HDU-6198-矩阵快速幂求斐波那契+思维

#include <bits/stdc++.h>

using namespace std;

typedef long long ll;

const int MAXN=2;

const ll MOD=998244353;

struct Matrix{

ll a[MAXN][MAXN];

void init(){

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

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

a[i][i]=1;

}

};

Matrix mul(Matrix a, Matrix b)

{

Matrix ans;

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

{

for(int j=0;j<MAXN;j++)

{

ans.a[i][j]=0;

for(int k=0;k<MAXN;k++)

ans.a[i][j]=(ans.a[i][j]+(a.a[i][k]\*b.a[k][j])%MOD)%MOD;

}

}

return ans;

}

Matrix qpow(Matrix a, ll n)

{

Matrix ans;

ans.init();

while(n)

{

if(n&1)

ans = mul(ans, a);

a=mul(a,a);

n>>=1;

}

return ans;

}

int main()

{

ll n;

while(scanf("%lld",&n)!=EOF)

{

n=2\*n+3;

Matrix a;

a.a[0][0]=1;a.a[0][1]=1;

a.a[1][0]=1;a.a[1][1]=0;

Matrix s;

s.a[0][0]=1;

s.a[1][0]=0;

Matrix ans=qpow(a,n);

ans=mul(ans,s);

printf("%lld\n",ans.a[1][0]-1);

}

return 0;

}

HDU-6201-Spfa最长路

#include <iostream>

#include <cstdio>

#include <algorithm>

#include <queue>

#include <cstring>

using namespace std;

const int MAXN=100000+100;

struct node{

int to,w,next;

}edge[MAXN<<2];

int cnt,head[MAXN],dis[MAXN];

bool vis[MAXN];

void Init()

{

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

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

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

cnt=0;

}

void add\_edge(int from,int to,int w)

{

edge[cnt].w=w;

edge[cnt].to=to;

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

head[from]=cnt++;

}

void Spfa(int s)

{

queue<int> q;

dis[s]=0;

q.push(s);

vis[s]=true;

while(!q.empty())

{

int t=q.front();

q.pop();

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

{

if(dis[edge[i].to]<dis[t]+edge[i].w)

{

dis[edge[i].to]=dis[t]+edge[i].w;

if(!vis[edge[i].to])

{

q.push(edge[i].to);

vis[edge[i].to]=true;

}

}

}

vis[t]=false;

}

}

int main()

{

int T;

scanf("%d",&T);

int n;

while(T--)

{

Init();

scanf("%d",&n);

int temp;

int s=n+1,e=n+2;

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

{

scanf("%d",&temp);

add\_edge(s,i,temp);

add\_edge(i,e,-temp);

}

int from,to,w;

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

{

scanf("%d %d %d",&from,&to,&w);

add\_edge(from,to,-w);

add\_edge(to,from,-w);

}

Spfa(s);

printf("%d\n",dis[e]);

}

return 0;

}

HDU-4407-Sum(容斥1-n与m互素的数个数变形)

#include <iostream>

#include <cstdio>

#include <algorithm>

#include <vector>

#include <cmath>

#include <map>

using namespace std;

typedef long long ll;

vector<ll> v;

map<ll,ll> vis;

ll Get\_sum(ll p,ll n)

{

v.clear();

for(ll i=2;i\*i<=p;i++)

{

if(p%i==0)

{

v.push\_back(i);

while(p%i==0)

p/=i;

}

}

if(p>1)

v.push\_back(p);

ll tsum=0;

for(ll i=1;i<(1LL<<v.size());i++)

{

ll cnt=0,mul=1;

for(ll j=0;j<v.size();j++)

{

if(i&(1<<j))

{

cnt++;

mul\*=v[j];

}

}

ll temp=n/mul;

if(cnt&1)

tsum+=(((temp\*(temp+1))/2)\*mul);

else

tsum-=(((temp\*(temp+1))/2)\*mul);

}

return ((n\*(n+1))/2)-tsum;

}

int main()

{

int T;

scanf("%d",&T);

ll n,m;

while(T--)

{

vis.clear();

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

ll com,x,y,p;

while(m--)

{

scanf("%lld",&com);

if(com==1)

{

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

ll tans=Get\_sum(p,y)-Get\_sum(p,x-1);

for(map<ll,ll>::iterator it=vis.begin();it!=vis.end();it++)

{

if(x<=it->first&&it->first<=y)

