ICPC Templates For Grooming

WAKing @ NWPU

October 20, 2020

Contents

1 数据结构

1.1 线段树

```
struct SegmentTree// 区间和,区间加
{
    ll data[maxn<<2],lz[maxn<<2];
    int L[maxn<<2],R[maxn<<2];
    void pushup(int k) {
        data[k]=data[k<<1]+data[k<<1|1];
    }
    void build(int l,int r,int k) {
        L[k]=l;
        R[k]=r;
        data[k]=lz[k]=0;
        if(l==r) {
            return ;
        }
        int mid=(l+r)>>1;
        build(l,mid,k<<1);
        build(mid+1,r,k<<1|1);
        pushup(k);
    }
    void lzadd(int k,ll x) {
        lz[k]+=x;
        data[k]+=(R[k]-L[k]+1)*x;
    }
    void pushdown(int k) {
        if(lz[k]==0) return ;
        lzadd(k<<1,lz[k]);
        lzadd(k<<1|1,lz[k]);
        lz[k]=0;
    }
    void change(int pos,int k,ll x) {
        if(L[k]==R[k]) {
            data[k]=x;
            return ;
        }
        pushup(k);
        int mid=(L[k]+R[k])>>1;
        if(mid>=pos) change(pos,k<<1,x);
        else change(pos,k<<1|1,x);
        pushup(k);
    }
    void add(int l,int r,int k,ll x) {
        if(L[k]>=l&&R[k]<=r) {
            lzadd(k,x);
            return ;
        }
        pushdown(k);
        int mid=(L[k]+R[k])>>1;
        if(mid>=l) add(l,r,k<<1,x);
        if(mid<r) add(l,r,k<<1|1,x);
```

```
        pushup(k);
    }
    ll query(int l,int r,int k) {
        if(L[k]>=l&&R[k]<=r) return data[k];
        pushdown(k);
        ll ans=0;
        int mid=(L[k]+R[k])>>1;
        if(mid>=l) ans=ans+query(l,r,k<<1);
        if(mid<r) ans=ans+query(l,r,k<<1|1);
        return ans;
    }
}tree;
```

1.2 线段树合并

```
struct SegmentTree {
    int cnt=0,L[maxn << 5], R[maxn << 5], data[maxn<<5];// 空间巨大,merge函数可以优化
    void pushup(int x) {
        data[x]=data[L[x]]+data[R[x]];
    }
    void ins(int &now,int l,int r,int pos) {
        if(!now) now=++cnt;
        if(l==r) {
            data[now]++;
            return ;
        }
        int mid=(l+r)>>1;
        if(pos<=mid) ins(L[now],l,mid,pos);
        else ins(R[now],mid+1,r,pos);
        pushup(now);
    }
    int merge(int u,int v,int l,int r) {// 保留u,v 构造一颗新的树
        if(!u||!v) return u|v;
        int mid=(l+r)>>1,now=++cnt;
        if(l==r) data[now]=data[u]+data[v];
        else {
            L[now]=merge(L[u],L[v],l,mid);
            R[now]=merge(R[u],R[v],mid+1,r);
            pushup(now);
        }
        return now;
    }
    int merge(int u,int v,int l,int r) {// 将u合并到v, v不能再被使用.节约空间
        if(!u||!v) return u|v;
        int mid=(l+r)>>1;
        if(l==r) {
            data[u]+=data[v];
        } else {
            L[u]=merge(L[u],L[v],l,mid);
            R[u]=merge(R[u],R[v],mid+1,r);
            pushup(u);
        }
```

```
38        return u;
39    }
40    int query(int now,int l,int r,int ql,int qr) { // 区间求和
41        if(ql<=l&&qr>=r) return data[now];
42        int mid=(l+r)>>1,ans=0;
43        if(mid>=ql) ans+=query(L[now],l,mid,ql,qr);
44        if(mid<qr) ans+=query(R[now],mid+1,r,ql,qr);
45        return ans;
46    }
47 };
```

1.3 主席树

```
1  struct ZXTree {
2     int cnt=0, T[maxn], L[maxn << 4], R[maxn << 4], sum[maxn << 4];
3     void init() { cnt=0; }
4     int build(int l, int r) {
5        int rt = ++cnt;
6        sum[rt] = 0;
7        if (l < r) {
8           int mid = (l + r) >> 1;
9           L[rt] = build(l, mid);
10          R[rt] = build(mid + 1, r);
11       }
12       return rt;
13    }
14    int update(int pre, int l, int r, int x) {
15       int rt = ++cnt;
16       L[rt] = L[pre];
17       R[rt] = R[pre];
18       sum[rt] = sum[pre] + 1;
19       if (l < r)
20       {
21          int mid = (l + r) >> 1;
22          if (x <= mid)
23             L[rt] = update(L[pre], l, mid, x);
24          else
25             R[rt] = update(R[pre], mid + 1, r, x);
26       }
27       return rt;
28    }
29    int query_kth(int u, int v, int l, int r, int k) {//第k小
30       if (l >= r)
31          return l;
32       int x = sum[L[v]] - sum[L[u]], mid = (l + r) >> 1;
33       if (x >= k)
34          return query(L[u], L[v], l, mid, k);
35       else
36          return query(R[u], R[v], mid + 1, r, k - x);
37    }
38    int query_cnt(int u,int v,int l,int r,int x) {//小于k的数目
39       if(r<=x) return sum[v]-sum[u];
```

```
40        int ans=0,mid=(l+r)>>1;
41        ans+=query_cnt(L[u],L[v],l,mid,x);
42        if(x>mid) ans+=query_cnt(R[u],R[v],mid+1,r,x);
43        return ans;
44    }
45 };
```

1.4 splay

```
1  struct Splay {
2     int rt, tot, fa[N], ch[N][2], val[N], cnt[N], sz[N];
3   void maintain(int x) { sz[x] = sz[ch[x][0]] + sz[ch[x][1]] + cnt[x]; }
4   bool get(int x) { return x == ch[fa[x]][1]; }
5   void clear(int x) {
6     ch[x][0] = ch[x][1] = fa[x] = val[x] = sz[x] = cnt[x] = 0;
7   }
8   void rotate(int x) {
9     int y = fa[x], z = fa[y], chk = get(x);
10    ch[y][chk] = ch[x][chk ^ 1];
11    fa[ch[x][chk ^ 1]] = y;
12    ch[x][chk ^ 1] = y;
13    fa[y] = x;
14    fa[x] = z;
15    if (z) ch[z][y == ch[z][1]] = x;
16    maintain(x);
17    maintain(y);
18  }
19  void splay(int x) {
20    for (int f = fa[x]; f = fa[x], f; rotate(x))
21      if (fa[f]) rotate(get(x) == get(f) ? f : x);
22    rt = x;
23  }
24  void ins(int k) {
25    if (!rt) {
26      val[++tot] = k;
27      cnt[tot]++;
28      rt = tot;
29      maintain(rt);
30      return;
31    }
32    int cnr = rt, f = 0;
33    while (1) {
34      if (val[cnr] == k) {
35        cnt[cnr]++;
36        maintain(cnr);
37        maintain(f);
38        splay(cnr);
39        break;
40      }
41      f = cnr;
42      cnr = ch[cnr][val[cnr] < k];
43      if (!cnr) {
```

```
      val[++tot] = k;
      cnt[tot]++;
      fa[tot] = f;
      ch[f][val[f] < k] = tot;
      maintain(tot);
      maintain(f);
      splay(tot);
      break;
    }
  }
}
int rk(int k) {
  int res = 0, cnr = rt;
  while (1) {
    if (k < val[cnr]) {
      cnr = ch[cnr][0];
    } else {
      res += sz[ch[cnr][0]];
      if (k == val[cnr]) {
        splay(cnr);
        return res + 1;
      }
      res += cnt[cnr];
      cnr = ch[cnr][1];
    }
  }
}
int kth(int k) {
  int cnr = rt;
  while (1) {
    if (ch[cnr][0] && k <= sz[ch[cnr][0]]) {
      cnr = ch[cnr][0];
    } else {
      k -= cnt[cnr] + sz[ch[cnr][0]];
      if (k <= 0) {
        splay(cnr);
        return val[cnr];
      }
      cnr = ch[cnr][1];
    }
  }
}
int pre() {
  int cnr = ch[rt][0];
  while (ch[cnr][1]) cnr = ch[cnr][1];
  splay(cnr);
  return cnr;
}
int nxt() {
  int cnr = ch[rt][1];
  while (ch[cnr][0]) cnr = ch[cnr][0];
  splay(cnr);
  return cnr;
```

```
97    }
98    void del(int k) {
99      rk(k);
100     if (cnt[rt] > 1) {
101       cnt[rt]--;
102       maintain(rt);
103       return;
104     }
105     if (!ch[rt][0] && !ch[rt][1]) {
106       clear(rt);
107       rt = 0;
108       return;
109     }
110     if (!ch[rt][0]) {
111       int cnr = rt;
112       rt = ch[rt][1];
113       fa[rt] = 0;
114       clear(cnr);
115       return;
116     }
117     if (!ch[rt][1]) {
118       int cnr = rt;
119       rt = ch[rt][0];
120       fa[rt] = 0;
121       clear(cnr);
122       return;
123     }
124     int cnr = rt;
125     int x = pre();
126     fa[ch[cnr][1]] = x;
127     ch[x][1] = ch[cnr][1];
128     clear(cnr);
129     maintain(rt);
130   }
131 } tree;
```

