CCU_Guardians Code Book

Tsung

November 22, 2019

- 注意事項

1. 開long long
2. 測印表機
3. initial

- Vimrc

```
set nu
set relativenumber
set autoindent
set smartindent
set cindent
set backspace=2
set confirm
set mouse=a
set tabstop=4
set softtabstop=4
set smarttab
set shiftwidth=4
imap {<CR> {<CR>}<Esc>ko
```

- Header

```
#include<algorithm>
#include<iostream>
#include<cstdio>
#include<cstdlib>
#include<cstring>
#include<map>
#include<cmath>
#include<queue>
#include<stack>
#include<set>
#include<string>
#include<utility>
#include<vector>
#include<cctype>
```

- 圖論
  - BFS

```
void bfs(int s){
    bool visit[N];
    memset(visit,false,sizeof(visit));
    queue<int> q;
    q.push(s);
    visit[s]=true;
    while(!q.empty()){
        int u=q.front();
        q.pop();
        for(int v:graph[u]){
            if(!visit[v]){
                visit[v]=true;
                q.push(v);
            }
        }
    }
}
```

  - DFS

```
bool visit[N];
    void dfs(int u){
    if(visit[u]) return ;
    visit[u]=true;
    for(int v:graph[u]){
        dfs(v);
    }
}
```

– Dijkstra

```
int dis[N];
bool visit[N];
int dijkstra(int s,int t){
    memset(dis,0x3f3f3f3f.sizeof(dis));
    memset(visit,false,sizeof(visit));
    priority_queue<pii,vii,greater<pii> > pq;
    dis[s]=0;
    pq.push({dis[s],s});

    while(!pq.empty()){
        int u=pq.top().second; pq.pop();
        if(visit[u]) continue;
        visit[u]=true;
        for(auto e:g[u]){
            if(dis[u]+e.w<dis[e.v]){
                dis[e.v]=dis[u]+e.w;
                pq.push({dis[e.v],v});
            }
        }
    }

    return dis[t];
}
```

– Bellman-Ford

```
int dis[N];
bool bellman_ford(int s,int t){
    memset(dis,0x3f3f3f3f,sizeof(dis));
    dis[s]=0;
    for(int i=0;i<n;i++){
        for(int j=0;j<E;j++){
            Edge &e=edge[j];
            if(dis[e.to]>dis[e.from]+e.w){
                dis[e.to]=dis[e.from]+e.w;
                if(i==n-1) return true;
            }
        }
    }
    return false;
}
```

– Floyd-Warshall

```
int adj[N][N];
void Floyd_Warshall(){
    memset(adj,0x3f3f3f3f,sizeof(adj));
    for(int i=0;i<n;i++) adj[i][i]=0;
    // input the edge
    for(int k=0;k<n;k++){
        for(int i=0;i<n;i++){
            for(int j=0;j<n;j++){
                adj[i][j]=min(adj[i][j],adj[i][k]+adj[k][j
                    ]);
            }
        }
    }
}
```

– MST(Kruskal)

```
// include "Union Find - Disjoint Set"
int kruskal(){
    // m is the number of edge
    UFDS uf;
    uf.init();
    int ans=0;
    sort(edge,edge+m); // sorting by weighted.
    for(int i=0;i<m;i++){
        if(uf.find(edge[i].a)!=uf.find(edge[i].b)){
            uf.Union(edge[i].a,edge[i].b);
            ans+=edge[i].w;
        }
    }
    return ans;
}
```

– MST(Prim)

```
int prim(){
    int ans=0;
    bool used[n]={false};
    priority_queue<pii,vector<pii>,greater<pii> > pq;
    pq.push(pii(0,0));

    while(!pq.empty()){
        pii vur = pq.top();
        pq.pop();

        int u=cur.second;
        if(used[u]) continue;
        ans+=cur.first;
        used[u]=true;
        for(int i=0;i<(int)g[u].size();i++){
            int v=g[u][i].first,w=g[u][i].second;
            if(used[v]==false) pq.push(pii(w,v));
        }
    }
    return ans;
}
```

– Bipartite Matching

```
int match[MAX_N+4];
bool used[MAX_N+4];
vector<int> edge[MAX_N+4];
int n,x,y,m;
bool dfs(int v){
    for(int u:edge[v]){
        if(used[u]) continue;
        used[u]=true;
        int temp=match[u];
        if(temp<0 || dfs(temp)){
            match[u]=v;
            return true;
        }
    }
    return false;
}
int bipartite_matching(){
    int ans=0;
    memset(match,-1,sizeof(match));
    for(int i=0;i<x;i++){
        memset(used,false,sizeof(used));
        if(dfs(i)) ans++;
    }
    return ans;
}
```