{

if(\_\_gcd(it->first,p)==1)

tans-=it->first;

if(\_\_gcd(it->second,p)==1)

tans+=it->second;

}

}

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

}

else

{

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

vis[x]=p;

}

}

}

return 0;

}

树链剖分BZOJ1036

树链剖分就是把树拆成一系列链，然后用数据结构对链进行维护。

通常的剖分方法是轻重链剖分，所谓轻重链就是对于节点u的所有子结点v，size[v]最大的v与u的边是重边，其它边是轻边，其中size[v]是以v为根的子树的节点个数，全部由重边组成的路径是重路径，根据论文上的证明，任意一点到根的路径上存在不超过logn条轻边和logn条重路径。

这样我们考虑用数据结构来维护重路径上的查询，轻边直接查询。

通常用来维护的数据结构是线段树，splay较少见。

具体步骤

预处理

第一遍dfs求出树每个结点的深度deep[x]，其为根的子树大小size[x]

以及祖先的信息fa[x][i]表示x往上距离为2^i的祖先

第二遍dfs

根节点为起点，向下拓展构建重链

选择最大的一个子树的根继承当前重链

其余节点，都以该节点为起点向下重新拉一条重链

给每个结点分配一个位置编号，每条重链就相当于一段区间，用数据结构去维护。

把所有的重链首尾相接，放到同一个数据结构上，然后维护这一个整体即可

修改操作

1、单独修改一个点的权值

根据其编号直接在数据结构中修改就行了。

2、修改点u和点v的路径上的权值

（1）若u和v在同一条重链上

直接用数据结构修改pos[u]至pos[v]间的值。

（2）若u和v不在同一条重链上

一边进行修改，一边将u和v往同一条重链上靠，然后就变成了情况（1）。

查询操作

查询操作的分析过程同修改操作

题目不同，选用不同的数据结构来维护值，通常有线段树和splay

#include<cstdio>

#include<cstring>

#include<cstdlib>

#include<iostream>

#define inf 0x7fffffff

#define N 30005

#define M 60005

using namespace std;

int n,q,cnt,sz;

int v[N],dep[N],size[N],head[N],fa[N];

int pos[N],bl[N];

struct data{int to,next;}e[M];

struct seg{int l,r,mx,sum;}t[100005];

void insert(int u,int v)

{

e[++cnt].to=v;e[cnt].next=head[u];head[u]=cnt;

e[++cnt].to=u;e[cnt].next=head[v];head[v]=cnt;

}

void ini()

{

scanf("%d",&n);

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

{

int x,y;

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

insert(x,y);

}

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

}

void dfs1(int x)

{

size[x]=1;

for(int i=head[x];i;i=e[i].next)

{

if(e[i].to==fa[x])continue;

dep[e[i].to]=dep[x]+1;

fa[e[i].to]=x;

dfs1(e[i].to);

size[x]+=size[e[i].to];

}

}

void dfs2(int x,int chain)

{

int k=0;sz++;

pos[x]=sz;//分配x结点在线段树中的编号

bl[x]=chain;

for(int i=head[x];i;i=e[i].next)

if(dep[e[i].to]>dep[x]&&size[e[i].to]>size[k])

k=e[i].to;//选择子树最大的儿子继承重链

if(k==0)return;

dfs2(k,chain);

for(int i=head[x];i;i=e[i].next)

if(dep[e[i].to]>dep[x]&&k!=e[i].to)

dfs2(e[i].to,e[i].to);//其余儿子新开重链

}

void build(int k,int l,int r)//建线段树

{

t[k].l=l;t[k].r=r;

if(l==r)return;

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

build(k<<1,l,mid);

build(k<<1|1,mid+1,r);

}

void change(int k,int x,int y)//线段树单点修改

{

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

if(l==r){t[k].sum=t[k].mx=y;return;}

if(x<=mid)change(k<<1,x,y);

else change(k<<1|1,x,y);

t[k].sum=t[k<<1].sum+t[k<<1|1].sum;

t[k].mx=max(t[k<<1].mx,t[k<<1|1].mx);