1.5 LCT

```
1  namespace LCT{
2     const int N=maxn;
3     int ch[N][2],rev[N],fa[N],stk[N];
4     bool son(int x) {return ch[fa[x]][1] == x;}
5     bool isroot(int x) {return ch[fa[x]][1]!=x && ch[fa[x]][0]!=x; }
6     void reverse(int x) {swap(ch[x][1],ch[x][0]); rev[x] ^= 1;}
7     void pushup(int x) {
8        .......
9     }
10    void pushdown(int x) {if(rev[x])reverse(ch[x][0]) , reverse(ch[x][1]) , rev[x]
          ^= 1;}
11    void rot(int x) {
12        int y = fa[x] , z = fa[y] , c = son(x);
13       if(!isroot(y))ch[z][son(y)] = x; fa[x] = z;
```

```
14        ch[y][c] = ch[x][!c]; fa[ch[y][c]] = y;
15        ch[x][!c] = y; fa[y] = x; pushup(y);
16    }
17    void splay(int x) {
18        int top = 0; stk[++top] = x;
19        for( int i = x; !isroot(i) ; i = fa[i])stk[++top] = fa[i];
20        while(top)pushdown(stk[top--]);
21        for( int y = fa[x]; !isroot(x) ; rot(x) , y = fa[x])
22            if(!isroot(y))son(x) ^ son(y) ? rot(x) : rot(y);
23        pushup(x); return;
24    }
25    void access(int x) { for( int y = 0; x; y = x,x = fa[x])splay(x),ch[x][1] = y,
          pushup(x); }
26    void makeroot(int x) {access(x); splay(x); reverse(x);}
27    int findroot(int x) {access(x); splay(x); while(ch[x][0])x=ch[x][0]; return x;}
28    void split(int x,int y) {makeroot(x); access(y); splay(y);}
29    void link(int x,int y) {if((!x)||(!y))return; makeroot(x); fa[x] = y; }
30    void cut(int x,int y) {if((!x)||(!y))return; split(x,y); ch[y][0] = fa[x] = 0;
          pushup(y);}
31    bool adj(int x,int y) {
32        split(x,y);
33        return ch[y][0]==x&&ch[x][0]==0&&ch[x][1]==0;
34    }
35 }
36 /* 维护子树大小(不是splay的大小)
37 void pushup(int x) {
38    sz[x]=sz[ch[x][0]]+sz[ch[x][1]]+sz2[x]+1;
39 }
40 void access(int x) {
41    for( int y = 0; x; y = x,x = fa[x]) {
42        splay(x);
43        sz2[x]+=sz[ch[x][1]]-sz[y];
44        ch[x][1] = y,pushup(x);
45    }
46 }
47 void link(int x,int y) {
48    split(x,y);
49    fa[x]=y;sz2[y]+=sz[x];
50    pushup(y);
51 }
52 */
```

2 图论

2.1 迪杰斯特拉

```
1 vector<pair<ll,ll>> g[maxn];
2 ll dis[maxn];
3 void dij(int s) {
4    memset(dis,0x3f,sizeof(dis));
5    dis[s]=0;
6    priority_queue<pair<ll,ll>> q;
```

```
    q.push({0,s});
    while(!q.empty()) {
        pair<ll,ll> now=q.top();q.pop();
        now.fi=-now.fi;
        for(auto x:g[now.se]) {
            if(dis[x.fi]>dis[now.se]+x.se) {
                dis[x.fi]=dis[now.se]+x.se;
                q.push({-dis[x.fi],x.fi});
            }
        }
    }
}
```

2.2 SPFA

```
struct edge {
    int cost, to;
    edge(int x = 0, int y = 0): to(x), cost(y) {}
};
vector<edge> g[maxn];
int dis[maxn], cnt[maxn], n, m;
bool vis[maxn];
queue<int> q;
bool spfa(int s) {
    memset(vis, false, sizeof(vis));
    memset(dis, 0x3f, sizeof(dis));
    vis[s] = true;
    dis[s] = 0;
    while (!q.empty()) q.pop();
    q.push(s);
    while (!q.empty()) {
        int now = q.front(); q.pop();
        vis[now] = false;
        for (int i = 0; i < g[now].size(); i++) {
            int to = g[now][i].to;
            if (dis[to] > dis[now] + g[now][i].cost) {
                dis[to] = dis[now] + g[now][i].cost;
                if (!vis[to]) {
                    vis[to] = true;
                    q.push(to);
                    if (++cnt[to] > n) return false;
                }
            }
        }
    }
    return true;
}
```

2.3 tarjan

```
#include <bits/stdc++.h>
```

```cpp
using namespace std;
typedef long long ll;
typedef unsigned long long ull;
#define met(s) memset(s, 0, sizeof(s))
const int mod = 100000007;
const int maxn = 100000;
const int inf = 0x3f3f3f3f;
struct node {
    int to, next;
} edge[maxn];
int head[maxn], tot;
void addedge(int u, int v) {
    edge[++tot].to = v;
    edge[tot].next = head[u];
    head[u] = tot;
}
int Stack[maxn], top; //栈
int dfn[maxn], low[maxn], belong[maxn], scc, Index, num[maxn]; //belong[i]：i属于第i个强连通分量，num[i]:第i个强联通分量的个数
bool Instack[maxn];
void tarjan(int u) {
    int v;
    low[u] = dfn[u] = ++Index;
    Stack[top++] = u;
    Instack[u] = true;
    for (int i = head[u]; i; i = edge[i].next) {
        int v = edge[i].to;
        if (!dfn[v]) {
            tarjan(v);
            if (low[u] > low[v]) low[u] = low[v];
        }
        else if (Instack[v] && low[u] > low[v]) low[u] = low[v];
    }
    if (low[u] == dfn[u]) {
        scc++;
        do {
            v = Stack[--top];
            Instack[v] = false;
            belong[v] = scc;
            num[scc]++;
        } while (v != u);
    }
}
void solve(int n) {
    memset(dfn, 0, sizeof(dfn));
    memset(Instack, false, sizeof(Instack));
    memset(num, 0, sizeof(num));
    Index = scc = top = 0;
    for (int i = 1; i <= n; i++) {
        if (!dfn[i]) tarjan(i);
    }
}
void init() {
```

```
    tot = 0;
    memset(head, 0, sizeof(head));
}
```

2.4 LCA 倍增

```
vector<int> g[maxn];
int dep[maxn], father[maxn][maxbit], lg[maxn];
namespace LCA {
    void init() {
        lg[0] = -1;
        for (int i = 1; i < maxn; i++) lg[i] = lg[i >> 1] + 1;
    }
    void dfs(int now, int fa) {
        dep[now] = dep[fa] + 1;
        father[now][0] = fa;
        for (int i = 1; i <= lg[dep[now]]; i++)
            father[now][i] = father[father[now][i - 1]][i - 1];
        for (int i = 0; i < g[now].size(); i++) {
            if (g[now][i] != fa) dfs(g[now][i], now);
        }
    }
    int LCA(int u, int v) {
        if (dep[u] < dep[v]) swap(u, v);
        while (dep[u] != dep[v]) u = father[u][lg[dep[u] - dep[v]]];
        if (u == v) return u;
        for (int i = lg[dep[u]]; i >= 0; i--) {
            if (father[u][i] != father[v][i]) {
                u = father[u][i];
                v = father[v][i];
            }
        }
        return father[u][0];
    }
}
```

2.5 虚树

```
vi g2[maxn];
namespace faketree {
    int sta[maxn],top=0;
    void insert(int x) {
        if(top==0) {
            sta[++top]=x;
            return ;
        }
        int lca=LCA::LCA(sta[top],x);
        if(lca==sta[top]) {
            sta[++top]=x;
            return ;
        }
```

```
        while(top>1&&dep[lca]<=dep[sta[top-1]]) {
            g2[sta[top-1]].pb(sta[top]);
            top--;
        }
        if(lca!=sta[top]) {
            g2[lca].pb(sta[top]);
            sta[top]=lca;
        }
        sta[++top]=x;
    }
    void end() {
        while(top>1) {
            g2[sta[top-1]].pb(sta[top]);
            top--;
        }
        top=0;
    }
}
```