– Max Flow

```
struct Edge{
    int to,cap,rev;
    Edge(int a,int b,int c){
        to=a;
        cap=b;
        rev=c;
    }
};
const int INF=0x3f3f3f3f;
const int MAX_V=20000+10;
vector<vector<edge> > g(MAX_V);
int level[MAX_V];
int iter[MAX_V];

inline void add_edge(int u,int v,int cap){
    g[u].push_back((Edge){v,cap,(int)g[v].size()});
    g[v].push_back((Edge){u,0,(int)g[u].size()-1});
}
void bfs(int s){
    memset(level,-1,sizeof(level));
    queue<int> q;
    level[s]=0;
    q.push(s);
    while(!q.empty()){
        int v=q.front();
        q.pop();
        for(int i=0;i<(int)g[v].size();i++){
            const Edge& e = g[v][i];
            if(e.cap>0 && level[e.to]<0){
                level[e.to]=level[v]+1;
                q.push(e.to);
            }
        }
    }
}
int dfs(int v,int t,int f){
    if(v==t) return f;
    for(int& i=iter[v];i<(int)g[v].size();i++){
        Edge& e=g[v][i];
        if(e.cap>0 && level[v]<level[e.to]){
            int d=dfs(e.to,t,min(f,e.cap));
            if(d>0){
                e.cap-=d;
                g[e.to][e.rev].cap+=d;
                return d;
            }
        }
    }
    return 0;
}
int max_flow(int s,int t){
    int flow=0;
    for(;;){
        bfs(s);
        if(level[t]<0) return flow;
        memset(iter,0,sizeof(iter));
        int f;
        while((f=dfs(s,t,INF))>0){
            flow+=f;
        }
    }
}
```

– TSP(DP)

```
const MAX_N=10000;
const int INF=0x3f3f3f3f;
int N,dp[1<<MAX_N][MAX_N];
int TSP(){
    for(int S=0;S<(1<<N);S++) fill(dp[S],dp[S]+N,INF);
    dp[(1<<N)-1][0]=0;
    for(int S=(1<<N)-2;S>=0;S--){
        for(int v=0;v<N;v++){
            for(int u=0;u<N;u++){
                if(!((S>>u)&1)){
                    dp[S][v]=min(dp[S][v],dp[S | (1<<u)][u
                        ]+d[v][u]);
                }
            }
        }
    }
    return dp[0][0];
}
```

– Max Clique

```
const int MAX_N=100;
int len[MAX_N+4];
int adj[MAX_N+4][MAX_N+4],vis[MAX_N+4][MAX_N+4];
int List[MAX_N+4][MAX_N+4];
int mc[MAX_N+4];
void dfs(int num){
    if(len[num]==0){
        if(num>ans){
            ans=num;
            found=true;
        }
        return ;
    }
    for(int k=0;k<len[num] && !found;k++){
        if(num+len[num]-k<=ans) break;
        int i=List[num][k];
        if(num+mc[i]<=ans) break;
        int j;
        for(int j=k+1,len[num+1]=0;j<len[num];j++){
            if(adj[i][List[num][j]]){
                List[num+1][len[num+1]++]=List[num][j];
            }
        }
        dfs(num+1);
    }
}
void max_clique(){
    mc[n]=ans=1;
    for(int i=n-1;i;i--){
        found=false;
        len[1]=0;
        for(int j=i+1;j<=n;j++){
            if(adj[i][j]) List[1][len[1]++]=j;
        }
        dfs(1);
        mc[i]=ans;
    }
}
```

– 找關節點

```
const int MAX_V=10010;
int V,E;
vector<int> g[MAX_V];
int dfn[MAX_V],low[MAX_V],tot;
vector<pii> ans;

void dfs(int x,int fa){
    dfn[x]=low[x]=++tot;
    for(int i=0;i<g[x].size();i++){
        int v=g[x][i];
        if(v==fa) continue;
        if(!dfn[v]){
            dfs(v,x);
            low[x]=min(low[x],low[v]);
            if(low[v]>dfn[x]){
                ans.push_back({x,v});
            }
        }
        else{
            low[x]=min(low[x],dfn[v]);
        }
    }
}

void bcc_tarjan(){
    for(int i=0;i<V;i++){
        if(!dfn[i]) dfs(i,-1);
    }
}
```

– Minimum Mean Cycle

```
struct edge{
    int to,w;
    edge(int a=0,int b=0):
    to(a),w(b){}
};
vector<edge> g[1024];
int  dp[1024][1024], inq[1024][1024];
const int MAXE=1000 ;
void solve(int source,int N){
    for(int i=0;i<N;i++)
    for  (int j=0;j<=MAXE;j++)
        dp[i][j]=0x3f3f3f3f;
    queue<int> X,E;
    int u,v,w,e;
    dp[source][0]=0 ;
    X.push(source),E.push(0);
    while(!X.empty()) {
        u=X.front(),X.pop();
        e=E.front(),E.pop();
        inq[u][e]= 0 ;
        if(e==MAXE) continue ;
        for(int i=0;i<g[u].size();i++){
            v=g[u][i].to,w=g[u][i].w;
            if(dp[v][e+1]>dp[u][e]+w) {
                dp[v][e+ 1]=dp[u][e]+w;
                if(!inq[v][e+1]){
                    inq[v][e+1]=1;
                    X.push(v),E.push(e+1);
                }
            }
        }
    }
}
int  main ()  {
    int N,M,S,Q;
    int x,y,w;
    while(scanf("%d %d %d",&N,&M,&S)==3){
        for (int i=0;i<N;i++)
            g[i].clear();
        for(int i=0;i<M;i++) {
            scanf ("%d %d %d",&x,&y,&w);
            g[x].push_back(edge(y,w));
        }
        solve(S,N);
        scanf("%d" ,&Q);
        while(Q--){
            scanf("%d",&x);
            double ret= 1e+30 ;
            int e=-1 ;
            if(x==S)
                ret=0,e=0 ;
            else{
                for(int i=1;i<=MAXE;i++){
                    if(dp[x][i]!=0x3f3f3f3f){
                        if((double)dp[x][i]/i<ret){
                            ret=(double)dp[x][i]/i;
                            e=i;
                        }
                    }
                }
            }
            if  (e==-1)
                puts("No Path");
            else
                printf ("%.4lf %d\n",ret,e);
        }
        puts("");
    }
    return  0 ;
}
```

