**分治**

2020.09.05

目录

[分治 2](#_Toc50206526)

[归并排序 2](#_Toc50206527)

[整体二分 2](#_Toc50206528)

[CDQ分治 6](#_Toc50206529)

[代码 6](#_Toc50206530)

[典型模型 9](#_Toc50206531)

[点分治 9](#_Toc50206532)

# 分治

## 归并排序

## 整体二分

整体二分就是将一个量（一般为答案），进行二分，对于已经满足的，就分到mid以左的部分递归进行二分，直到左边界等于右边界，如果没有满足就直接剪掉已经得到的部分，分到mid以右的部分进行递归实现。

与分治最大的一个区别就是，一般来说整体二分是先处理影响再向下递归，而分治为先递归再处理影响。

模板 2527

#include<iostream>   
#include<stdio.h>   
#include<string.h>   
#include<algorithm>   
#include<vector>   
#define mid ((nl+nr)>>1)   
#define maxn 300005   
using namespace std;   
typedef long long ll;   
inline int read()   
{   
 int x=0,f=1;char ch=getchar();   
 while(ch<'0'||ch>'9'){if(ch=='-')f=-1;ch=getchar();}   
 while(ch>='0'&&ch<='9'){x=x\*10+ch-'0';ch=getchar();}   
 return x\*f;   
}   
//以下为基本数组   
int n,m,k;   
int res[maxn];   
int id[maxn],temp[maxn];   
int bel[maxn],ned[maxn];   
int z[maxn],y[maxn],t[maxn];   
vector<int>edge[maxn];   
//以下为树状数组   
ll dat[maxn];   
int lowbit(int now)   
{   
 return (now&(-now));   
}   
void update(int now,int add)   
{   
 for(int i=now;i<=m;i+=lowbit(i))   
 {   
 dat[i]+=add;   
 }   
}   
ll query(ll pos)   
{   
 ll ans=0;   
 for(int i=pos;i>=1;i-=lowbit(i))   
 {   
 ans+=dat[i];   
 }   
 return ans;   
}   
void insert(int now,int op)   
{   
 if(z[now]<=y[now])   
 {   
 update(z[now],op\*t[now]);   
 update(y[now]+1,-op\*t[now]);   
 }   
 else   
 {   
 update(z[now],op\*t[now]);   
 update(1,op\*t[now]);   
 update(y[now]+1,-op\*t[now]);   
 }   
}   
ll sum[maxn];   
//dfs(当前处理到的询问，可能成为答案的区间)   
void dfs(int ql,int qr,int nl,int nr)   
{   
 if(ql>qr) return;   
 if(nl==nr)   
 {   
 for(int i=ql;i<=qr;i++)   
 {   
 res[id[i]]=nl;   
 }   
 return;   
 }   
   
 for(int i=nl;i<=mid;i++)   
 {   
 insert(i,1);   
 }   
   
 for(int i=ql;i<=qr;i++)//这里要注意是ql到qr，不要超了，否则复杂度就过大了。   
 {   
 sum[id[i]]=0;   
 }   
   
 for(int i=ql;i<=qr;i++)   
 {   
 int len1=edge[id[i]].size();   
 for(int j=0;j<len1;j++)   
 {   
 sum[id[i]]+=query(edge[id[i]][j]);   
 if(sum[id[i]]>=ned[id[i]]) break;   
 }   
 }   
 int cnt=0;   
 for(int i=ql;i<=qr;i++)//满足的放一边，不满足的放另外一边，因此要先统计   
 {   
 if(sum[id[i]]>=ned[id[i]])   
 cnt++;   
 }   
 int l1=ql,l2=ql+cnt;   
 for(int i=ql;i<=qr;i++)   
 {   
 if(sum[id[i]]>=ned[id[i]])   
 {   
 temp[l1++]=id[i];   
 }   
 else   
 {   
 temp[l2++]=id[i];   
 ned[id[i]]-=sum[id[i]];//这里记得减去之前的   
 }   
 }   
   
 for(int i=ql;i<=qr;i++)   
 id[i]=temp[i];   
   