2.6 树链剖分

```
int dep[maxn],son[maxn],siz[maxn],tp[maxn],fa[maxn];// tp链顶
int id[maxn],rk[maxn],cnt=0;// id dfs序, rk 反dfs序
vector<int> g[maxn];
void dfs1(int now,int f,int d) {
    dep[now]=d,fa[now]=f,siz[now]=1;
    for(int x:g[now]) {
        if(x==f) continue;
        dfs1(x,now,d+1);
        siz[now]+=siz[x];
        if(siz[x]>siz[son[now]]) son[now]=x;
    }
}
void dfs2(int now,int t) {
    tp[now]=t,id[now]=++cnt,rk[cnt]=now;
    if(!son[now]) return ;
    dfs2(son[now],t);
    for(int x: g[now]) {
        if(x==fa[now]||x==son[now]) continue;
        dfs2(x,x);
    }
}
```

2.7 长链剖分

```
int dep[maxn],height[maxn],son[maxn];
vector<int> g[maxn];
void dfs1(int now,int fa) {
    height[now]=dep[now]=dep[fa]+1;
    for(int x:g[now]) {
        if(x==fa) continue;
```

```
7         dfs1(x,now);
8         if(height[x]>height[now]) {
9             height[now]=height[x];
10            son[now]=x;
11        }
12    }
13 }
14 // 求每个节点子树中,距离为k的节点数量,找到一个k,使节点数最多
15 int tmp[maxn],*dp[maxn],*p,ans[maxn];
16 void dfs2(int now,int fa) {
17    dp[now][0]=1;
18    if(son[now]) {
19        dp[son[now]]=dp[now]+1;
20        dfs2(son[now],now);
21        ans[now]=ans[son[now]]+1;
22    }
23    for(int x:g[now]) {
24        int len=height[x]-dep[x]+1;
25        if(x==fa||x==son[now]) continue;
26        dp[x]=p;
27        p+=len;
28        dfs2(x,now);
29        for(int j=0;j<len;j++) {
30            dp[now][j+1]+=dp[x][j];
31            if(dp[now][j+1]>dp[now][ans[now]]||(dp[now][j+1]==dp[now][ans[now]]&&ans[
                  now]>j+1)) {
32                ans[now]=j+1;
33            }
34        }
35    }
36    if(dp[now][ans[now]]==1) ans[now]=0;
37 }
38 int main()
39 {
40    ios::sync_with_stdio(false);cin.tie(0);
41    int n;
42    cin>>n;
43    rep(i,1,n-1) {
44        int u,v;
45        cin>>u>>v;
46        g[u].pb(v);
47        g[v].pb(u);
48    }
49    dfs1(1,0);
50    p=tmp;
51    dp[1]=p;
52    p+=height[1];
53    dfs2(1,0);
54    rep(i,1,n) cout<<ans[i]<<'\n';
55    return 0;
56 }
```

2.8 有向图欧拉回路

```cpp
// 有向图具有欧拉回路的条件:
// 1.所有点出度等于入度
// 2.基图连通
vector<pair<int,bool>> g[maxn];
vector<int>ans;// need reverse
void dfs(int now) {
    for(auto &x:g[now]) {
        if(x.se) continue;
        x.se=true;
        dfs(x.fi);
    }
    ans.pb(now);
}
```

2.9 一般图匹配

```cpp
int n,m;
// match[i], i所对应的匹配点
int match[maxn],pre[maxn],vis[maxn],fa[maxn],tim[maxn],idx,ans;
vi g[maxn];
queue<int>Q;
int find(int x){return x==fa[x]?x:fa[x]=find(fa[x]);}
int lca(int x,int y) {
    for (++idx;;swap(x,y))
        if (x) {
            x=find(x);
            if (tim[x]==idx) return x;
            else tim[x]=idx,x=pre[match[x]];
        }
}
void blossom(int x,int y,int p) {
    while (find(x)!=p) {
        pre[x]=y;y=match[x];
        if (vis[y]==2) vis[y]=1,Q.push(y);
        if (find(x)==x) fa[x]=p;
        if (find(y)==y) fa[y]=p;
        x=pre[y];
    }
}
int Aug(int S) {
    for (int i=1;i<=n;++i)
        vis[i]=pre[i]=0,fa[i]=i;
    while (!Q.empty()) Q.pop();
    Q.push(S);vis[S]=1;
    while (!Q.empty()) {
        int u=Q.front();Q.pop();
        for(int v:g[u]) {
            if (find(u)==find(v)||vis[v]==2) continue;
            if (!vis[v]) {
                vis[v]=2;pre[v]=u;
```

```
            if (!match[v]) {
                for (int x=v,lst;x;x=lst)
                    lst=match[pre[x]],match[x]=pre[x],match[pre[x]]=x;
                return 1;
            }
            vis[match[v]]=1,Q.push(match[v]);
        }
        else {
            int gg=lca(u,v);
            blossom(u,v,gg);blossom(v,u,gg);
        }
    }
  }
  return 0;
}
int MaxMatch() {
  int ans=0;
  rep(i,1,n) {
    if(!match[i]) ans+=Aug(i);
  }
  return ans;
}
```

2.10 网络流

2.10.1 最大流 (Dinic)

```
struct E {
  int to, cp;
  E(int to, int cp): to(to), cp(cp) {}
};

struct Dinic {
  static const int M = 1E5 * 5;
  int m, s, t;
  vector<E> edges;
  vector<int> G[M];
  int d[M];
  int cur[M];
  void init(int n, int s, int t) {
    this->s = s; this->t = t;
    for (int i = 0; i <= n; i++) G[i].clear();
    edges.clear(); m = 0;
  }
  void addedge(int u, int v, int cap) {
    edges.emplace_back(v, cap);
    edges.emplace_back(u, 0);
    G[u].push_back(m++);
    G[v].push_back(m++);
  }
  bool BFS() {
    memset(d, 0, sizeof d);
```

```
26        queue<int> Q;
27        Q.push(s); d[s] = 1;
28        while (!Q.empty()) {
29           int x = Q.front(); Q.pop();
30           for (int& i: G[x]) {
31              E &e = edges[i];
32              if (!d[e.to] && e.cp > 0) {
33                 d[e.to] = d[x] + 1;
34                 Q.push(e.to);
35              }
36           }
37        }
38        return d[t];
39     }
40     int DFS(int u, int cp) {
41        if (u == t || !cp) return cp;
42        int tmp = cp, f;
43        for (int& i = cur[u]; i < G[u].size(); i++) {
44           E& e = edges[G[u][i]];
45           if (d[u] + 1 == d[e.to]) {
46              f = DFS(e.to, min(cp, e.cp));
47              e.cp -= f;
48              edges[G[u][i] ^ 1].cp += f;
49              cp -= f;
50              if (!cp) break;
51           }
52        }
53        return tmp - cp;
54     }
55     int go() {
56        int flow = 0;
57        while (BFS()) {
58           memset(cur, 0, sizeof cur);
59           flow += DFS(s, inf);
60        }
61        return flow;
62     }
63  } DC;
```

2.10.2 上下界网络流

```
1  struct E {
2     int to, cp,id;
3     E(int to, int cp,int id): to(to), cp(cp),id(id) {}
4  };
5
6  struct Dinic {
7     static const int M = 1E5 * 5;
8     int m, s, t;
9     vector<E> edges;
10    vector<int> G[M];
11    int d[M];
```

```
12      int cur[M];
13      void change(int _s,int _t) {
14          s=_s,t=_t;
15      }
16      void init(int n, int s, int t) {
17          this->s = s; this->t = t;
18          for (int i = 0; i <= n; i++) G[i].clear();
19          edges.clear(); m = 0;
20      }
21      void addedge(int u, int v, int cap,int id) {
22          edges.emplace_back(v, cap,id);
23          edges.emplace_back(u, 0,0);
24          G[u].push_back(m++);
25          G[v].push_back(m++);
26      }
27      bool BFS() {
28          memset(d, 0, sizeof d);
29          queue<int> Q;
30          Q.push(s); d[s] = 1;
31          while (!Q.empty()) {
32              int x = Q.front(); Q.pop();
33              for (int& i: G[x]) {
34                  E &e = edges[i];
35                  if (!d[e.to] && e.cp > 0) {
36                      d[e.to] = d[x] + 1;
37                      Q.push(e.to);
38                  }
39              }
40          }
41          return d[t];
42      }
43      int DFS(int u, int cp) {
44          if (u == t || !cp) return cp;
45          int tmp = cp, f;
46          for (int& i = cur[u]; i < G[u].size(); i++) {
47              E& e = edges[G[u][i]];
48              if (d[u] + 1 == d[e.to]) {
49                  f = DFS(e.to, min(cp, e.cp));
50                  e.cp -= f;
51                  edges[G[u][i] ^ 1].cp += f;
52                  cp -= f;
53                  if (!cp) break;
54              }
55          }
56          return tmp - cp;
57      }
58      int go() {
59          int flow = 0;
60          while (BFS()) {
61              memset(cur, 0, sizeof cur);
62              flow += DFS(s, inf);
63          }
64          return flow;
```

```
65      }
66  } DC;
67  int flow[maxn];
68  // 有源汇最大流
69  int main()
70  {
71      ios::sync_with_stdio(false);cin.tie(nullptr);
72      int n,m,s,t;
73      cin>>n>>m>>s>>t;
74      DC.init(n+3,n+1,n+2);
75      DC.addedge(t,s,inf,0);
76      rep(i,1,m) {
77          int u,v,down,up;
78          cin>>u>>v>>down>>up;
79          DC.addedge(u,v,up-down,i);
80          flow[u]-=down;
81          flow[v]+=down;
82      }
83      rep(i,1,n) {
84          if(flow[i]>0) DC.addedge(n+1,i,flow[i],0);
85          if(flow[i]<0) DC.addedge(i,n+2,-flow[i],0);
86      }
87
88      DC.go();
89      bool f=true;
90      for(int x:DC.G[n+1]) {
91          if(DC.edges[x].cp>0) {
92              f=false;
93              break;
94          }
95      }
96      if(f) {
97          int ans=0;
98          for(int x:DC.G[t]) {
99              if(DC.edges[x].id==0&&DC.edges[x].to!=s) {
100                 ans+=DC.edges[x].cp;
101             }
102         }
103         rep(i,1,n+2) {
104             for(int x:DC.G[i]) {
105                 if(DC.edges[x].id==0) {
106                     DC.edges[x].cp=0;
107                 }
108             }
109         }
110         DC.change(s,t);
111         ans+=DC.go();
112         cout<<ans<<'\n';
113     }
114     else cout<<"please go home to sleep"<<'\n';
115     return 0;
116 }
```