– Minimum Cut Max flow

```
struct Point{
    int x,y,z;
};
int n,m,ss,tt;
queue<int>q;
int dis[N],minv[N];
bool vis[N];
struct Edge{int to;int value;int cost;int next;}e[M<<1];
struct Pre{int id;int node;}pre[M<<1];
int head[N],cnt=-1;
Point a[510];
void add(int from,int to,int value,int cost){
    cnt++; e[cnt].to=to; e[cnt].value=value;
    e[cnt].cost=cost;
    e[cnt].next=head[from]; head[from]=cnt;
}
bool spfa(int s,int t){
    q=queue<int>();
    memset(dis,0x3f,sizeof(dis));
    memset(vis,0,sizeof(vis));
    memset(pre,-1,sizeof(pre));
    memset(minv,0x3f,sizeof(minv));
    dis[s]=0;
    vis[s]=1;
    q.push(s);
    while(!q.empty()){
        int x=q.front();
        q.pop();
        vis[x]=0;
        for(int i=head[x];i>-1;i=e[i].next){
            int now=e[i].to;
            if(dis[now]>dis[x]+e[i].cost&&e[i].value){
                dis[now]=dis[x]+e[i].cost;
                minv[now]=min(minv[x],e[i].value);
                pre[now].id=i;
                pre[now].node=x;
                if(!vis[now]){
                    vis[now]=1;
                    q.push(now);
                }
            }
        }
    }
    return dis[t]!=INF;
}
void MCMF(int s,int t,int &maxflow,int &mincost){
    while(spfa(s,t)){
        for(int i=t;i!=s;i=pre[i].node){
            e[pre[i].id].value-=minv[t];
            e[pre[i].id^1].value+=minv[t];
        }
        maxflow+=minv[t];
        mincost+=minv[t]*dis[t];
    }
}
int disss(Point a,Point b){
    return (a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y)+(a.z-b.
       z)*(a.z-b.z);
}
int main(){
    memset(head,-1,sizeof(head));
    scanf("%d",&n);
    for(int i=0;i<2*n;i++){
        scanf("%d%d%d",&a[i].x,&a[i].y,&a[i].z);
    }
    for(int i=0;i<n;i++)
        for(int j=n;j<2*n;j++){
            int w=disss(a[i],a[j]);
            add(i,j,1,w);
            add(j,i,0,-w);
        }
    ss=2*n;
    tt=2*n+1;
    for(int i=0;i<n;i++){
        add(ss,i,0x3f3f3f3f,0);
        add(i+n,tt,0x3f3f3f3f,0);
    }
    int mf=0,mc=0;
    MCMF(ss,tt,mf,mc);
    printf("%d %d\n",mf,mc);
    return 0;
}
```

– 找橋

```
const int MAX_V=10010;
int V,E;
vector<int> g[MAX_V];
int dfn[MAX_V],low[MAX_V],tot,bccid[MAX_V],bcc_cnt;
bool cut[MAX_V];
stack<int> S;

void dfs(int x,int fa){
    int child=0;
    dfn[x]=low[x]=++tot;
    S.push(x);
    for(int i=0;i<(int)g[x].size();i++){
        int v=g[x][i];
        if(!dfn[v]){
            dfs(v,x);
            child++;
            low[x]=min(low[x],low[v]);
            if(low[v]>=dfn[x]){
                cut[x]=true;
                while(1){
                    int u=S.top(); S.pop();
                    bccid[x]=bcc_cnt;
                    bcc_cnt++;
                }
            }
        }
        else if(dfn[v]<dfn[x] && v!=fa){
            low[x]=min(low[x],dfn[v]);
        }
    }
    if(fa==-1 && child<2) cut[x]=false;
}
void bcc_tarjan(){
    bcc_cnt=0;
    for(int i=0;i<V;i++){
        if(!dfn[i]) dfs(i,-1);
    }
}
```

• 樹

– Distance Tree

```
int parent[N],childnum[N],value[N],nodenum[N];
int distance_tree(){
    queue<int> inque;
    memset(parent,-1,sizeof(parent));
    memset(childinum,0,sizeof(childnum));
    for(int i=2;i<=n;i++){
        scanf("%d",&parent[i]);
        childnum[parent[i]]++;
    }
    for(int i=2;i<=n;i++){
        scanf("%d",&value[i]);
    }
    for(int i=2;i<=n;i++){
        if(childnum[i]==1) inque.push(i);
    }
    for(int i=1;i<=n;i++) nodenum[i]=1;
        while(!inque.empty()){
            int u=inque.front();
            inque.pop();
            nodenum[parent[u]]+=nodenum[u];
            childnum[parent[u]]--;
            if(childnum[parent[u]]==0){
                inque.push(parent[u]);
            }
        }
        int ans=0;
        for(int i=2;i<=n;i++){
            ans+=(nodenum[1]-nodenum[i])*nodenum[i]*value[
               i];
        }
    return ans*2;
}
```