}

int querysum(int k,int x,int y)//线段树区间求和

{

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

if(l==x&&y==r)return t[k].sum;

if(y<=mid)return querysum(k<<1,x,y);

else if(x>mid)return querysum(k<<1|1,x,y);

else {return querysum(k<<1,x,mid)+querysum(k<<1|1,mid+1,y);}

}

int querymx(int k,int x,int y)//线段树区间求最大值

{

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

if(l==x&&y==r)return t[k].mx;

if(y<=mid)return querymx(k<<1,x,y);

else if(x>mid)return querymx(k<<1|1,x,y);

else {return max(querymx(k<<1,x,mid),querymx(k<<1|1,mid+1,y));}

}

int solvesum(int x,int y)

{

int sum=0;

while(bl[x]!=bl[y])

{

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

sum+=querysum(1,pos[bl[x]],pos[x]);

x=fa[bl[x]];

}

if(pos[x]>pos[y])swap(x,y);

sum+=querysum(1,pos[x],pos[y]);

return sum;

}

int solvemx(int x,int y)

{

int mx=-inf;

while(bl[x]!=bl[y])

{

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

mx=max(mx,querymx(1,pos[bl[x]],pos[x]));

x=fa[bl[x]];

}

if(pos[x]>pos[y])swap(x,y);

mx=max(mx,querymx(1,pos[x],pos[y]));

return mx;

}

void solve()

{

build(1,1,n);

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

change(1,pos[i],v[i]);

scanf("%d",&q);

char ch[10];

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

{

int x,y;scanf("%s%d%d",ch,&x,&y);

if(ch[0]=='C'){v[x]=y;change(1,pos[x],y);}

else

{

if(ch[1]=='M')

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

else

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

}

}

}

int main()

{

ini();

dfs1(1);

dfs2(1,1);

solve();

return 0;

}

最长上升子序列

类似的，我们可以通过二分查找中改变“上确界”和“下确界”，以及符号（“<”和“<=”或“>”、“>=”等），求出最长不下降、不上升、严格下降子序列等问题。

#include<cstdio>

#include<algorithm>

const int MAXN=200001;

int a[MAXN];

int d[MAXN];

int main()

{

int n;

scanf("%d",&n);

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

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

d[1]=a[1];

int len=1;

for(int i=2;i<=n;i++)

{

if(a[i]>d[len])

d[++len]=a[i];

else

{

int j=std::lower\_bound(d+1,d+len+1,a[i])-d;

d[j]=a[i];

}

}

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

return 0;

}

2进制分解

#include <iostream>

#include <cstdio>

#include <algorithm>

#include <cmath>

using namespace std;

int main()

{

int n;

cin>>n;

cout<<(int)log2(n)+1<<endl;

return 0;

}

容斥1-n与m互素的数个数

vector<LL> p;

LL fun(LL n,LL m)

{

p.clear();

for(LL i=2;i\*i<=n;i++)

{

if(n%i==0)

{

p.push\_back(i);

while(n%i==0)

n/=i;

}

}

if(n>1) p.push\_back(n);///求n的素因子

LL S=0;

for(LL i=1;i<(1<<p.size());i++)///从1一直枚举到1<<素因子个数

{

LL cnt=0;

LL mul=1;

for(LL j=0;j<p.size();j++)///枚举每个素因子

{

if(i&(1<<j))///若对应位置为1那么计入

{

cnt++;

mul\*=p[j];

}

}

if(cnt&1)///容斥原理，选取的个数为奇数个，加

S+=m/mul;

else S-=m/mul;///选取的个数为偶数个，减

}

return m-S;///S求得的是1-m中与n不互素的数的个数，返回的是1-m中与n互素的数的个数

}

最小正整数解

int exGcd(int a,int b,int &x,int &y)

{

if(b==0)

{

x=1;y=0;

return a;

}

int r=exGcd(b,a%b,x,y);

int t=x;x=y;y=t-a/b\*y;

return r;

}

int cal(int a,int b,int c)

{

int x,y;

int gcd=(a,b,x,y);

if(c%gcd!=0)

return -1;//代表无解

// ax0+by0=gcd(a,b) 方程一

//同时乘以c/gcd(a,b)得

// (a\*c/gcd(a,b))\*x0+(b\*c/gcd(a,b))\*y0=c;

// 令 x1=c/gcd(a,b)\*x0 y1=c/gcd(a,b)\*y0;

// 则可得 ax1+by1=c 方程二

// 这时得出方程的一个解 x1=x0\*c/gcd(a,b) y1=y0\*c/gcd(a,b)

x\*=c/gcd; //将 方程一的一个特解转化成方程2的一个特解

//套用上文的公式可得对方程二

// b'=b/gcd(a,b);

b/=gcd;

if(b<0)//处理小于0的特殊情况

b=-b;