 for(int i=nl;i<=mid;i++)   
 {   
 insert(i,-1);//记得清除之前的更新   
 }   
 dfs(ql,ql+cnt-1,nl,mid);   
 dfs(ql+cnt,qr,mid+1,nr);   
}   
int main()   
{   
 n=read();m=read();   
 for(int i=1;i<=m;i++)   
 {   
 int x;   
 x=read();   
 edge[x].push\_back(i);   
 }   
 for(int i=1;i<=n;i++)   
 {   
 ned[i]=read();   
 id[i]=i;   
 }   
 k=read();   
 for(int i=1;i<=k;i++)   
 {   
 z[i]=read();   
 y[i]=read();   
 t[i]=read();   
 }   
   
 k++;   
 z[k]=1;y[k]=m;t[k]=1000000009;   
 dfs(1,n,1,k);   
   
 for(int i=1;i<=n;i++)   
 {   
 if(res[i]!=k) printf("%d\n",res[i]);   
 else printf("NIE\n");   
 }   
}

例题：

2527 入门题，每次下一半的流星雨，看每个位置下够没下够，够了向左递归，没够向右递归。

3110 有区间加数的k大数查询，对答案的值进行二分，修改操作跟着一起二分，注意内部要按时间排序，因为修改操作会对答案有影响。

4009 给定树上两种路径，第二种路径有权值，求第一种路径包含的第二种路径里权值的第k大。

这是一道**神仙题**，思路我觉得很有推广的余地。

把第一种路径看成点 这里假设 。

如果第一种路径包含了第二种路径，且第二种路径的不是它的某个端点，那么必然有

如果第一种路径包含了第二种路径，且第二种路径的是它的某个端点，我们设路径上的第一个儿子，那么下列两个式子一定有一个成立。

起点在的左边，终点在y的底下

起点在y的底下，终点在x的右边。

这样我们就成功把第二种路径拆成了一个，或者两个互不相交的矩形，现在要求的是对一个点来说，覆盖它的矩形里的权值第大，整体二分之后扫面线统计即可。

## CDQ分治

### 代码

CDQ分治的精髓就在于，最初是按某个维度X排好序，先递归处理左边的，再递归处理右边的，递归回来之后首先有之前排好序的性质，即左边的X小于等于右边的X，又有了左边的每个元素是按照另外一维Y排好序的。这样递归上来之后，接着依赖于这种排好序的性质，再处理左边对右边的影响，顺便对Y进行排序再往回回溯。

CDQ一般求解二、三维偏序问题。

代码 3262陌上花开

#include<iostream>  
#include<stdio.h>  
#include<string.h>  
#include<algorithm>  
#define maxn 200005  
#define mid ((l+r)>>1)  
using namespace std;  
struct nod  
{  
 int x,y,z,id;  
 nod()  
 {  
 x=0;  
 y=0;  
 z=0;  
 id=0;  
 }  
}a[maxn],b[maxn];  
int n,k;  
int dat[maxn];  
int ans[maxn];  
bool operator<(nod a,nod b)  
{  
 return a.x!=b.x ? a.x<b.x : (a.y!=b.y ? a.y<b.y : a.z<b.z);  
}  
bool operator==(nod a,nod b)  
{  
 return a.x==b.x && a.y==b.y && a.z==b.z;  
}  
int lowbit(int now)  
{  
 return (now&(-now));  
}  
void add(int x,int val)  
{  
 for(int i=x;i<=k;i+=lowbit(i))  
 dat[i]+=val;  
}  
int query(int x)  
{  
 int ans=0;  
 for(int i=x;i;i-=lowbit(i))  
 ans+=dat[i];  
 return ans;  
}  
void cdq(int l,int r)  
{  
 if(l>=r) return;  
 cdq(l,mid);  
 cdq(mid+1,r);  
 int s1=l,s2=mid+1;  
 while(s1<=mid || s2<=r)  
 {  
 if(s2>r || (s1<=mid && a[s1].y<=a[s2].y))  
 {  
 add(a[s1].z,1);  
 b[l+(s1-l)+(s2-(mid+1))]=a[s1];  
 s1++;  
 }  
 else  
 {  
 ans[a[s2].id]+=query(a[s2].z);  
 b[l+(s1-l)+(s2-(mid+1))]=a[s2];  
 s2++;  
 }  
 }  
 for(int i=l;i<=mid;i++)  
 add(a[i].z,-1);  
 for(int i=l;i<=r;i++)  
 a[i]=b[i];  
}  
int cnt[maxn];  
int main()  
{  
 scanf("%d%d",&n,&k);  
 for(int i=1;i<=n;i++)  
 {  
 scanf("%d%d%d",&a[i].x,&a[i].y,&a[i].z);  
 a[i].id=i;  
 }  
 sort(a+1,a+1+n);  
   