– LCA 倍增法

```
const int MAX_N=10000,MAX_LOG_N=14;
int N,root,dep[MAX_N],par[MAX_LOG_N][MAX_N];
vector<int> child[MAX_N];
void dfs(int u,int p,int d){
    dep[u]=d;
    for(int i=0;i<(int)child.size();i++){
        int v=child[u][i];
        if(v!=p) dfs(v,u,d+1);
    }
}
void build(){
    dfs(root,-1,0);
    for(int i=0;i+1<MAX_LOG_N;i++){
        for(int u=0;u<N;u++){
            if(par[i][u]==-1) par[i+1][u]=-1;
            else par[i+1][u]=par[i][par[i][u]];
        }
    }
}
int lca(int u,int v){
    if(dep[u]>dep[v]) swap(u,v);
    int diff=dep[v]-dep[u];
    for(int i=0;i<MAX_LOG_N;i++){
        if(diff & (1<<i)) v=par[i][v];
    }
    if(u==v) return u;
    for(int i=MAX_LOG_N-1;i>=0;i--){
        if(par[i][u]!=par[i][v]){
            u=par[i][u];
            v=par[i][v];
        }
    }
    return par[0][u];
}
```

– LCA Tarjan(搭配併查集)

```
struct Node{
    int v,ind;
};
int n,q;
int parent[MAX_N+4];
vector<int> edge[MAX_N+4];
vector<Node> ques[MAX_N+4];
bool visit[MAX_N+4];
UFDS uf;
int ans[MAX_N+4];
int LCA[MAX_N+4][MAX_N+4];
void tarjan(int u){
    if(visit[u]) return ;
    visit[u]=true;
    for(int v:edge[u]){
        tarjan(v);
        uf.Union(u,v);
    }
    for(Node v:ques[u]){
        if(visit[v.v]){
            ans[v.ind]=uf.Find(v.v);
        }
    }
}
int main(){
    int root;
    scanf("%d%d%d",&n,&q,&root);
    //make graph
    for(int i=0;i<q;i++){
        int u,v;
        scanf("%d%d",&u,&v);
        ques[u].push_back({v,i});
        ques[v].push_back({u,i});
    }
    uf.init(n,-1);
    tarjan(root);
    for(int i=0;i<q;i++){
        printf("%d ",ans[i]);
    }
    printf("\n");
}
```

– Tree Diameter(Weighted)

```
struct Edge{
    int v,w;
};
vector<Edge> edge[10010];
int dist[10010];
int bfs(int s){
    queue<int> q;
    memset(dist,-1,sizeof(dist));
    dist[s]=0;
    q.push(s);
    int maxi=0,ind=s;
    while(!q.empty()){
        int u=q.front(); q.pop();
        for(int i=0;i<(int)edge[u].size();i++){
            Edge e=edge[u][i];
            if(dist[e.v]==-1){
                dist[e.v]=dist[u]+e.w;
                if(dist[e.v]>maxi){
                    maxi=dist[e.v];
                    ind=e.v;
                }
                q.push(e.v);
            }
        }
    }
    return ind;
}
```

– Tree Center(Unweighted)

```
int diameter=0,radius[N],deg[N];
int findRadius(){
    queue<int> q;
    for(int i=0;i<n;i++){
        if(deg[i]==1) q.push(i);
    }
    int mx=0;
    while(!q.empty()){
        int u=q.front(); q.pop();
        for(int v:g[u]){
            deg[v]--;
            if(deg[v]==1){
                q.push(v);
                radius[v]=radius[u]+1;
                mx=max(mx,radius[v]);
            }
        }
    }
    int cnt=0;
    for(int i=0;i<n;i++){
        if(radius[i]==mx) cnt++;
    }
    diameter=max(diameter,mx*2+(cnt==2));
    return mx+(cnt==2);
}
```

- DP
  - LCS

```
int dp[504][504];
int LCS(string s,string t){
    memset(dp,0,sizeof(dp));
    for(int i=0;i<(int)s.size();i++){
        for(int j=0;j<(int)t.size();j++){
            if(s[i]==t[j]) dp[i+1][j+1]=dp[i][j]+1;
            else dp[i+1][j+1]=max(dp[i+1][j],dp[i][j+1]);
        }
    }
    return dp[s.size()][t.size()];
}
```

  - LCS Alignment

```
int dp[504][504];
int LCS(string s,string t,int wrong,int connect,int gap){
    for(int i=0;i<=s.size();i++){
        dp[i][0]=gap*i;
    }
    for(int i=0;i<=t.size();i++){
        dp[0][i]=gap*i;
    }
    for(int i=0;i<(int)s.size();i++){
        for(int j=0;j<(int)t.size();j++){
            if(s[i]==t[j]){
                dp[i+1][j+1]=dp[i][j]+connect;
            }
            else{
                dp[i+1][j+1]=max(dp[i+1][j]+gap,dp[i][j
                    +1]+gap);
                dp[i+1][j+1]=max(dp[i+1][j+1],dp[i][j]+
                    connect);
            }
        }
    }
    return dp[s.size()][t.size()];
}
```