2.10.3 费用流

```
struct E {
    int from, to, cp, v;
    E() {}
    E(int f, int t, int cp, int v) : from(f), to(t), cp(cp), v(v) {}
};
struct MCMF {
    int n, m, s, t;
    vector<E> edges;
    vector<int> G[M];
    bool inq[M];
    int d[M], p[M], a[M];
    void init(int _n, int _s, int _t) {
        n = _n; s = _s; t = _t;
        rep(i, 0, n) G[i].clear();
        edges.clear(); m = 0;
    }
    void addedge(int from, int to, int cap, int cost) {
        edges.emplace_back(from, to, cap, cost);
        edges.emplace_back(to, from, 0, -cost);
        G[from].push_back(m++);
        G[to].push_back(m++);
    }
    bool BellmanFord(int &flow, int &cost) {
        rep (i,0,n) d[i] = inf;
        memset(inq, 0, sizeof inq);
        d[s] = 0, a[s] = inf, inq[s] = true;
        queue<int> Q; Q.push(s);
        while (!Q.empty()) {
            int u = Q.front(); Q.pop();
            inq[u] = false;
            for (int& idx: G[u]) {
                E &e = edges[idx];
                if (e.cp && d[e.to] > d[u] + e.v) {
                    d[e.to] = d[u] + e.v;
                    p[e.to] = idx;
                    a[e.to] = min(a[u], e.cp);
                    if (!inq[e.to]) {
                        Q.push(e.to);
                        inq[e.to] = true;
                    }
                }
            }
        }
        if (d[t] == inf) return false;
        flow += a[t];
        cost += a[t] * d[t];
        int u = t;
        while (u != s) {
            edges[p[u]].cp -= a[t];
            edges[p[u] ^ 1].cp += a[t];
            u = edges[p[u]].from;
```

```
52          }
53          return true;
54      }
55      int go() {
56          int flow = 0, cost = 0;
57          while (BellmanFord(flow, cost));
58          return cost;
59      }
60  } MM;
```

2.11 二分图

2.11.1 匈牙利

```
1   int book[505][505], k, m, n;
2   int cx[505], cy[505], vis[505];
3   void init() {
4       memset(cx, -1, sizeof(cx));
5       memset(cy, -1, sizeof(cy));
6       memset(vis, 0, sizeof(vis));
7       memset(book, 0, sizeof(book));
8   }
9   int line(int x) {
10      int i;
11      for (i = 1; i <= n; i++) {
12          if (book[x][i] && !vis[i]) {
13              vis[i] = 1;
14              if (cy[i] == -1 || line(cy[i])) {
15                  cx[x] = i;
16                  cy[i] = x;
17                  return 1;
18              }
19          }
20      }
21      return 0;
22  }
23  int maxmatch() {
24      int sum = 0, i;
25      for (i = 1; i <= m; i++) {
26          if (cx[i] == -1) {
27              memset(vis, 0, sizeof(vis));
28              sum += line(i);
29          }
30      }
31      return sum;
32  }
```

2.11.2 HK

```
1   namespace HK {
2       vector<int> G[maxn];
3       int uN;
```

```
4     int Mx[maxn],My[maxn];
5     int dx[maxn],dy[maxn];
6     int dis;
7     bool used[maxn];
8     bool searchp() {
9        queue<int> q;
10       dis=inf;
11       memset(dx,-1,sizeof(dx));
12       memset(dy,-1,sizeof(dy));
13       for(int i=0;i<uN;i++) {
14          if(Mx[i]==-1) {
15             q.push(i);
16             dx[i]=0;
17          }
18       }
19       while(!q.empty()) {
20          int u=q.front();q.pop();
21          if(dx[u]>dis) break;
22          for(int v:G[u]) {
23             if(dy[v]==-1) {
24                dy[v]=dx[u]+1;
25                if(My[v]==-1) dis=dy[v];
26                else {
27                   dx[My[v]]=dy[v]+1;
28                   q.push(My[v]);
29                }
30             }
31          }
32       }
33       return dis!=inf;
34    }
35    bool DFS(int u) {
36       for(int v:G[u]) {
37          if(!used[v]&&dy[v]==dx[u]+1) {
38             used[v]=true;
39             if(My[v]!=-1&&dy[v]==dis) continue;
40             if(My[v]==-1||DFS(My[v])) {
41                My[v]=u;
42                Mx[u]=v;
43                return true;
44             }
45          }
46       }
47       return false;
48    }
49    int MaxMatch() {
50       int res=0;
51       memset(Mx,-1,sizeof(Mx));
52       memset(My,-1,sizeof(My));
53       while(searchp()) {
54          memset(used,false,sizeof(used));
55          for(int i=0;i<uN;i++) {
56             if(Mx[i]==-1&&DFS(i)) res++;
```

```
            }
        }
        return res;
    }

}
```

2.11.3 KM(DFS)

```
int nx, ny; //两边的点的数目
int g[maxn][maxn], linker[maxn];//linker[i]:右边第i个点的匹配对象
int lx[maxn], ly[maxn], slack[maxn];
bool visx[maxn], visy[maxn]; //x,y是否被访问
bool dfs(int x) {
    visx[x] = true;
    for (int y = 0; y < ny; y++) {
        if (visy[y]) continue;
        int tmp = lx[x] + ly[y] - g[x][y];
        if (tmp == 0) {
            visy[y] = true;
            if (linker[y] == -1 || dfs(linker[y])) {
                linker[y] = x;
                return true;
            }
        }
        else if (slack[y] > tmp) slack[y] = tmp;
    }
    return false;
}
int km() {
    memset(linker, -1, sizeof(linker));
    memset(ly, 0, sizeof(ly));
    for (int i = 0; i < nx; i++) {
        lx[i] = -inf;
        for (int j = 0; j < ny; j++) {
            if (g[i][j] > lx[i]) lx[i] = g[i][j];
        }
    }
    for (int x = 0; x < nx; x++) {
        for (int i = 0; i < ny; i++) slack[i] = inf;
        while (1) {
            memset(visx, false, sizeof(visx));
            memset(visy, false, sizeof(visy));
            if (dfs(x)) break;
            int d = inf;
            for (int i = 0; i < ny; i++) {
                if (!visy[i] && d > slack[i]) d = slack[i];
            }
            for (int i = 0; i < nx; i++) {
                if (visx[i]) lx[i] -= d;
            }
            for (int i = 0; i < ny; i++) {
```

```
            if (visy[i]) ly[i] += d;
            else slack[i] -= d;
         }
      }
   }
   int res = 0;
   for (int i = 0; i < ny; i++) {
      if (linker[i] != -1) res += g[linker[i]][i];
   }
   return res;
}
```