//对特解x +- kb' 找到最小整数解

//设x=kb'+r

//那么我们想要求的整数解就是r

//直接取模运算即可

int ans=x%b;

//把负数的r转化成正数的

if(ans<=0)

ans+=b;

return ans;

}

字典树优化

#include <iostream>

#include <cstdio>

#include <algorithm>

#include <cstring>

using namespace std;

const int sigma\_size=26;

struct Node{

Node \*L,\*R;

char ch;

int flag;

Node(){

L=R=NULL;

ch=flag=0;

}

};

Node \*root;

void Init()

{

root=new Node();

}

void Insert(char \*s)

{

int len=strlen(s);

Node \*now=root;

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

{

Node \*j,\*pre;

if(now->L==NULL)

{

now->L=new Node();

now->L->ch=s[i];

}

for(j=now->L;j;j=j->R)

{

pre=j;

if(j->ch==s[i])

break;

}

if(pre->ch!=s[i])

{

pre->R=new Node();

pre->R->ch=s[i];

j=pre->R;

}

now=j;

now->flag++;

}

}

int Query(char \*s)

{

int len=strlen(s);

Node \*now=root;

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

{

Node \*j;

for(j=now->L;j;j=j->R)

if(j->ch==s[i])

break;

if(j==NULL)

return 0;

now=j;

}

return now->flag;

}

int main()

{

char s[20];

Init();

while(gets(s),strcmp(s,""))

Insert(s);

while(gets(s)!=NULL)

printf("%d\n",Query(s));

return 0;

}

KMP

#include <iostream>

#include <cstdio>

#include <algorithm>

#include <cstring>

using namespace std;

void Get\_fail(char \*P, int \*f){

int m = strlen(P);

f[0] = 0; f[1] = 0;

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

int j = f[i];

while(j && P[i] != P[j]) j=f[j];

f[i+1] = (P[i] == P[j] ? j + 1 : 0);

}

}

int KMP(char \*P,char \*T,int \*f){

int n = strlen(T), m = strlen(P);

Get\_fail(P,f);

int j = 0;

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

while(j && P[j] != T[i]) j=f[j];

if(P[j] == T[i]) j++;

if(j == m)

return i - m + 1;

}

return -1;

}

int main(){

char P[1000], T[1000];

int f[1000];

cin >> P >> T;

cout << KMP(P,T,f) << endl;

return 0;

}

HDU – 1255 矩形面积交 扫描线

#include <iostream>

#include <cstdio>

#include <algorithm>

#include <vector>

using namespace std;

const int MAXN=10000;

vector<double > v;

struct Edge{

double l,r,h;

int f;

bool operator <(const Edge& a) const{

return h<a.h;

}

}edge[MAXN<<1];

struct Tree{

int l,r;

double len;

int f;

}tree[MAXN<<3];

int Get\_id(double key)

{

return lower\_bound(v.begin(),v.end(),key)-v.begin()+1;

}

void Build(int l,int r,int cur)

{

tree[cur].l=l;tree[cur].r=r;tree[cur].f=0;tree[cur].len=0;

if(l+1==r)

return;

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

Build(l,mid,cur<<1);

Build(mid,r,cur<<1|1);

}

void Pushup(int cur)

{

if(tree[cur].f>=1)

tree[cur].len=v[tree[cur].r-1]-v[tree[cur].l-1];

else if(tree[cur].l+1==tree[cur].r)

tree[cur].len=0;

else

tree[cur].len=tree[cur<<1].len+tree[cur<<1|1].len;

}

void Update(int l,int r,int val,int cur)

{

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

{

tree[cur].f+=val;

Pushup(cur);

return;

}

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

if(l<mid)

Update(l,r,val,cur<<1);

if(r>mid)

Update(l,r,val,cur<<1|1);

Pushup(cur);

}

double Query(int num,int cur)

{

if(num+tree[cur].f>=2)

return tree[cur].len;

if(tree[cur].l+1==tree[cur].r)

return 0;

return Query(num+tree[cur].f,cur<<1)+Query(num+tree[cur].f,cur<<1|1);

}

int main()

{

int T;

scanf("%d",&T);

while(T--)

{

int n;

scanf("%d",&n);

double xd,yd,xu,yu;

int cnt=0;

