int v=es[i].to;

if(v!=hson[x]&&v!=fa[x])

{

dfs2(v,v);

}

}

}

//树剖↑

//线段树 ↓

struct Tre

{

LL sum,tag;

}Tr[1000100];

void update(LL id)

{

Tr[id].sum=Tr[l(id)].sum+Tr[r(id)].sum;

}

void pushdown(LL l,LL r,LL id)

{

Tr[r(id)].tag+=Tr[id].tag;

Tr[l(id)].tag+=Tr[id].tag;

Tr[id].sum+=(r-l+1)\*Tr[id].tag;

Tr[id].tag=0;

}

void build(LL l,LL r,LL id)

{

if(l==r)

{

Tr[id].sum=a[real[l]];

return;

}

LL mid=(l+r)/2;

build(mid+1,r,r(id));

build(l,mid,l(id));

update(id);

}

void add(LL al,LL ar,LL x,LL l,LL r,LL id)

{

if(l>ar||r<al) return;

if(al==l&&ar==r) {Tr[id].tag+=x;return;}

pushdown(l,r,id);

LL mid=(r+l)/2;

if(mid>=al) add(al,min(mid,ar),x,l,mid,l(id));

if(mid<ar) add(max(mid+1,al),ar,x,mid+1,r,r(id));

pushdown(l,mid,l(id));

pushdown(mid+1,r,r(id));

update(id);

}

LL query(LL al,LL ar,LL l,LL r,LL id)

{

if(l>r) return 0LL;

if(al==l&&ar==r) {return Tr[id].tag\*(r-l+1)+Tr[id].sum;}

pushdown(l,r,id);

LL mid=(r+l)/2;

LL t=0;

if(mid>=al) t+=query(al,min(mid,ar),l,mid,l(id));

if(mid<ar) t+=query(max(mid+1,al),ar,mid+1,r,r(id));

return t;

}

//线段树 ↑

//树剖用线段树维护↓

void tr\_add(LL x,LL v)//x点及其子孙权值加v

{

add(sid[x],sid[x]+size[x]-1,v,1,n,1);

}

LL tr\_query(LL x)//x点及其子孙权值和

{

return query(sid[x],sid[x]+size[x]-1,1,n,1)%p;

}

LL chain\_add(LL x,LL y,LL v)//x到y的链上每个点加v

{

LL tx,ty;

tx=top[x];ty=top[y];

while(tx!=ty)

{

if(dep[tx]<dep[ty])

{

swap(x,y);

swap(tx,ty);

}

add(sid[tx],sid[x],v,1,n,1);

x=fa[tx];tx=top[x];

}

if(dep[x]<dep[y]) swap(x,y);

add(sid[y],sid[x],v,1,n,1);

}

LL chain\_query(LL x,LL y)//查询x到y的链上权值和

{

LL tx,ty,t=0;

tx=top[x];ty=top[y];

while(tx!=ty)

{

if(dep[tx]<dep[ty]) {swap(tx,ty);swap(x,y);}

(t+=query(sid[tx],sid[x],1,n,1))%=p;

x=fa[tx];tx=top[x];

}

if(dep[x]<dep[y]) swap(x,y);

(t+=query(sid[y],sid[x],1,n,1))%=p;

return t;

}

//树剖用线段树维护↑

int main()

{

scanf("%d%d%d%d",&n,&m,&r,&p);

for(int i=1;i<=n;i++) scanf("%lld",&a[i]);

for(int i=1;i<n;i++)

{

int x,y;

scanf("%d%d",&x,&y);

link(x,y);link(y,x);

}

dep[r]=1;

dfs1(r);

dfs2(r,r);

build(1,n,1);

for(int i=1;i<=m;i++)

{

int q;

scanf("%d",&q);

if(q==1)

{

LL x,y,z;

scanf("%lld%lld%lld",&x,&y,&z);

chain\_add(x,y,z);

}

if(q==2)

{

LL x,y;

scanf("%lld%lld",&x,&y);

cout<<chain\_query(x,y);

cout<<endl;

}

if(q==3)

{

LL x,z;

scanf("%lld%lld",&x,&z);

tr\_add(x,z);

}

if(q==4)

{

LL x;

scanf("%lld",&x);

cout<<tr\_query(x);

cout<<endl;