  - LIS

```
vector<int> LIS_list;
int a[N];
int LIS(){
    LIS_list.push_back(a[0]);
    for(int i=1;i<n;i++){
        if(a[i]>LIS_list.back()) LIS_list.push_back(a[i]);
        else{
            *lower_bound(LIS_list.begin(),LIS_list.end(),a
                [i])=a[i];
        }
    }
    return (int)LIS_list.size();
}
```

  - WLIS

```
int solve(){
    map<int,int> mymp;
    mymp[-1]=0;
    mymp[a[0]]=w[0];
    for(int i=1;i<n;i++){
        auto ret=mymp.lower_bound(a[i]);
        auto ret2=ret;
        ret--;
        if(ret2==mymp.end()) mymp[a[i]]=ret->second+w[i];
        else if(ret->second+w[i]>ret2->second){
            while(ret->second+w[i]>ret2->second){
                mymp.erase(ret2->first);
                ret2=mymp.lower_bound(a[i]);
                if(ret2==mymp.end()) break;
            }
            mymp[a[i]]=ret->second+w[i];
        }
        else if(mymp.find(a[i])==mymp.end()){
            mymp[a[i]]=ret->second+w[i];
        }
    }
    return mymp.rbegin()->second;
}
```

– Cutting Stick

```
int l,n;
int dp[2000][2000];
int a[2000];
int cut(int l,int r){
    int cost=0x3f3f3f3f;
    if(dp[l][r]) return dp[l][r];
    if(r-l==2) return dp[l][r]=a[r]-a[l];
    if(r-l<2) return 0;
    for(int i=l+1;i<r;i++){
        cost=min(cost,a[r]-a[l]+cut(l,i)+cut(i,r));
    }
    return dp[l][r]=cost;
}
int cutting_stick(){
    return cut(0,n+1);
}
```

– HyberCube

```
int nbit,n;
int weight[200200];
int dp[200200]; //initial to -1
int hybercube(int goal){
    if(goal==0) return dp[0]=weight[0];
    if(dp[goal]>=0) return dp[goal];
    int maxi=0;
    for(int i=0;i<nbit;i++){
        if(goal & (1<<i)){
            int k=hybercube(goal^(1<<i));
            maxi=max(k,maxi);
        }
    }
    dp[goal]=maxi+weight[goal];
    return dp[goal];
}
```

– 01 背包問題

```
struct Pack{
    int v,w;
};
int n,w;
int dp[1000004];
pack a[505];
int knapsack_01(){
    for(int i=1;i<=n;i++){
        for(int j=w;j>=0;j--){
            if(a[i].w<=j){
                dp[j]=max(dp[j],dp[j-a[i].w]+a[i].v);
            }
        }
    }
    return dp[w];
}
```

– 無限背包問題

```
const int N=100,W=100000;
int cost[N],weight[N];
int c[W+1];

int unbounded_knapsack(int n,int w){
    memset(c,0,sizeof(c));
    for(int i=0;i<n;i++){
        for(int j=weight[i];j<=w;j++){
            c[j]=max(c[j],c[j-weight[i]]+cost[i]);
        }
    }
    return c[w];
}
```

– 有限背包問題

```
const int N=100,W=100000;
int cost[N],weight[N],number[N];
int c[W+1];

int unbounded_knapsack(int n,int w){
    memset(c,0,sizeof(c));
    for(int i=0;i<n;i++){
        int num=min(number[i],w/weight[i]);
        for(int k=1;num>0;k*=2){
            if(k<num) k=num;
            num-=k;
            for(int j=w;j>=wieght[i]*k;j--){
                c[j]=max(c[j],c[j-weight[i]*k]+cost[i]*k);
            }
        }
    }
    return c[w];
}
```

– Sum of Subset

```
int a[200200];
int n,m;
bool rec[200200];
void sum_of_subset(){
    int total=0;
    rec[0]=true;
    for(int i=0;i<n;i++){
        for(int j=total;j>=0;j--){
            if(rec[j]) rec[j+a[i]]=true;
        }
    }
    total+=a[i];
}
```

– Maximum Submatrix

```
const int A,B,C; //dimension
int max1D(int a[]){
    int ans=-1e9;
    int s=0;
    for(int i=0;i<C;i++){
        if(s>=0) s+=a[i];
        else s=a[i];
        ans=max(ans,s);
    }
    return ans;
}
int max2D(int a[][C]){
    int ans=-1e9;
    int s[C];
    for(int i=0;i<B;i++){
        memset(s,0,sizeof(s));
        for(int j=i;j<B;j++){
            for(int k=0;k<C;k++){
                s[k]+=a[j][k];
            }
            ans=max(ans,max1D(s));
        }
    }
    return ans;
}
int max3D(int a[][B][C]){
    int ans=-1e9;
    int s[B][C];
    for(int i=0;i<A;i++){
        memset(s,0,sizeof(s));
        for(int j=i;j<A;j++){
            for(int k=0;k<B;k++){
                for(int l=0;k<C;l++){
                    s[k][l]+=a[j][k][l];
                }
            }
            ans=max(ans,max2D(s));
        }
    }
    return ans;
}
```