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

{

scanf("%lf %lf %lf %lf",&xd,&yd,&xu,&yu);

v.push\_back(xd);

v.push\_back(xu);

edge[cnt].l=xd;edge[cnt].r=xu;edge[cnt].h=yd;edge[cnt++].f=1;

edge[cnt].l=xd;edge[cnt].r=xu;edge[cnt].h=yu;edge[cnt++].f=-1;

}

sort(v.begin(),v.end());

sort(edge,edge+cnt);

v.erase(unique(v.begin(),v.end()),v.end());

double ans=0;

Build(1,v.size()+100,1);

for(int i=0;i<cnt-1;i++)

{

Update(Get\_id(edge[i].l),Get\_id(edge[i].r),edge[i].f,1);

ans+=Query(0,1)\*(edge[i+1].h-edge[i].h);

}

printf("%.2lf\n",ans);

}

return 0;

}

HDU – 2642 二维树状数组

#include <iostream>

#include <cstdio>

#include <algorithm>

using namespace std;

const int MAXN=2000;

int tree[MAXN][MAXN];

int book[MAXN][MAXN];

int lowbit(int x)

{

return x&-x;

}

void Update(int x,int y,int val)

{

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

for(int j=y;j<MAXN;j+=lowbit(j))

tree[i][j]+=val;

}

int Query(int x,int y)

{

int ans=0;

for(int i=x;i>0;i-=lowbit(i))

for(int j=y;j>0;j-=lowbit(j))

ans+=tree[i][j];

return ans;

}

int main()

{

int n;

scanf("%d",&n);

while(n--)

{

char com;

int t1,t2,t3,t4;

scanf(" %c %d %d",&com,&t1,&t2);

t1++,t2++;

if(com=='B'&&!book[t1][t2])

{

Update(t1,t2,1);

book[t1][t2]=1;

}

else if(com=='D'&&book[t1][t2])

{

Update(t1,t2,-1);

book[t1][t2]=0;

}

else if(com=='Q')

{

scanf("%d %d",&t3,&t4);

t3++,t4++;

if(t3>t4)

swap(t3,t4);

if(t1>t2)

swap(t1,t2);

printf("%d\n",Query(t2,t4)-Query(t1-1,t4)-Query(t2,t3-1)+Query(t1-1,t3-1));

}

}

return 0;

}

POJ – 3321 DFS序

#include <iostream>

#include <cstdio>

#include <algorithm>

#include <cstring>

using namespace std;

const int MAXN=(int)1e5+10;

struct node{

int to,next;

}edge[2\*MAXN];

int head[MAXN],in[MAXN],out[MAXN];

int cnt=0,tot=0;

int c[MAXN];

bool vis[MAXN];

void add\_edge(int from,int to)

{

edge[cnt].to=to;

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

head[from]=cnt++;

}

void dfs(int cur,int pre)

{

in[cur]=++tot;

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

if(edge[i].to!=pre)

dfs(edge[i].to,cur);

out[cur]=tot;

}

int lowbit(int x)

{

return (-x)&x;

}

void add(int x,int n,int val)

{

while(x<=n)

{

c[x]+=val;

x+=lowbit(x);

}

}

int sum(int x)

{

int ans=0;

while(x)

{

ans+=c[x];

x-=lowbit(x);

}

return ans;

}

int main()

{

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

int n;

scanf("%d",&n);

int u,v;

for(int i=0;i<n-1;i++)

{

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

add\_edge(u,v);

add\_edge(v,u);

}

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

{

vis[i]=true;

add(i,n,1);

}

dfs(1,0);

int m,temp;

char com;

scanf("%d",&m);

while (m--)

{

scanf(" %c %d",&com,&temp);

if(com=='Q')

printf("%d\n",sum(out[temp])-sum(in[temp]-1));

else

{

if(vis[in[temp]])

{

add(in[temp],n,-1);

vis[in[temp]]=false;

}

else

{

add(in[temp],n,1);

vis[in[temp]]=true;

}

}

}

return 0;

}

CodeForces - 787D 线段树优化建图+最短路

/\*

@resouces: codeforces 787D

@date: 2017-3-27

@author: QuanQqqqq

@algorithm: 线段树 + dijkstra nlogn

\*/

#include <cstdio>

#include <algorithm>

#include <iostream>

#include <vector>

#include <queue>

#define MAXN 500005

#define ll long long

using namespace std;

const ll LLINF = 0x3f3f3f3f3fLL;

typedef pair<ll,int> pli;

struct Edge{

int to,weight;