}

}

}

4.10 Floyd最小环

const int N=105,M=1e4+5,INF=1e8+5;//1E9+1E9+1E9溢出

inline int read(){

char c=getchar();int x=0,f=1;

while(c<'0'||c>'9'){if(c=='-')f=-1;c=getchar();}

while(c>='0'&&c<='9'){x=x\*10+c-'0';c=getchar();}

return x\*f;

}

int n,m,u,v,w,g[N][N];

int d[N][N],ans=INF;

void floyd(){

ans=INF;

for(int k=1;k<=n;k++){

for(int i=1;i<=k-1;i++)

for(int j=i+1;j<=k-1;j++)

ans=min(ans,g[i][k]+g[k][j]+d[i][j]);

for(int i=1;i<=n;i++)

for(int j=1;j<=n;j++)

d[i][j]=min(d[i][j],d[i][k]+d[k][j]);

}

}

int main(){

while(scanf("%d%d",&n,&m)!=EOF){

for(int i=1;i<=n;i++) for(int j=i+1;j<=n;j++) d[i][j]=d[j][i]=g[i][j]=g[j][i]=INF;

for(int i=1;i<=m;i++){

u=read();v=read();w=read();

d[u][v]=d[v][u]=g[u][v]=g[v][u]=w;

}

floyd();

if(ans==INF) puts("No solution.");

else printf("%d\n",ans);

}

}

4.11 最小生成树

const int N=5005,M=2e5+5,INF=1e9+5;

inline int read(){

char c=getchar();int x=0,f=1;

while(c<'0'||c>'9'){if(c=='-')f=-1;c=getchar();}

while(c>='0'&&c<='9'){x=x\*10+c-'0';c=getchar();}

return x\*f;

}

int n,m,u,v,w;

int cnt=0;

struct edge{

int u,v,w;

bool operator <(const edge &r)const{return w<r.w;}

}e[M];

int fa[N];

inline int find(int x){return x==fa[x]?x:fa[x]=find(fa[x]);}

int kruskal(){

int ans=0,cnt=0;

for(int i=1;i<=n;i++) fa[i]=i;

sort(e+1,e+1+m);

for(int i=1;i<=m;i++){

int u=e[i].u,v=e[i].v,w=e[i].w;

int f1=find(u),f2=find(v);

if(f1!=f2){

ans+=w;

fa[f1]=f2;

cnt++;

if(cnt==n-1) break;

}

}

return ans;

}

int main(){

n=read();m=read();

for(int i=1;i<=m;i++){

e[i].u=read();e[i].v=read();e[i].w=read();

}

int ans=kruskal();

printf("%d",ans);

}

4.12 点分治

#include<iostream>

#include<cstring>

#include<cstdio>

#include<algorithm>

#include<cmath>

#include<queue>

#include<vector>

#define inf 200000000

#define maxm 100005

#define maxn 10005

#define md 1000000007

using namespace std;

typedef long long int ll;

typedef double db;

int n,m,k;

int cnt=0,head[maxn],sum=0,root=0;

struct node{

int to,next,w;

}edge[maxn\*2];

int siz[maxn],dep[maxn],vis[maxn],ans[10000005],f[maxn],o[maxn];

void addedge(int x,int y,int z)

{

cnt++;

edge[cnt].to=y;

edge[cnt].w=z;

edge[cnt].next=head[x];

head[x]=cnt;

}

void getroot(int u,int fa)

{

siz[u]=1;f[u]=0;

for(int i=head[u];i;i=edge[i].next)

{

int v=edge[i].to;

if(v==fa||vis[v])continue;

getroot(v,u);

siz[u]+=siz[v];

f[u]=max(f[u],siz[v]);

}

f[u]=max(f[u],sum-siz[u]);

if(f[u]<f[root])root=u;

}

void getdeep(int u,int fa)

{

o[++cnt]=dep[u];

for(int i=head[u];i;i=edge[i].next)

{

int v=edge[i].to;

if(v==fa||vis[v])continue;

dep[v]=dep[u]+edge[i].w;

getdeep(v,u);

}

}

void calc(int u,int d,int add)

{

cnt=0;dep[u]=d;

getdeep(u,0);

for(int i=1;i<=cnt;i++)

for(int j=1;j<=cnt;j++)

ans[o[i]+o[j]]+=add;

}

void solve(int u)

{

calc(u,0,1);vis[u]=1;

for(int i=head[u];i;i=edge[i].next)