- Interval Problem
  - Unweighted Independent Interval

```
struct inter{
    int L,R
}
int n;
inter a[200200];
int unweighted_independent_interval(){
    sort(a,a+n,cmp) //sorting by right point;
    int ans=1;
    int temp=a[0].R;
    for(int i=1;i<n;i++){
        if(a[i].L>=temp){
            temp=a[i].R;
            ans++;
        }
    }
    return ans;
}
```

  - weighted Independent Interval

```
inter a[100100];
int n,dp[100100];
int binary(int x){
    int ll=0,lr=x,mid;
    while(lr-ll>1){
        mid=(lr+ll)/2;
        if(a[mid].right>a[x].left) lr=mid;
        else ll=mid;
    }
    return dp[ll];
}
int independent_interval(){
    sort(a+1,a+1+n,cmp); //sorting by right point;
    dp[0]=0; dp[1]=a[1].w;
    for(int i=2;i<=n;i++){
        dp[i]=max(dp[i-1],a[i].w+binary(i));
    }
    return dp[n];
}
```

  - Shortest Path on interval

```
int dis[100100];
Inter inter[100100];
void SPOI(){
    sort(inter,inter+n,cmp); //sortint by left point
    inter[n].ll=inter[n-1].lr;
    inter[n]=0x3f3f3f3f;
    inter[n].w=0;
    n++;
    memset(dis,0x3f3f3f3f,sizeof(dis));
    priority_queue<pii,vector<pii>,greater<pii> > pq;
    pq.push({0,0});
    for(int i=0;i<n;i++){
        while(!pq.empty() && pq.top().second<inter[i].ll){
            pq.pop();
        }
        dis[i]=pq.top().first;
        pq.push({dis[i]+inter[i].w,inter[i].lr});
    }
    return dis[n-1];
}
```

- 數學
  - Fast Power

```
lint fp(lint x,lint y,lint p){
    lint t=1;
    while(y){
        if(y&1){
            t*=x;
            t%=p;
        }
        x*=x;
        x%=p;
        y>>=1;
    }
    return t;
}
```

  - Combination

```
lint binomialCoeff(lint n,lint k){
    lint res=1;
    if(k>n-k) k=n-k;
    for(int i=0;i<k;i++){
        res*=(n-i);
        res/=(i+1);
    }
    return res;
}
```

  - Matrix Determinant

```
typedef long long lint;
typedef vector<lint> vec;
typedef vector<vec> mat;
lint determinant(mat m){
    const int n=m.size();
    lint det=1;
    for(int i=0;i<n;i++){
        for(int j=i+1;j<n;j++){
            int a=i,b=j;
            while(m[b][i]){
                lint q=m[a][i]/m[b][i];
                for(int k=0;k<n;k++){
                    m[a][k]=m[a][k]-m[b][k]*q;
                }
                swap(a,b);
            }
            if(a!=i){
                swap(m[i],m[j]);
                det=-det;
            }
        }
        if(m[i][i]==0) return 0;
        else det*=m[i][i];
    }
    return det;
}
```

  - Pascal

```
#define N 210
lint c[N][N];
void Combination(){
    for(lint i=0;i<N;i++){
        c[i][0]=1; c[i][i]=1;
    }
    for(lint i=2;i<N;i++){
        for(lint j=1;j<=i;j++){
            c[i][j]=(c[i-1][j]+c[i-1][j-1])%m;
        }
    }
}
```

– Prime Table

```
bool p[10010000];
vector<int> prime;
void prime_table(){
    for(int i=2;i<10010000;i++){
        if(!p[i]){
            for(int j=i+i;j<10010000;j+=i){
                p[j]=true;
            }
        }
    }
    for(int i=2;i<=10010000;i++){
        if(!p[i]) prime.push_back(i);
    }
}
```

– Exgcd

```
void Exgcd(lint a,lint b,lint &d,lint &x,lint &y){
    if(b==0){
        x=1;
        y=0;
        d=a;
        return ;
    }
    Exgcd(b,a%b,d,y,x);
    y-=a/b*x;
    return ;
}
```

– Matrix Mutiplication

```
for(int i=0;i<a;i++){
    for(int j=0;j<d;j++){
        C[i][j]=0;
        for(int k=0;k<c;k++){
            C[i][j]+=A[i][k]*B[k][j];
        }
    }
}
```

– 中國剩餘定理

```
int a[N],m[N],M;
int CRT(){
    M=1;
    for(int i=0;i<n;i++){
        M*=m[i];
    }
    int ans=0,x,y,d;
    for(int i=0;i<n;i++){
        Exgcd(M/m[i],m[i],d,x,y);
        while(x<0) x+=m[i];
        ans+=a[i]*x*(M/m[i]);
        ans%=M;
    }
    return ans;
}
```