Edge(int \_to = 0,int \_weight = 0) : to(\_to),weight(\_weight){}

};

priority\_queue<pli > Q;

vector<vector<Edge> > G(MAXN);

bool vis[MAXN];

int id[MAXN][2];

vector<int> vs;

ll d[MAXN];

int idx;

void addEdge(int from,int to,int weight){

G[from].push\_back(Edge(to,weight));

}

void build(int l,int r,int rt,int wh){ //wh0为从上到下的线段树，wh1为从下到上的线段树

id[rt][wh] = ++idx;

if(l == r){

if(wh == 0){

addEdge(id[rt][wh],l,0);

} else {

addEdge(l,id[rt][wh],0);

}

return;

}

int mid = l + r >> 1;

build(l,mid,rt << 1,wh);

build(mid + 1,r,rt << 1 | 1,wh);

if(wh == 0){

addEdge(id[rt][wh],id[rt << 1][wh],0);

addEdge(id[rt][wh],id[rt << 1 | 1][wh],0);

} else {

addEdge(id[rt << 1][wh],id[rt][wh],0);

addEdge(id[rt << 1 | 1][wh],id[rt][wh],0);

}

}

void get(int l,int r,int L,int R,int rt,int wh){

if(L <= l && r <= R){

vs.push\_back(id[rt][wh]);

return;

}

int m = l + r >> 1;

if(m >= L){

get(l,m,L,R,rt << 1,wh);

}

if(m < R){

get(m + 1,r,L,R,rt << 1 | 1,wh);

}

}

void dijkstra(int s,int n){

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

vis[i] = false;

d[i] = LLINF;

}

Q.push({-0,s});

d[s] = 0;

while(!Q.empty()){

int u = Q.top().second;

Q.pop();

if(vis[u]){

continue;

}

vis[u] = true;

for(int i = 0;i < G[u].size();i++){

Edge e = G[u][i];

int v = e.to,w = e.weight;

if(w + d[u] < d[v]){

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

Q.push({-d[v],v});

}

}

}

}

void init(int n){

while(!Q.empty()){

Q.pop();

}

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

G[i].clear();

}

}

int main(){

int n,q,s,l,r,t,u,v,c;

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

init(n \* 5);

idx = n;

build(1,n,1,0);

build(1,n,1,1);

while(q--){

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

if(t == 1){

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

addEdge(u,v,c);

} else if(t == 2) {

vs.clear();

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

get(1,n,l,r,1,0);

for(int i = 0;i < vs.size();i++){

addEdge(u,vs[i],c);

}

} else {

vs.clear();

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

get(1,n,l,r,1,1);

for(int i = 0;i < vs.size();i++){

addEdge(vs[i],u,c);

}

}

}

dijkstra(s,5 \* n);

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

if(d[i] == LLINF){

d[i] = -1;

}

printf("%lld ",d[i]);

}

}

主席树 区间第K小 POJ 2104

#include<iostream>

#include<cstdio>

#include<algorithm>

#include<vector>

using namespace std;

const int MAXN=1e5+10;

struct Tree{

int l,r,sum;

}T[MAXN\*40];

vector<int> v;

int cnt,root[MAXN],a[MAXN];

void Init()

{

cnt=0;

T[cnt].l=0;T[cnt].r=0;T[cnt].sum=0;

root[cnt]=0;

v.clear();

}

int getid(int x)

{

return lower\_bound(v.begin(),v.end(),x)-v.begin()+1;

}

void Update(int l,int r,int &x,int y,int pos)

{

T[++cnt]=T[y],T[cnt].sum++,x=cnt;

if(l==r)

return;

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

if(mid>=pos)

Updata(l,mid,T[x].l,T[y].l,pos);

else

Updata(mid+1,r,T[x].r,T[y].r,pos);

}

int Query(int l,int r,int x,int y,int k)

{

if(l==r)

return l;

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

int sum=T[T[y].l].sum-T[T[x].l].sum;

if(sum>=k)

return Query(l,mid,T[x].l,T[y].l,k);

else

return Query(mid+1,r,T[x].r,T[y].r,k-sum);

}

int main()

{

Init();

int n,m;

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

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

{

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

v.push\_back(a[i]);

}

sort(v.begin(),v.end());

v.erase(unique(v.begin(),v.end()),v.end());

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

Updata(1,n,root[i],root[i-1],getid(a[i]));

int l,r,k;