{

int v=edge[i].to;

if(vis[v])continue;

calc(v,edge[i].w,-1);

sum=siz[v];root=0;

getroot(v,0);

solve(root);

}

}

int main()

{

scanf("%d%d",&n,&m);

for(int i=1;i<n;i++)

{

int x,y,z;

scanf("%d%d%d",&x,&y,&z);

addedge(x,y,z);addedge(y,x,z);

}

sum=f[0]=n;

getroot(1,0);

solve(root);

for(int i=1;i<=m;i++)

{

scanf("%d",&k);

if(ans[k])puts("AYE");else puts("NAY");

}

return 0;

}

4.13 LCA

4.13.1倍增版

#include<iostream>

#include<cstring>

#include<algorithm>

#include<cstdio>

using namespace std;

int n,m,s;

int x,y;

int fa[22][500005],d[500005];

bool vis[500005];

struct node

{

int from,to,next;

}edge[1000005];

int cnt=0,head[500005];

void addedge(int from,int to){

cnt++;

edge[cnt].from=from;

edge[cnt].to=to;

edge[cnt].next=head[from];

head[from]=cnt;

}

void dfs(int now,int dep)

{

vis[now]=1;

d[now]=dep;

for(int i=head[now];i;i=edge[i].next)

{

int v=edge[i].to;

if(vis[v])continue;

fa[0][v]=now;

dfs(v,dep+1);

}

}

int lca(int l,int r)

{

if(d[l]<d[r])swap(l,r);

for(int i=21;i>=0;i--)

{

if((d[l]-d[r])>>i&1)

{

l=fa[i][l];

}

}

if(l==r)return l;

for(int i=21;i>=0;i--)

{

if(fa[i][l]!=fa[i][r])

{

l=fa[i][l];

r=fa[i][r];

}

}

return fa[0][l];

}

int main()

{

scanf("%d%d%d",&n,&m,&s);

for(int i=1;i<n;i++)

{

scanf("%d%d",&x,&y);

addedge(x,y);addedge(y,x);

}

dfs(s,0);

for(int i=1;i<=21;i++)

for(int j=1;j<=n;j++)

{

fa[i][j]=fa[i-1][fa[i-1][j]];

}

for(int i=1;i<=m;i++)

{

scanf("%d%d",&x,&y);

printf("%d\n",lca(x,y));

}

return 0;

}

4.13.2 ST表版本

#include<iostream>

#include<cstdio>

#include<cstring>

#include<cmath>

using namespace std;

int n,m,t;

const int N=500005,M=20;

int dp[2\*N][M];

bool vis[N];

struct edge

{

int u,v,w,next;

}e[2\*N];

int tot,head[N];

void add(int u,int v,int w,int &k)

{

e[k].u=u;e[k].v=v;e[k].w=w;

e[k].next=head[u];head[u]=k++;

u=u^v;v=u^v;u=u^v;

e[k].u=u;e[k].v=v;e[k].w=w;

e[k].next=head[u];head[u]=k++;

}

int ver[2\*N],d[2\*N],first[N],dir[N];

//ver:节点编号 d：深度 first：点编号位置 dir：距离

void dfs(int u,int dep)

{

vis[u]=1;ver[++tot]=u;

first[u]=tot;d[tot]=dep;

for(int k=head[u];k!=-1;k=e[k].next)

if(!vis[e[k].v])

{

int v=e[k].v,w=e[k].w;

dir[v]=dir[u]+w;

dfs(v,dep+1);

ver[++tot]=u;d[tot]=dep;

}

}

void st(int n)

{

for(int i=1;i<=n;i++)dp[i][0]=i;

for(int j=1;(1<<j)<=n;j++)

{

for(int i=1;i+(1<<j)-1<=n;i++)

{

int a=dp[i][j-1],b=dp[i+(1<<(j-1))][j-1];//保存的是编号

dp[i][j]=d[a]<d[b]?a:b;

}

}

}

int rmq(int l,int r)

{

int k=0;

while((1<<(k+1))<=r-l+1)k++;

int a=dp[l][k],b=dp[r-(1<<k)+1][k];

return d[a]<d[b]?a:b;

}

int lca(int u,int v)

{

int x=first[u],y=first[v];

if(x>y)swap(x,y);

int res=rmq(x,y);

return ver[res];

}

int main()

{

int q,num=0;

scanf("%d%d%d",&n,&q,&t);

memset(head,-1,sizeof(head));

memset(vis,0,sizeof(vis));

for(int i=1;i<n;i++)