2.11.4 KM(BFS)

```
ll w[maxn][maxn];
ll lx[maxn],ly[maxn];
int linker[maxn];//linker[i]:右边第i个点的匹配对象
ll slack[maxn];
int n;
bool visy[maxn];
int pre[maxn];
void bfs(int k) {
   int x,y=0,yy=0,delta;
   memset(pre,0,sizeof(pre));
   for(int i=1;i<=n;i++) slack[i]=inf;
   linker[y]=k;
   while(1) {
      x=linker[y];delta=inf;visy[y]=true;
      for(int i=1;i<=n;i++) {
         if(!visy[i]) {
            if(slack[i]>lx[x]+ly[i]-w[x][i]) {
               slack[i]=lx[x]+ly[i]-w[x][i];
               pre[i]=y;
            }
            if(slack[i]<delta) delta=slack[i],yy=i;
         }
      }
      for(int i=0;i<=n;i++) {
         if(visy[i])lx[linker[i]]-=delta,ly[i]+=delta;
         else slack[i]-=delta;
      }
      y=yy;
      if(linker[y]==-1) break;
   }
   while(y) linker[y]=linker[pre[y]],y=pre[y];
}
ll km() {
   memset(lx,0,sizeof(lx));
   memset(ly,0,sizeof(ly));
   memset(linker,-1,sizeof(linker));
   for(int i=1;i<=n;i++) {
      memset(visy,false,sizeof(visy));
```

```
39        bfs(i);
40    }
41    ll ans=0;
42    for(int i=1;i<=n;i++) ans+=w[linker[i]][i];
43    return ans;
44 }
```

3 字符串

3.1 kmp

```
1  int Next[maxn];
2  void getNext(string a) {
3     int len = a.size(), k = -1, i = 0;
4     Next[0] = -1;
5     while (i < len)
6     {
7        while (k != -1 && a[i] != a[k]) k = Next[k];
8        Next[++i] = ++k;
9     }
10 }
11 int kmp(string a1,string a2) {
12    int i, j = 0;
13    for (i = 0; i < a1.size(); i++) {
14       while (j != -1 && a1[i] != a2[j]) j = Next[j];
15       j++;
16       if (j == a2.size()) return i - a2.size() + 2;
17    }
18    return -1;
19 }
```

3.2 exkmp

```
1  // next[i]: x[i....m-1]与x[0....m-1]的最长公共前缀
2  // extend[i]: y[i...n-1]与x[0....m-1]的最长公共前缀
3  void pre_exkmp(char x[],int m,int next[]) {
4     next[0]=m;
5     int j=0;
6     while(j+1<m&&x[j]==x[j+1]) j++;
7     next[1]=j;
8     int k=1;
9     for(int i=2;i<m;i++) {
10       int p=next[k]+k-1,L=next[i-k];
11       if(i+L<p+1) next[i]=L;
12       else {
13          j=max(0,p-i+1);
14          while(i+j<m&&x[i+j]==x[j]) j++;
15          next[i]=j;
16          k=i;
17       }
18    }
```

```
}
void EKMP(char x[],int m,char y[],int n,int next[],int extend[]) {
    pre_exkmp(x,m,next);
    int j=0;
    while(j<n&&j<m&&x[j]==y[j]) j++;
    extend[0]=j;
    int k=0;
    for(int i=1;i<n;i++) {
        int p=extend[k]+k-1,L=next[i-k];
        if(i+L<p+1) extend[i]=L;
        else {
            j=max(0,p-i+1);
            while(i+j<n&&j<m&&y[i+j]==x[j]) j++;
            extend[i]=j;
            k=i;
        }
    }
}
```

3.3 最小表示法

```
int minpos(string a) {
    int k=0,i=0,j=1,n=a.size();
    while(k<n&&i<n&&j<n) {
        if(a[(i+k)%n]==a[(j+k)%n]) k++;
        else {
            if(a[(i+k)%n]>a[(j+k)%n]) i=i+k+1;
            else j=j+k+1;
            if(i==j) i++;
            k=0;
        }
    }
    i=min(i,j);
}
string trans(string a,int pos) {
    string ans;
    rep(i,0,a.size()-1) ans+=a[(i+pos)%a.size()];
    return ans;
}
```

3.4 AC 自动机

```
struct ACAM {
    int next[maxn][26],fail[maxn],id[maxn],root,cnt;
    int match[maxn];
    int newnode() {
        rep(i,0,25) next[cnt][i]=-1;
        id[cnt++]=0;
        return cnt-1;
    }
    void init() {
```

```
        cnt=0;
        root=newnode();
    }
    void ins(char a[],int x) {
        int len=strlen(a),now=root;
        rep(i,0,len-1) {
            if(next[now][a[i]-'a']==-1)
                next[now][a[i]-'a']=newnode();
            now=next[now][a[i]-'a'];
        }
        id[now]=x;
    }
    void build() {
        queue<int> q;
        fail[root]=root;
        rep(i,0,25) {
            if(next[root][i]==-1) {
                next[root][i]=root;
            }
            else {
                fail[next[root][i]]=root;
                match[next[root][i]]=root;
                q.push(next[root][i]);
            }
        }
        while(!q.empty()) {
            int now=q.front();
            q.pop();
            rep(i,0,25) {
                if(next[now][i]==-1) next[now][i]=next[fail[now]][i];
                else {
                    fail[next[now][i]]=next[fail[now]][i];
                    q.push(next[now][i]);
                    int tmp=fail[next[now][i]];
                    if(id[tmp]>0) match[next[now][i]]=tmp;
                    else match[next[now][i]]=match[tmp];
                }
            }
        }
    }
}AC;
```

3.5 SA

```
struct SuffixArray
{
    int sa[maxn], rank[maxn], ws[maxn], wv[maxn], wa[maxn], wb[maxn], height[maxn],
        st[maxbit][maxn], N;
    bool cmp(int *r, int a, int b, int l){return r[a]==r[b] and r[a+l]==r[b+l];}
    void build(int *r, int n, int m)
    {
        N=n;
```

```
8       n++;
9       int i, j, k=0, p, *x=wa, *y=wb, *t;
10      for(i=0;i<m;i++)ws[i]=0;
11      for(i=0;i<n;i++)ws[x[i]=r[i]]++;
12      for(i=1;i<m;i++)ws[i]+=ws[i-1];
13      for(i=n-1;i>=0;i--)sa[--ws[x[i]]]=i;
14      for(p=j=1;p<n;j<<=1,m=p)
15      {
16          for(p=0,i=n-j;i<n;i++)y[p++]=i;
17          for(i=0;i<n;i++)if(sa[i]>=j)y[p++]=sa[i]-j;
18          for(i=0;i<n;i++)wv[i]=x[y[i]];
19          for(i=0;i<m;i++)ws[i]=0;
20          for(i=0;i<n;i++)ws[wv[i]]++;
21          for(i=1;i<m;i++)ws[i]+=ws[i-1];
22          for(i=n-1;i>=0;i--)sa[--ws[wv[i]]]=y[i];
23          for(t=x,x=y,y=t,p=1,i=1,x[sa[0]]=0;i<n;i++)
24              x[sa[i]]=cmp(y,sa[i-1],sa[i],j)?p-1:p++;
25      }
26
27      for(i=0;i<n;i++)rank[sa[i]]=i;
28
29      for(i=0;i<n-1;height[rank[i++]]=k)
30          for(k?k--:0,j=sa[rank[i]-1];r[i+k]==r[j+k];k++);
31  }
32  void build_st() //st表
33  {
34      int i, k;
35      for(i=1;i<=N;i++)st[0][i]=height[i];
36      for(k=1;k<=maxbit;k++)
37          for(i=1;i+(1<<k)-1<=N;i++)
38              st[k][i]=min(st[k-1][i],st[k-1][i+(1<<k)-1]);
39  }
40  int lcp(int x, int y) //最长公共前缀
41  {
42      int l=rank[x], r=rank[y];
43      if(l>r)swap(l,r);
44      if(l==r)return N-sa[l];
45      int t=log2(r-l);
46      return min(st[t][l+1],st[t][r-(1<<t)+1]);
47  }
48 }SA;
```

3.6 SAM

```
1  namespace sam {
2      const int M = maxn << 1;
3      int t[M][26], len[M] = {-1}, fa[M], sz = 2, last = 1;
4      ll endpos[M];
5      void init() { memset(t, 0, (sz + 10) * sizeof t[0]); sz = 2; last = 1; }
6      void ins(int ch) {
7          int p = last, np = last = sz++;endpos[np]=1;
8          len[np] = len[p] + 1;
```

```
        for (; p && !t[p][ch]; p = fa[p]) t[p][ch] = np;
        if (!p) { fa[np] = 1; return; }
        int q = t[p][ch];
        if (len[p] + 1 == len[q]) fa[np] = q;
        else {
            int nq = sz++; len[nq] = len[p] + 1;
            memcpy(t[nq], t[q], sizeof t[0]);
            fa[nq] = fa[q];
            fa[np] = fa[q] = nq;
            for (; t[p][ch] == q; p = fa[p]) t[p][ch] = nq;
        }
    }
    //拓扑排序
    int c[M] = {1}, a[M];
    void rsort() {
        rep (i, 1, sz-1) c[i] = 0;
        rep (i, 1, sz-1) c[len[i]]++;
        rep (i, 1, sz-1) c[i] += c[i - 1];
        rep (i, 1, sz-1) a[--c[len[i]]] = i;
    }
    //计算endpos大小
    void calendpos() {
        rsort();
        per(i,sz-1,2) {
            endpos[fa[a[i]]]+=endpos[a[i]];
        }
    }
}
```