 for(int i=n;i>=1;i--)  
 {  
 if(a[i]==a[i+1])  
 ans[a[i].id]=ans[a[i+1].id]+1;  
 //如果两个元素完全相等的话，它右边的也会对它的答案产生贡献，需要加上。  
 }  
   
 cdq(1,n);  
 for(int i=1;i<=n;i++)  
 cnt[ans[i]]++;  
 for(int i=0;i<n;i++)  
 printf("%d\n",cnt[i]);   
}

3262陌上花开 做法都在上文讲了，这里处理左边对右边影响的时候，利用了双指针的思路，先移动左边的保证右边的所对应的y大于等于左边的，接着通过树状数组统计第三维。不过要注意如果两个元素完全相等的话，它右边的也会对它的答案产生贡献，需要加上。

### 典型模型

#### 优化dp

#### 二维平面统计

## 点分治

代码（IOI2011 Race）

#include<iostream>

#include<cstdio>

#include<cstdlib>

#include<cstring>

#include<algorithm>

#include<cmath>

using namespace std;

const int N = 2e5 + 10;

const int M = 1e6 + 10;

const int inf = 1e6 + 10;

struct EDGE{

int to, nxt, v;

EDGE(){}

EDGE(int x, int y, int z){to = x; nxt = y; v = z;}

}edge[N \* 2];

int t[N], size[N], msize[N], dist[N], dep[N], mdist[M], vist[N];

int ans, cnt, tot, top, rt, k;

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

{

edge[++cnt] = EDGE(y, t[x], v);

t[x] = cnt;

}

void getrt(int x,int f)

{

size[x] = 1;

msize[x] = 0;

for(int p = t[x]; p; p = edge[p].nxt)

{

int y = edge[p].to;

if(!vist[y] && f != y)

{

getrt(y, x);

size[x] += size[y];

msize[x] = max(msize[x], size[y]);

}

}

msize[x] = max(msize[x], tot - size[x]);

if(msize[rt] > msize[x]) rt = x;

}

void getdep(int x, int f, int v, int u)

{

if(v > k) return;

dist[++top] = v;

dep[top] = u;

for(int p = t[x]; p; p = edge[p].nxt)

{

int y = edge[p].to;

if(!vist[y] && y != f)

getdep(y, x, v + edge[p].v, u + 1);

}

}

void getans(int x, int f)

{

top = 0;

for(int p = t[x]; p; p = edge[p].nxt)

{

int y = edge[p].to;

if(!vist[y] && y != f)

{

int pre = top;

getdep(y, x, edge[p].v, 1);

for(int i = pre + 1; i <= top; i++)

if(k >= dist[i]) ans = min(ans, mdist[k - dist[i]] + dep[i]);

for(int i = pre + 1; i <= top; i++)

if(dist[i] <= k) mdist[dist[i]] = min(mdist[dist[i]], dep[i]);

}

}

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

if(dist[i] <= k)

mdist[dist[i]] = inf;

}

void solve(int x)

{

getans(x, 0);

vist[rt] = 1;

for(int p = t[x]; p; p = edge[p].nxt)

{

int y = edge[p].to;

if(!vist[y])

{

tot = size[y];

rt = 0;

getrt(y,x);

solve(rt);

}

}

}

int main()

{

int n, x, y, v;

ans = inf;

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

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

{

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

x++; y++;

addedge(x, y, v);

addedge(y, x, v);

}

msize[0] = inf;

tot = n;

getrt(1,0);

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

mdist[i] = inf;

solve(rt);

if(ans == inf)

ans = -1;

printf("%d",ans);

return 0;

}