{

int u,v,w;

scanf("%d%d",&u,&v);

add(u,v,1,num);

}

tot=0;dir[1]=0;

dfs(t,1);

st(2\*n-1);

while(q--)

{

int u,v;

scanf("%d%d",&u,&v);

int ans=lca(u,v);

printf("%d\n",ans);

}

return 0;

}

4.13.3 树剖版本

#include<iostream>

#include<cstdio>

#include<cstring>

#include<queue>

#include<set>

#include<map>

#include<vector>

#include<stack>

#include<cmath>

#include<string>

#include<algorithm>

#include<iomanip>

#define md 1000000007

#define inf\_m 1000005

#define inf\_n 500005

#define eps 1e-6

typedef long long int ll;

using namespace std;

int n,q,s,cnt=0,cnt2=0;

//cnt建图用 cnt2 dfs用

struct node

{

int to,nxt;

}edge[inf\_m];

int head[inf\_n];

int siz[inf\_n],son[inf\_n],dep[inf\_n],fa[inf\_n],top[inf\_n],id[inf\_n],rnk[inf\_n];

/\*

siz:子树大小 son:重儿子编号 dep:深度 fa:父亲 top:所在重链顶端节点编号

id:保存每个节点剖分后的新编号(id->dfsid) rnk:dfs序(dfsid->id)

\*/

void addedge(int x,int y)

{

cnt++;

edge[cnt].to=y;

edge[cnt].nxt=head[x];

head[x]=cnt;

}

void dfs1(int t,int f)

{

fa[t]=f;

siz[t]=1;

dep[t]=dep[f]+1;

for(int i=head[t];i;i=edge[i].nxt)

{

int v=edge[i].to;

if(v!=f)

{

dfs1(v,t);

siz[t]+=siz[v];

if(son[t]==0||siz[v]>siz[son[t]])

son[t]=v;

}

}

}

void dfs2(int t,int f)

{

top[t]=f;

id[t]=cnt2;

rnk[cnt2]=t;

cnt2++;

if(!son[t])return;

dfs2(son[t],f);

for(int i=head[t];i;i=edge[i].nxt)

{

int v=edge[i].to;

if(v!=son[t]&&v!=fa[t])

{

dfs2(v,v);

}

}

}

int query(int x,int y)

{

int fx=top[x],fy=top[y];

while(fx!=fy)

{

if(dep[fx]>dep[fy])

{

x=fa[fx];

}

else y=fa[fy];

fx=top[x];fy=top[y];

}

if(dep[x]>dep[y])return y;else return x;

}

int main()

{

scanf("%d%d%d",&n,&q,&s);

for(int i=1;i<n;i++)

{

int x,y;

scanf("%d%d",&x,&y);

addedge(x,y);

addedge(y,x);

}

dfs1(s,0);

dfs2(s,s);

for(int i=1;i<=q;i++)

{

int x,y;

scanf("%d%d",&x,&y);

printf("%d\n",query(x,y));

}

return 0;

}

5.2 树状数组

void update(int x,int y,int n){

for(int i=x;i<=n;i+=lowbit(i))

c[i] += y;

}

int getsum(int x){

int ans = 0;

for(int i=x;i;i-=lowbit(i))

ans += c[i];

return ans;

}

5.3线段树（区间修改）

#define maxn 100005

typedef long long int ll;

int n,m;

struct node

{

int l,r;ll sum,add;

}tree[maxn\*5];

int a[maxn];

void build(int pos,int l,int r)

{

tree[pos].l=l;tree[pos].r=r;

if(l==r)

{

tree[pos].sum=a[l];

return;

}

int mid=(l+r)>>1;

build(pos\*2,l,mid);

build(pos\*2+1,mid+1,r);

tree[pos].sum=tree[pos\*2].sum+tree[pos\*2+1].sum;

}

void spread(int t)

{

if(tree[t].add)

{

tree[t\*2].sum+=tree[t].add\*(ll)(tree[t\*2].r-tree[t\*2].l+1);

tree[t\*2+1].sum+=tree[t].add\*(ll)(tree[t\*2+1].r-tree[t\*2+1].l+1);

tree[t\*2].add+=tree[t].add;

tree[t\*2+1].add+=tree[t].add;

tree[t].add=0;

}

}

void add(int t,int l,int r,ll x)