3.7 manacher

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef unsigned long long ull;
#define met(s) memset(s, 0, sizeof(s))
const int MOD = 100000007;
const int MAXN = 1000000;
const int inf = 0x3f3f3f3f;
char str_new[220010], str[110010];
int p[220025];
int init() {
    int len = strlen(str);
    str_new[0] = '$';
    str_new[1] = '#';
    int t = 2;
    for (int i = 0; i < len; i++) {
        str_new[t++] = str[i];
        str_new[t++] = '#';
    }
    return t;
}
```

```cpp
int manacher() {
    memset(str_new, 0, sizeof(str_new));
    int len = init();
    int id, mx = 0, Max = 0;
    for (int i = 0; i < len; i++) {
        if (i < mx) p[i] = min(p[2 * id - i], mx - i);
        else p[i] = 1;
        while (str_new[i - p[i]] == str_new[i + p[i]]) p[i]++;
        if (mx < i + p[i]) {
            mx = i + p[i];
            id = i;
        }
        Max = max(Max, p[i] - 1);
    }
    for(int i=0;i<len;i++) p[i]--;
    return Max;
}
int main() {
    while (scanf("%s", str) != EOF) {
        int ans = manacher();
        printf("%d\n", ans );
    }
    return 0;
}
```

3.8 PAM

```cpp
// cnt[i]: 表示该节点在原串中出现的次数....记得count()
namespace pam {
    const int N=maxn<<1;
    int t[N][26], fa[N], len[N], rs[N], cnt[N], num[N];
    int sz, n, last;
    int _new(int l) {
        len[sz] = l; cnt[sz] = num[sz] = 0;
        return sz++;
    }
    void init() {
        memset(t, 0, sz * sizeof t[0]);
        rs[n = sz = 0] = -1;
        last = _new(0);
        fa[last] = _new(-1);
    }
    int get_fa(int x) {
        while (rs[n - 1 - len[x]] != rs[n]) x = fa[x];
        return x;
    }
    void ins(int ch) {
        rs[++n] = ch;
        int p = get_fa(last);
        if (!t[p][ch]) {
            int np = _new(len[p] + 2);
            num[np] = num[fa[np] = t[get_fa(fa[p])][ch]] + 1;
```

```
            t[p][ch] = np;
        }
        ++cnt[last = t[p][ch]];
        // len[last];以当前字符结尾的最长回文后缀
        // num[last]; 以当前字符结尾的回文串数量(回文后缀)
    }
    void count() {
        for(int i=sz-1;i>=2;i--)
            cnt[fa[i]]+=cnt[i];
    }
}
```

3.9 lyndon 分解

```
vector<string> duval(string const& s) {
  int n = s.size(), i = 0;
  vector<string> factorization;
  while (i < n) {
    int k = i + 1, j = i;
    while (k < n && s[j] <= s[k]) {
      if (s[j] < s[k])
        j = i;
      else
        j++;
      k++;
    }
    while (i <= j) {
      factorization.push_back(s.substr(i, k - j));
      i += k - j;
    }
  }
  return factorization;
}
```

4 数学

4.1 素数表

```
int prime[maxn],p[maxn];
void init(int up) {
    rep(i,1,up) p[i]=i;
    rep(i,2,up) {
        if(p[i]==i) prime[++prime[0]]=i;
        for(int j=1;j<=prime[0]&&i<=up/prime[j];j++) {
            p[i*prime[j]]=prime[j];
            if(i%prime[j]==0) break;
        }
    }
}
vector<pii> smallfc(int x) {
    vector<pii> ans;
```

```
    while(x!=1) {
        pii now;
        now.fi=p[x];
        while(x%now.fi==0) {
            now.se++;
            x/=now.fi;
        }
        ans.pb(now);
    }
    reverse(ans.begin(),ans.end());
    return ans;
}
vector<pair<ll,int>> bigfc(ll x) {
    vector<pair<ll,int>> ans;
    rep(i,1,prime[0]) {
        if(prime[i]>x/prime[i]) break;
        if(x%prime[i]!=0) continue;
        int cnt=0;
        while(x%prime[i]==0) {
            cnt++;
            x/=prime[i];
        }
        ans.pb({prime[i],cnt});
    }
    return ans;
}
```

4.2 欧拉函数

```
//phi[1]=1, 1与1是互质的
int phi[maxn];
void euler(int n) {
    memset(phi,0,sizeof(phi));
    phi[1]=1;
    for(int i=2;i<=n;i++) {
        if(!phi[i]) {
            for(int j=i;j<=n;j++) {
                if(!phi[j]) phi[j]=j;
                phi[j]=phi[j]/i*(i-1);
            }
        }
    }
}
```

4.3 PollardRho

```
int t,s=20,cnt;
long long fac[1001];
long long ksc(long long x,long long y,long long mod){
    long long res=0;
    while(y) {
```

```
6         if(y&1)
7         res=(res+x)%mod;
8         x=(x<<1)%mod;
9         y>>=1;
10    }
11    return res;
12 }
13 long long ksm(long long x,long long y,long long mod) {
14    long long res=1;
15    while(y) {
16       if(y&1)
17       res=ksc(res,x,mod);
18       x=ksc(x,x,mod);
19       y>>=1;
20    }
21    return res;
22 }
23 int miller_rabin(long long n) {
24    if(n==2)
25    return 1;
26    if(n<2||!(n%2))
27    return 0;
28    long long u,pre,x;
29    int k=0;
30    u=n-1;
31    while(!(u&1)) {
32       ++k;
33       u>>=1;
34    }
35    for(int i=1;i<=s;++i) {
36       x=rand()%(n-2)+2;
37       x=ksm(x,u,n);
38       pre=x;
39       for(int j=1;j<=k;++j) {
40          x=ksc(x,x,n);
41          if(x==1&&pre!=1&&pre!=n-1)
42          return 0;
43          pre=x;
44       }
45       if(x!=1)
46       return 0;
47    }
48    return 1;
49 }
50 long long gcd(long long a,long long b){//注意与一般的gcd不一样
51    if (a==0) return 1;//pollard_rho的需要
52    if (a<0) return gcd(-a,b);//可能有负数
53    while (b){
54       long long t=a%b; a=b; b=t;
55    }
56    return a;
57 }
58 long long pollard_rho(long long n,long long c) {//找因子
```

```
59
60      long long i=1,k=2;//用来判断是否形成了环
61      long long xx=rand()%n,y=xx;
62      while(1) {
63          i++;
64          xx=(ksc(xx,xx,n)+c)%n;
65          long long d=gcd(y-xx,n);
66          if(1<d&&d<n)//找到一个因数
67          return d;
68          if(y==xx)//出现循环，那么这次寻找失败
69          return n;
70          if(i==k){//相当于每次找连续k这么多次取模有没有得到相同余数
71              y=xx;
72              k<<=1;
73          }
74      }
75  }
76  void find(long long n) {//通过找因数来找质因子
77      if(miller_rabin(n)) {
78          fac[++cnt]=n;//记录质因子
79          return;
80      }
81      long long p=n;
82      while(p>=n)
83      p=pollard_rho(p,rand()%(n-1)+1);//如果转了一圈还是p，则继续while循环
84      //p是当前找到的一个因数（不一定是质因数），再分别找p和n/p的质因数
85      find(p);
86      find(n/p);
87  }
88  int main()
89  {
90      srand(time(0)+19260817);
91      scanf("%d",&t);
92      while(t--)
93      {
94          long long x;
95          scanf("%lld",&x);
96          if(miller_rabin(x))
97          {
98              printf("Prime\n");
99              continue;
100         }
101         cnt=0;
102         find(x);
103         sort(fac+1,fac+cnt+1);
104         for(int i=1;i<=cnt;i++) cout<<fac[i]<<'\n';
105     }
106     return 0;
107 }
```

4.4 线性求逆元

```
inv[1]=1;for(int i=2;i<maxn;i++)inv[i]=inv[mod%i]*(mod-mod/i)%mod;
```

4.5 组合数学

```
ll jc[maxn],inv[maxn];
void cinit(int n) {
    jc[0]=inv[0]=1;
    rep(i,1,n) jc[i]=jc[i-1]*i%mod;
    inv[n]=qpow(jc[n],mod-2);
    per(i,n-1,1) inv[i]=inv[i+1]*(i+1)%mod;
}
ll C(int n,int m,int p=mod) { return m>n?0:jc[n]*inv[m]%p*inv[n-m]%p; }
ll lucas(ll n,ll m,ll p=mod){ return m==0?1:lucas(n/p,m/p,p)*C(n%p,m%p,p)%p; }
```

4.6 FFT