– 大數分解 (含大質數)

```
lint n,ans;
lint fm(lint x,lint y,lint p){
    lint t=0;
    x%=p;
    y%=p;
    while(y){
        if(y&1){
            t+=x;
            t%=p;
        }
        x+=x;
        x%=p;
        y>>=1;
    }
    return t;
}
bool isprime(lint n){
    if(n==2ll) return true;
    if(n%2ll==0ll || n==1ll) return false;
    lint temp=n-1;
    lint s=0;
    while(temp%2ll==0ll){
        s++;
        temp/=2ll;
    }
    lint d=temp;
    for(int cnt=0;cnt<30;cnt++){
        lint a=rand()%(n-1)+1;
        bool flag=false;
        lint res0=fp(a,d,n);
        for(int i=0;i<s;i++){
            lint res=fp(a,((1<<i)*d),n);
            if(res+1==n || res0==1ll){
                flag=true;
                break;
            }
        }
        if(!flag){
            return false;
        }
    }
    return true;
}
lint f(lint x,lint c,lint n){
    return (fm(x,x,n)+c)%n;
}
lint pollard_rho(lint x,lint c){
    lint i=1,k=2;
    lint x0=rand()%x;
    lint y=x0;
    while(1){
        i++;
        x0=f(x0,c,x);
        lint d=gcd((y-x0)>0?(y-x0):(x0-y),x);
        if(d!=1 && d!=x) return d;
        if(y==x0) return x;
        if(i==k){
            y=x0;
            k+=k;
        }
    }
}
void decom(lint n){
    if(isprime(n)){
    ans=min(ans,n);
    return ;
}
    lint p=n;
    while(p>=n){
        p=pollard_rho(p,rand()%(n-1)+1);
    }
    decom(p);
    decom(n/p);
}
```

– 原根個數 (歐拉函數)

```
int n;
int euler(int n){
    int res=n,a=n;
    for(int i=2;i*i<=a;i++){
        if(a%i==0){
            res=res/i*(i-1);
            while(a%i==0) a/=i;
        }
    }
    if(a>1) res=res/a*(a-1);
    return res;
}
int main(){
    while(scanf("%d",&n)!=EOF){
        printf("%d\n",euler(euler(n)));
    }
}
```

– baby giant step (離散對數)

```
//unorder_set is better.
map<lint,int> mp;
lint bsgs(lint a,lint b,lint p){
    int i;
    if(p%a==0) return -1;
    lint m=ceil(sqrt(p));
    lint l=b;
    mp[l]=1;
    for(int i=1;i<=m;i++){
        l*=a;
        l%=p;
        mp[l]=i+1;
    }
    l=fp(a,m,p);
    lint r=l;
    for(int i=1;i<=m;i++){
        if(mp[r]){
            lint ans=i*m-mp[r]+1;
            mp.clear();
            return ans;
        }
        r*=l;
        r%=p;
    }
    mp.clear();
    return -1;
}
int main(){
    lint p,b,n;
    while(scanf("%lld%lld%lld",&p,&b,&n)!=EOF){
        //b^x==n (mod p)
        lint ans=bsgs(b,n,p);
        if(ans==-1) printf("no solution\n");
        else printf("%lld\n",ans);
    }
return 0;
}
```

- 資料結構
  - 線段樹

```
struct SEG{
    lint seg[400400],lazy[400400],dflt;
    void init(lint n,lint val){
        dflt=val;
        memset(seg,0,sizeof(seg));
        memset(lazy,0,sizeof(lazy));
    }
    void gather(lint root,lint l,lint r){
        seg[root]=seg[root*2]+seg[root*2+1];
    }
    void push(lint root,lint l,lint r){
        if(l==r) return ;
        if(lazy[root]!=dflt){
            lint m=(r+l)/2;
            seg[root*2]+=(m-l+1)*lazy[root];
            seg[root*2+1]+=(r-m)*lazy[root];
            lazy[root*2]+=lazy[root];
            lazy[root*2+1]+=lazy[root];
            lazy[root]=dflt;
        }
    }
    void build(lint root,lint l,lint r,lint arr[]){
        if(r==l){
            seg[root]=arr[l];
            lazy[root]=dflt;
            return ;
        }
        lint m=(r+l)/2;
        build(root*2,l,m,arr);
        build(root*2+1,m+1,r,arr);
        gather(root,l,r);
    }
    lint query(lint a,lint b,lint root,lint l,lint r){
        if(l>b || r<a) return 0;
        if(l>=a && r<=b){
            return seg[root];
        }
        lint m=(l+r)/2;
        push(root,l,r);
        lint res1=query(a,b,root*2,l,m);
        lint res2=query(a,b,root*2+1,m+1,r);
        gather(root,l,r);
        return res1+res2;
    }
    void update(lint a,lint b,lint value,lint root,lint l,
        lint r){
        if(l>b || r<a) return ;
        if(l>=a && r<=b){
        seg[root]+=(r-l+1)*value;
        lazy[root]+=value;
        return ;
    }
    lint m=(r+l)/2;
    push(root,l,r);
    update(a,b,value,root*2,l,m);
    update(a,b,value,root*2+1,m+1,r);
    gather(root,l,r);
    }
};
lint n,q;
lint a[100100];
int main(){
    scanf("%lld%lld",&n,&q);
    for(int i=1;i<=n;i++) scanf("%lld",&a[i]);
    SEG tree(n,0ll);
    tree.build(1ll,1ll,n,a);
    char ch;
    while(q--){
        scanf(" %c",&ch);
        lint x,y;
        scanf("%lld%lld",&x,&y);
        if(ch=='Q'){
            printf("%lld\n",tree.query(x,y,1ll,1ll,n));
        }
        else{
            lint val;
            scanf("%lld",&val);
            tree.update(x,y,val,1ll,1ll,n);
        }
    }
}
```