{

if(l<=tree[t].l&&r>=tree[t].r)

{

tree[t].sum+=(ll)(tree[t].r-tree[t].l+1)\*x;

tree[t].add+=x;

return;

}

spread(t);

int mid=(tree[t].l+tree[t].r)>>1;

if(l<=mid)add(t\*2,l,r,x);

if(r>mid)add(t\*2+1,l,r,x);

tree[t].sum=tree[t\*2].sum+tree[t\*2+1].sum;

}

ll query(int pos,int l,int r)

{

if(tree[pos].l>=l&&tree[pos].r<=r)

return tree[pos].sum;

spread(pos);

int mid=(tree[pos].l+tree[pos].r)>>1;

ll ans=0;

if(l<=mid)ans+=query(pos\*2,l,r);

if(r>mid)ans+=query(pos\*2+1,l,r);

return ans;

}

int main()

{

scanf("%d%d",&n,&m);

for(int i=1;i<=n;i++)

{

scanf("%d",&a[i]);

}

build(1,1,n);

for(int i=1;i<=m;i++)

{

int x,y,z,k;

scanf("%d",&z);

if(z==1)

{

scanf("%d%d%d",&x,&y,&k);

add(1,x,y,k);

}

else

{

scanf("%d%d",&x,&y);

printf("%lld\n",query(1,x,y));

}

}

return 0;

}

5.4并查集带权

for(int i=1;i<=n;i++) fa[i]=i,d[i]=0,s[i]=1;

int fa[N],d[N],s[N];

inline int find(int x){

if(x==fa[x]) return x;

int root=find(fa[x]);

d[x]+=d[fa[x]];

return fa[x]=root;

}

5.5 单调栈 求全flag最大子矩阵

void sol(int flag){

memset(tot,0,sizeof(tot));

for(int i=1;i<=n;i++){

top=0;

for(int j=1;j<=m;j++){

if(a[i][j]==flag) tot[j]++;

else tot[j]=0;

data t;

t.h=tot[j];t.l=1;t.pos=j;

while(top&&st[top].h>=t.h){

int l=st[top].l+j-1-st[top].pos,h=st[top].h;

ans1=max(ans1,min(l,h)\*min(l,h));

ans2=max(ans2,l\*h);

t.l+=st[top].l;

top--;

}

st[++top]=t;

}

while(top){

int l=st[top].l+m-st[top].pos,h=st[top].h;

ans1=max(ans1,min(l,h)\*min(l,h));

ans2=max(ans2,l\*h);

top--;

}

}

}

5.6 ST表

int a[N],f[N][21];

void init(int n){

for(int i=1;i<=n;i++) f[i][0]=a[i];

for(int j=1;j<=20;j++)

for(int i=1;i+(1<<j)-1<=n;i++)

f[i][j]=min(f[i][j-1],f[i+(1<<(j-1))][j-1]);

}

int RMQ(int l,int r){

int k=log(r-l+1)/log(2); //2^k<=l~r

return min(f[l][k],f[r-(1<<k)+1][k]);

}

6.2 数位DP

#include<iostream>

#include<cstring>

#include<cstdio>

#include<algorithm>

using namespace std;

int ansa[10],ansb[10];

void count (int now,int ans[],int times=1)

{

if(now<=0)return;

int p=now/10,d=now%10;

for(int i=1;i<=d;i++)ans[i]+=times;

while(p>0)

{

ans[p%10]+=(d+1)\*times;

p/=10;

}

for(int i=0;i<=9;i++)ans[i]+=times\*(now/10);

times\*=10;

count(now/10-1,ans,times);

return;

}

int main()

{

int a,b;

while(1)

{

scanf("%d%d",&a,&b);

if(a==0&&b==0)break;

memset(ansa,0,sizeof(ansa));

memset(ansb,0,sizeof(ansb));

if(a>b)

{

int temp=a;a=b;b=temp;

}

a--;

if(b>a)

{

count(b,ansb);

count(a,ansa);

}

for(int i=0;i<=9;i++)

printf("%d%c",ansb[i]-ansa[i],i==9?'\n':' ');

}

return 0;

}

6.3 CDQ分治（二维偏序）

#include<cstdio>

#include<cstring>

#include<iostream>

#define ll long long

using namespace std;

const int N=5000010;

int n,m,totx=0,tot=0; //totx是操作的个数,tot询问的编号

struct node{

int type,id;

ll val;

bool operator < (const node &a) const //重载运算符,优先时间排序

{

if (id!=a.id) return id<a.id;

else return type<a.type;