```
const double PI=acos(-1.0);
struct Complex {
    double x,y;
    Complex(double _x=0.0,double _y=0.0){ x=_x,y=_y; }
    Complex operator -(const Complex &b)const { return Complex(x-b.x,y-b.y); }
    Complex operator +(const Complex &b)const { return Complex(x+b.x,y+b.y); }
    Complex operator *(const Complex &b)const { return Complex(x*b.x-y*b.y,x*b.y+y*b
        .x); }
};
void change(Complex y[],int len) {
    int i,j,k;
    for(i=1,j=len/2;i<len-1;i++) {
        if(i<j) swap(y[i],y[j]);
        k=len/2;
        while(j>=k) {
            j-=k;
            k/=2;
        }
        if(j<k) j+=k;
    }
}
void fft(Complex y[],int len,int on) {
    change(y,len);
    for(int h=2;h<=len;h<<=1) {
        Complex wn(cos(-on*2*PI/h),sin(-on*2*PI/h));
        for(int j=0;j<len;j+=h) {
            Complex w(1,0);
            for(int k=j;k<j+h/2;k++) {
                Complex u=y[k];
                Complex t=w*y[k+h/2];
                y[k]=u+t;
                y[k+h/2]=u-t;
                w=w*wn;
            }
        }
```

```
    }
    if(on==-1) {
        for(int i=0;i<len;i++) y[i].x/=len;
    }
}
void conv(Complex a[],Complex b[],int n) {
    fft(a,n,1);
    fft(b,n,1);
    for(int i=0;i<n;i++) a[i]=a[i]*b[i];
    fft(a,n,-1);
}
```

4.7 NTT

```
ll qpow(ll a,int b) {
    ll ans=1;
    while(b) {
        if(b&1) ans=ans*a%mod;
        a=a*a%mod;
        b>>=1;
    }
    return ans;
}
const int G=3;
namespace NTT {
    ll wn[maxn << 2], rev[maxn << 2];
    int init(int n_) {
        int step = 0; int n = 1;
        for ( ; n < n_; n <<= 1) ++step;
        rep(i,1,n-1)
            rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << (step - 1));
        int g = qpow(G, (mod - 1) / n);
        wn[0] = 1;
        for (int i = 1; i <= n; ++i)
            wn[i] = wn[i - 1] * g % mod;
        return n;
    }
    void NTT(ll a[], int n, int f) {
        rep(i, 0, n-1)if(i < rev[i])
            std::swap(a[i], a[rev[i]]);
        for (int k = 1; k < n; k <<= 1) {
            for (int i = 0; i < n; i += (k << 1)) {
                int t = n / (k << 1);
                rep(j, 0, k-1) {
                    ll w = f == 1 ? wn[t * j] : wn[n - t * j];
                    ll x = a[i + j];
                    ll y = a[i + j + k] * w % mod;
                    a[i + j] = (x + y) % mod;
                    a[i + j + k] = (x - y + mod) % mod;
                }
            }
        }
```

```
39      if (f == -1) {
40          ll ninv = qpow(n, mod-2);
41          rep (i, 0, n-1) a[i] = a[i] * ninv % mod;
42      }
43  }
44  ll tmp1[maxn<<2],tmp2[maxn<<2];
45  void mul(ll a[],ll b[],int lena,int lenb) {// lena为a的项数
46      int len=init(lena+lenb+2);
47      rep(i,0,lena-1) tmp1[i]=a[i];
48      rep(i,lena,len-1) tmp1[i]=0;
49      rep(i,0,lenb-1) tmp2[i]=b[i];
50      rep(i,lenb,len-1) tmp2[i]=0;
51      NTT(tmp1,len,1); NTT(tmp2,len,1);
52      rep(i,0,len-1) a[i]=tmp1[i]*tmp2[i]%mod;
53      NTT(a,len,-1);
54  }
55  void mul(ll a[],ll b[],ll c[],int lena,int lenb) {
56      int len=init(lena+lenb+2);
57      rep(i,0,lena-1) tmp1[i]=a[i];
58      rep(i,lena,len-1) tmp1[i]=0;
59      rep(i,0,lenb-1) tmp2[i]=b[i];
60      rep(i,lenb,len-1) tmp2[i]=0;
61      NTT(tmp1,len,1); NTT(tmp2,len,1);
62      rep(i,0,len-1) c[i]=tmp1[i]*tmp2[i]%mod;
63      NTT(c,len,-1);
64  }
65  ll P[20][maxn<<2];
66  void solve(ll *a,int l,int r,int dep) {//(1+a[0]*x)(1+a[1]*x)(1+a[2]*x)
67      if(l==r) {
68          P[dep][2*l]=1;
69          P[dep][2*l+1]=a[l];
70          return ;
71      }
72      int mid=(l+r)>>1;
73      solve(a,l,mid,dep+1);
74      solve(a,mid+1,r,dep+1);
75      int lenl=mid-l+1,lenr=r-mid;
76      mul(P[dep+1]+2*l,P[dep+1]+2*mid+2,P[dep]+l*2,lenl+1,lenr+1);
77  }
78  ll tmp[maxn << 2];
79  void inv(ll *a,ll *b,int len) {//lena为a的项数
80      if(len==1) { b[0]=qpow(a[0],mod-2); return ;}
81      inv(a,b,(len+1)>>1);
82      int l=init(len*2+1);
83      rep(i,0,len-1) tmp[i]=a[i];
84      rep(i,len,l-1) tmp[i]=0;
85      NTT(tmp,l,1);
86      NTT(b,l,1);
87      rep(i,0,l-1) b[i]=(2-b[i]*tmp[i]%mod+mod)%mod*b[i]%mod;
88      NTT(b,l,-1);
89      rep(i,len,l-1) b[i]=0;
90  }
91  ll pa[maxn<<2],pb[maxn<<2],pb_inv[maxn<<2];
```

```
92    void div(ll *a,ll *b,int lena,int lenb,ll *Q,ll *R) {//lena为a的最高次数
93       rep(i,0,lena) pa[i]=a[lena-i];
94       rep(i,0,lenb) pb[i]=b[lenb-i];
95       rep(i,lena-lenb+1,lenb) pb[i]=0;
96       inv(pb,pb_inv,lena-lenb+1);
97       mul(pa,pb_inv,lena+1,lena-lenb+1);
98       rep(i,0,lena-lenb) Q[i]=pa[lena-lenb-i];
99       rep(i,0,lenb) pb[i]=b[i];
100      mul(pb,Q,lenb+1,lena-lenb+1);
101      rep(i,0,lenb-1) R[i]=(a[i]-pb[i]+mod)%mod;
102   }
103   ll invb[maxn<<2],D[maxn<<2],inv2=qpow(2,mod-2);
104   void sqrt(ll *a,ll *b,int len) {//要保证b[0]=1,否则需要二次剩余求
105      if(len==1) {b[0]=1;return ;}
106      sqrt(a,b,(len+1)>>1);
107      rep(i,0,len<<1) invb[i]=0;
108      inv(b,invb,len);
109      int l=init(2*len);
110      rep(i,0,len-1) D[i]=a[i];
111      rep(i,len,l-1) D[i]=0;
112      NTT(D,l,1);NTT(b,l,1);NTT(invb,l,1);
113      rep(i,0,l-1) b[i]=(b[i]+D[i]*invb[i]%mod)%mod*inv2%mod;
114      NTT(b,l,-1);
115      rep(i,len,l-1) b[i]=0;
116   }
117   void dao(ll *a,ll *b,int len) {
118      rep(i,0,len-1) b[i-1]=i*a[i]%mod;b[len-1]=0;
119   }
120   void jifen(ll *a,ll *b,int len) {
121      rep(i,0,len-1) b[i]=a[i-1]*qpow(i,mod-2)%mod;b[0]=0;
122   }
123   ll A[maxn<<2],B[maxn<<2];
124   void ln(ll *a,ll *b,int len) {
125      int n;for(n=1;n<=len;n<<=1);
126      rep(i,0,n<<1) A[i]=B[i]=0;
127      dao(a,A,n);inv(a,B,n);
128      int l=init(n*2);
129      NTT(A,l,1),NTT(B,l,1);
130      rep(i,0,l-1) A[i]=A[i]*B[i]%mod;
131      NTT(A,l,-1);jifen(A,b,n);
132   }
133   ll F[maxn<<2];
134   void exp(ll *a,ll *b,int len) {//保证a[0]=0;
135      if(len==1) { b[0]=1;return ;}
136      exp(a,b,(len+1)>>1);
137      ln(b,F,len);
138      int l=init(2*len+1);
139      rep(i,0,len-1) F[i]=(a[i]-F[i]+mod)%mod;
140      rep(i,len,l-1) F[i]=b[i]=0;
141      F[0]++;
142      NTT(F,l,1),NTT(b,l,1);
143      rep(i,0,l-1) b[i]=b[i]*F[i]%mod;
144      NTT(b,l,-1);
```