– 併查集

```
struct UFDS{
    int parent[N];
    void init(){
        memset(parent,-1,sizeof(parent));
    }
    int find(int x){
        return parent[x]<0?x:(parent[x]=find(parent[x]));
    }
    void Union(int x,int y){
        x=find(x);
        y=find(y);
        if(x==y) return ;
        if(parent[x]>parent[y]) swap(x,y);
        parent[x]+=parent[y];
        parent[y]=x;
    }
};
```

• 其他

– 有權重八皇后問題

```
int weight[N][N];
int maxi,mini;
int column[N],slash[N],backslash[N+N];
void backtrack(int column[],int slash[],int backslash[],
    int queens[],int i,int weight){
    if(i>=n){
        maxi=max(maxi,weight);
        mini=min(mini,weight);
    }
    else{
        for(int j=0;j<n;j++){
            if(isvisitable(i,j,column,slash,backslash)){
                queens[i]=j;
                column[j]=slash[i+j]=backslash[i-j+n]=1;
                backtrack(column,slash,backslash,queens,i
                    +1,weight+weighted[i][j]);
                cloumn[j]=slash[i+j]=backslash[i-j+n]=0;
            }
        }
    }
}
bool isvisitable(int i,int j,int column[],int slash[],int
    backslash[]){
    return !(column[j] || slash[i+j] || backslash[i-j+n]);
}
int queen_ans(){
    maxi=0;
    mini=0x3f3f3f3f;
    backtrack(column,slash,backslash,queens,0,0);
}
```

– 河內塔

```
void tos(int n,char b,char a,char c){
    if(n==1){
        printf("Move sheet from %c to %c\n",a,c);
    }
    else{
        tos(n-1,c,a,b);
        printf("Move sheet from %c to %c\n",a,c);
        tos(n-1,a,b,c);
    }
    return;
}
int main(){
    int n;
    while(scanf("%d",&n)!=EOF){
        int step=1,i;
        for(i=0;i<n;i++)
            step*=2;
        step--;
        printf("%d\n",step);
        tos(n,'B','A','C');
    }
    return 0;
}
```

– 二分搜

```
int binary(int x){
    int ll=0,lr=n,mid;
    while(lr-ll>1){
        mid=(lr-ll)/2+ll;
        if(mid>x) lr=x;
        else ll=x;
    }
    return lr;
}
```

– First Come First Serve

```
int FCFS(){
    int ll=0,lr=n,mid;
    while(lr-ll>1){
        bool flag=true;
        mid=(lr-ll)/2+ll;
        priority_queue<int,vector<int>,greater<int> > pq;
        for(int i=0;i<mid;i++){
            pq.push(0);
        }
        for(int i=0;i<n;i++){
            int k=pq.top(); pq.pop();
            if(k+a[i]>m){
                flag=false;
                break;
            }
            pq.push(k+a[i]);
        }
        if(flag) lr=mid;
        else ll=mid;
    }
    return lr;
}
```

– KMP

```
struct KMP_solution{
    vector<int> failure;
    void getfailure(string& needle){
        failure.assign(needle.size(),-1);
        for(int j=1;j<needle.size();j++){
            int i=failure[j-1];
            while((needle[j]!=needle[i+1]) && (i>=0)){
                i=failure[i];
            }
            if(needle[j]==needle[i+1]){
                failure[j]=i+1;
            }
        }
    }
    int KMP(string& haystack,string& needle){
        int i=0,j=0;
        while(i<haystack.size() && j<needle.size()){
            if(haystack[i]==needle[j]){
                i++;j++;
            }
            else{
                if(j==0) i++;
                else j=failure[j-1]+1
            }
        }
        if(j<needle.size()) return -1;
        else return i-needle.size();
    }
    int strStr(string haystack,string needle){
        getfailure(needle);
        return KMP(haystack,needle);
    }
};
```

– Baby Sitter

```
inter a[200200];
int n;
int baby_sitter(){
    sort(a,a+n,cmp); //sorting by left point
    int temp=-1,maxi=-1,ans=0;
    for(int i=0;i<n;i++){
        if(a[i].L<=temp+1){
            if(a[i].R>maxi) maxi=a[i].R;
        }
        else{
            temp=maxi;
            maxi=a[i].R;
            ans++;
        }
        if(maxi==199) break;
    }
    return ans;
}
```

– RMQ 倍增

```
int a[100100][20],n,q;
void rmq_st(int n){
    for(int j=1;j<=20;j++){
        for(int i=1;i<=n;i++){
            if(i+(1<<j)-1<=n){
                a[i][j]=min(a[i][j-1],a[i+(1<<(j-1))][j
                    -1]);
            }
        }
    }
}
int main(){
// input a[i][0]
    rmq_st(n);
    while(q--){
        int l,r;    scanf("%d%d",&l,&r);
        int k=log2(r-l+1);
        printf("%d ",min(a[l][k],a[r-(1<<k)+1][k]));
    }
}
```