}

};

node A[N],B[N];

ll ans[N];

void CDQ(int L,int R)

{

if (L==R) return;

int M=(L+R)>>1;

CDQ(L,M);

CDQ(M+1,R);

int t1=L,t2=M+1;

ll sum=0;

for (int i=L;i<=R;i++)

{

if ((t1<=M&&A[t1]<A[t2])||t2>R) //只修改左边区间内的修改值

{

if (A[t1].type==1) sum+=A[t1].val; //sum是修改的总值

B[i]=A[t1++];

}

else //只统计右边区间内的查询结果

{

if (A[t2].type==3) ans[A[t2].val]+=sum;

else if (A[t2].type==2) ans[A[t2].val]-=sum;

B[i]=A[t2++];

}

}

for (int i=L;i<=R;i++) A[i]=B[i];

}

int main()

{

scanf("%d%d",&n,&m);

for (int i=1;i<=n;i++)

{

tot++;

A[tot].type=1; A[tot].id=i; //修改操作

scanf("%lld",&A[tot].val);

}

for (int i=1;i<=m;i++)

{

int t;

scanf("%d",&t);

tot++;

A[tot].type=t;

if (t==1)

scanf("%d%lld",&A[tot].id,&A[tot].val);

else

{

int l,r;

scanf("%d%d",&l,&r);

totx++;

A[tot].val=totx; A[tot].id=l-1; //询问的前一个位置

tot++; A[tot].type=3; A[tot].val=totx; A[tot].id=r; //询问的后端点

}

}

CDQ(1,tot);

for (int i=1;i<=totx;i++) printf("%lld\n",ans[i]);

return 0;

}

6.4三分

#include<iostream>

#include<cstring>

#include<cstdio>

#include<algorithm>

#include<cmath>

#include<queue>

#define inf 200000000

#define maxm 10005

#define maxn 10005

using namespace std;

typedef long long int ll;

typedef double db;

const db eps=1e-7;

int n;

db l,r,a[15];

double f(double x)

{

double sum=0;

for(int i=0;i<=n;i++)

sum+=(db)pow(x,(db)i)\*a[i];

return sum;

}

int main()

{

scanf("%d%lf%lf",&n,&l,&r);

for(int i=n;i>=0;i--)scanf("%lf",&a[i]);

while(r-l>eps)

{

double mid=(l+r)/2,midd=(mid+r)/2;

if(f(mid)<f(midd))

l=mid;

else r=midd;

}

printf("%.5lf\n",f(l)>f(r)?l:r);

}

6.5矩阵快速幂

#include<iostream>

#include<cstring>

#include<cstdio>

#include<algorithm>

#include<cmath>

#include<queue>

#define inf 200000000

#define maxm 10005

#define maxn 205

#define md 1000000007

using namespace std;

typedef long long int ll;

struct mat

{

int n,m;

ll a[15][15];

};

mat mul(mat x,mat y)

{

mat ans;

memset(ans.a,0,sizeof(ans.a));

ans.n=x.n;ans.m=y.m;

for(int i=1;i<=x.n;i++)

for(int j=1;j<=y.m;j++)

for(int k=1;k<=x.m;k++)

ans.a[i][j]=(ans.a[i][j]+x.a[i][k]\*y.a[k][j])%md;

return ans;

}

mat matpow(mat c,int t)

{

if(t<=1)return c;

mat ans=c;t--;

while(t>0)

{

if(t%2==1)

{

ans=mul(ans,c);

}

t/=2;

c=mul(c,c);

}

return ans;

}

int main()

{

int T;

scanf("%d",&T);

mat c,d;

c.n=c.m=3;

c.a[1][1]=1;c.a[1][2]=1;c.a[1][3]=0;

c.a[2][1]=0;c.a[2][2]=0;c.a[2][3]=1;

c.a[3][1]=1;c.a[3][2]=0;c.a[3][3]=0;

d.n=1;d.m=3;

d.a[1][1]=d.a[1][2]=d.a[1][3]=1;

while(T--)

{

int n;

scanf("%d",&n);

if(n<=3)

{

printf("1\n");continue;

}

mat t=matpow(c,n-1);

//t=mul(d,t);

printf("%d\n",t.a[1][1]);

}

return 0;

}