```
145         rep(i,len,l-1) b[i]=0;
146     }
147 }
```

4.8 FWT

```
1  template<typename T>
2  void fwt(ll a[], int n, T f) {
3     for (int d = 1; d < n; d *= 2)
4        for (int i = 0, t = d * 2; i < n; i += t)
5           rep (j, 0, d-1)
6              f(a[i + j], a[i + j + d]);
7  }
8
9  void AND(ll& a, ll& b) { a =(a+b)%mod; }
10 void OR(ll& a, ll& b) { b =(a+b)%mod; }
11 void XOR (ll& a, ll& b) {
12    ll x = a, y = b;
13    a = (x + y) % mod;
14    b = (x - y + mod) % mod;
15 }
16 void rAND(ll& a, ll& b) { a =(a-b+mod)%mod; }
17 void rOR(ll& a, ll& b) { b =(b-a+mod)%mod; }
18 void rXOR(ll& a, ll& b) {
19    static ll INV2 = (mod + 1) / 2;
20    ll x = a, y = b;
21    a = (x + y) * INV2 % mod;
22    b = (x - y + mod) * INV2 % mod;
23 }
```

4.9 第二类斯特林数

```
1  ll sb[maxn<<2],sr[maxn<<2];
2  // 计算 k!*S(n,k),k=0,1,2,...,n, 复杂度(n*log n)
3  void stirling_second(int n){
4     int len=NTT::init(2*n+2);
5     for(int i=1; i<=n; i++) sr[i]=qpow(i, n)*inv[i]%mod;
6     for(int i=0; i<=n; i++){
7        if(i&1) sb[i]=mod-inv[i];
8        else sb[i]=inv[i];
9     }
10    NTT::mul(sr,sb,len);
11 }
12 // 单独计算 m!*S(n,m) 复杂度 O(m*log n)
13 ll calsr(ll n,ll m) {
14    ll ans=0;
15    rep(k,0,m) {
16       ll now=C(m,k)*qpow(m-k,n)%mod;
17       ans=(k&1)?(ans-now+mod)%mod:(ans+now)%mod;
18    }
19    return ans;
```

```
}
```

4.10 BSGS

```
ll BSGS(ll a, ll b, ll p) { // a^x = b (mod p)
    a %= p;
    if (!a && !b) return 1;
    if (!a) return -1;
    static map<ll, ll> mp; mp.clear();
    ll m = sqrt(p + 1.5);
    ll v = 1;
    rep (i, 1, m) {
        v = v * a % p;
        mp[v * b % p] = i;
    }
    ll vv = v;
    rep (i, 1, m ) {
        auto it = mp.find(vv);
        if (it != mp.end()) return i * m - it->second;
        vv = vv * v % p;
    }
    return -1;
}
```

4.11 高斯消元

```
typedef double LD;
const LD eps = 1E-10;
const int maxn = 2000 + 10;

int n, m;
LD a[maxn][maxn], x[maxn];
bool free_x[maxn];

inline int sgn(LD x) { return (x > eps) - (x < -eps); }

int gauss(LD a[maxn][maxn], int n, int m) {
    memset(free_x, 1, sizeof free_x); memset(x, 0, sizeof x);
    int r = 0, c = 0;
    while (r < n && c < m) {
        int m_r = r;
        rep (i, r + 1, n-1)
            if (fabs(a[i][c]) > fabs(a[m_r][c])) m_r = i;
        if (m_r != r)
            rep (j, c, m)
                swap(a[r][j], a[m_r][j]);
        if (!sgn(a[r][c])) {
            a[r][c] = 0;
            ++c;
            continue;
        }
```

```
26        rep (i, r + 1, n-1)
27           if (a[i][c]) {
28              LD t = a[i][c] / a[r][c];
29              rep (j, c, m + 1) a[i][j] -= a[r][j] * t;
30           }
31        ++r; ++c;
32     }
33     rep (i, r, n-1)
34        if (sgn(a[i][m])) return -1;
35     if (r < m) {
36        per (i, r - 1, 0) {
37           int f_cnt = 0, k = -1;
38           rep (j, 0, m-1)
39              if (sgn(a[i][j]) && free_x[j]) {
40                 ++f_cnt;
41                 k = j;
42              }
43           if(f_cnt > 0) continue;
44           LD s = a[i][m];
45           rep (j, 0, m-1)
46              if (j != k) s -= a[i][j] * x[j];
47           x[k] = s / a[i][k];
48           free_x[k] = 0;
49        }
50        return m - r;
51     }
52      per(i, m - 1, 0) {
53        LD s = a[i][m];
54        rep (j, i + 1, m-1)
55           s -= a[i][j] * x[j];
56        x[i] = s / a[i][i];
57     }
58     return 0;
59  }
```

4.12 BM 线性递推

```
1  namespace linear_seq
2  {
3     #define SZ(x) ((int)x.size())
4     using VI=vector<int>;
5     const int N=10010;
6     ll res[N],base[N],_c[N],_md[N];
7     vector<int> Md;
8     void mul(ll *a,ll *b,int k) {
9        for(int i = 0 ; i < k + k ; ++i)
10       _c[i]=0;
11       for(int i = 0 ; i < k ;++i)
12        if (a[i])
13        for(int j = 0 ;j < k ;++ j)
14           _c[i+j]=(_c[i+j]+a[i]*b[j])%mod;
15       for (int i=k+k-1;i>=k;i--)
```

```
16          if (_c[i])
17          for(int j = 0 ; j<(int ) Md.size() ; ++ j)
18              _c[i-k+Md[j]]=(_c[i-k+Md[j]]-_c[i]*_md[Md[j]])%mod;
19      for(int i =0 ; i< k ; ++i)
20      a[i]=_c[i];
21  }
22  int solve(ll n,VI a,VI b) {
23      ll ans=0,pnt=0;
24      int k=SZ(a);
25      assert( SZ(a) == SZ(b) );
26      for(int i = 0 ;i < k ; ++ i)
27       _md[k-1-i] = -a[i] ; _md[k] = 1 ;
28      Md.clear() ;
29      for(int i =0 ; i < k ; ++ i)
30          if (_md[i]!=0)
31              Md.push_back(i);
32      for(int i = 0; i< k ;++ i)
33       res[i]=base[i]=0;
34      res[0]=1;
35      while ((1ll<<pnt)<=n)
36      pnt++;
37      for (int p=pnt;p>=0;p--) {
38          mul(res,res,k);
39          if ((n>>p)&1) {
40              for (int i=k-1;i>=0;i--) res[i+1]=res[i];res[0]=0;
41              for(int j = 0 ;j < (int)Md.size() ; ++ j)
42               res[ Md[j] ]=(res[ Md[j] ]-res[k]*_md[Md[j]])%mod;
43          }
44      }
45      rep(i,0,k-1) ans=(ans+res[i]*b[i])%mod;
46      if (ans<0) ans+=mod;
47      return ans;
48  }
49  VI BM(VI s) {
50      VI C(1,1),B(1,1);
51      int L=0,m=1,b=1;
52      for(int n= 0 ;n < (int)s.size(); ++ n ) {
53          ll d=0;
54          for(int i =0 ; i < L +1 ;++ i)
55          d=(d+(ll)C[i]*s[n-i])%mod;
56          if (d==0) ++m;
57          else if (2*L<=n) {
58              VI T=C;
59              ll c=mod-d*qpow(b,mod-2)%mod;
60              while (SZ(C)<SZ(B)+m)
61                  C.push_back(0);
62              for(int i =0 ; i < (int)B.size(); ++ i)
63                  C[i+m]=(C[i+m]+c*B[i])%mod;
64              L=n+1-L; B=T; b=d; m=1;
65          } else {
66              ll c=mod-d*qpow(b,mod-2)%mod;
67              while (SZ(C)<SZ(B)+m)
68                  C.push_back(0);
```

```
69              for(int i = 0 ;i <(int) B.size() ; ++ i)
70                  C[i+m]=(C[i+m]+c*B[i])%mod;
71              ++m;
72          }
73      }
74      return C;
75  }
76  ll gao(VI a,ll n) {
77      VI c=BM(a);
78      c.erase(c.begin());
79      for( int i = 0 ; i < (int)c.size( );++i )
80          c[i]=(mod-c[i])%mod;
81      return (ll)solve(n,c,VI(a.begin(),a.begin()+SZ(c)));
82  }
83  };
```

4.13 O(1) 快速乘

```
1  inline ll mul(ll a,ll b,ll mod){
2      return (a*b-(ll)((long double)a/mod*b)*mod+mod)%mod;
3  }
```

5 其他

5.1 vimrc

```
1  set nu sw=4 ts=4 bs=2 ai smd
2  syntax on
3  set cindent
4  set noswapfile
5  set nobackup
6  set mouse =a
7  set showmatch
8  set autowrite
9  imap kj <esc>l
10 nmap <c-a> ggVG"+y
11 imap {<CR> {<CR>}<Esc>O
12 imap {<s-cr> {<CR>
13 map<F6> :call CR()<CR>
14 func! CR()
15 exec "w"
16 exec "!g++ % -std=c++11 -o %<.exe"
17 exec "! %<.exe "
18 endfunc
```