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

{

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

printf("%d\n",v[Query(1,n,root[l-1],root[r],k)-1]);

}

return 0;

}

主席树 SPOJ - DQUERY 求区间数字种类数

#include <cstdio>

#include <algorithm>

#include <iostream>

#include <map>

using namespace std;

const int MAXN=30010;

struct node{

int l,r,sum;

}T[MAXN\*40];

int cnt,a[MAXN],root[MAXN];

map<int,int> pos;

void Init()

{

cnt=0;

T[cnt].l=0;T[cnt].r=0;T[cnt].sum=0;

root[cnt]=0;

pos.clear();

}

void Update(int l,int r,int &now,int pre,int pos,int add)

{

T[++cnt]=T[pre];T[cnt].sum+=add;now=cnt;

if(l==r)

return;

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

if(pos<=mid)

Update(l,mid,T[cnt].l,T[pre].l,pos,add);

else

Update(mid+1,r,T[cnt].r,T[pre].r,pos,add);

}

int Query(int l,int r,int root,int left)

{

if(l>=left)

return T[root].sum;

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

if(mid>=left)

return Query(l,mid,T[root].l,left)+T[T[root].r].sum;

else

return Query(mid+1,r,T[root].r,left);

}

int main()

{

int n;

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

{

Init();

int temp;

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

{

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

if(!pos.count(a[i]))//直接加

{

Update(1,n,root[i],root[i-1],i,1);

}

else

{

Update(1,n,temp,root[i-1],pos[a[i]],-1);

Update(1,n,root[i],temp,i,1);

}

pos[a[i]]=i;

}

int q,l,r;

scanf("%d",&q);

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

{

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

printf("%d\n",Query(1,n,root[r],l));

}

}

return 0;

}

#include <iostream>

#include <cstdio>

#include <algorithm>

#include <map>

#include <cstring>

using namespace std;

const int maxn=30000+5;

map<int,int> mp;

int data[maxn],a[maxn],ans[200000+5];

struct node{

int l,r,id;

bool operator<(node t)const{

return r<t.r;

}

}q[200000+5];

int lowbit(int x)

{

return x&-x;

}

int Get\_sum(int i)

{

int ans=0;

while(i>0){

ans+=data[i];

i-=lowbit(i);

}

return ans;

}

void Update(int i,int x)

{

while(i<maxn){

data[i]+=x;

i+=lowbit(i);

}

}

int main()

{

int n;

while(~scanf("%d",&n))

{

memset(data,0,sizeof(data));

mp.clear();

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

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

int m;

scanf("%d",&m);

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

{

scanf("%d%d",&q[i].l,&q[i].r);

q[i].id=i;

}

sort(q,q+m);

int pre=1;

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

{

for(int j=pre;j<=q[i].r;j++)

{

if(mp[a[j]]!=0)

Update(mp[a[j]],-1);

Update(j,1);

mp[a[j]]=j;

}

pre=q[i].r+1;

ans[q[i].id]=Get\_sum(q[i].r)-Get\_sum(q[i].l-1);

}

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

printf("%d\n",ans[i]);

}

return 0;

}

指数循环节

≡ ，其中b≥φ(m)

离散对数模板

<http://www.spoj.com/problems/MOD/>

求解 ≡B (%C) 的最小正整数解 （0<=x<C）

#include <iostream>

#include <string.h>

#include <stdio.h>

#include <math.h>

using namespace std;

typedef long long LL;

const int MOD = 99991;

const int N = 100005;

struct Hash

{

bool f;

int id;

int val;

};

Hash hash[N];

void Init()

{

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

{

hash[i].f = 0;

hash[i].id = -1;

hash[i].val = -1;

}

}

void Insert(int id,LL val)

{

LL t = val % MOD;

while(hash[t].f && hash[t].val != val)

{

t++;

t %= MOD;

}

if(!hash[t].f)

{

hash[t].f = 1;

hash[t].id = id;

hash[t].val = val;

}

}

int Find(LL val)

{

LL t = val % MOD;

while(hash[t].f && hash[t].val != val)

{

t++;

t %= MOD;

}

if(!hash[t].f) return -1;

return hash[t].id;

}

LL gcd(LL a,LL b)

{

return b ? gcd(b,a%b):a;

}

void extend\_Euclid(LL a,LL b,LL &x,LL &y)

{

if(b == 0)

{

x = 1;

y = 0;

return;

}

extend\_Euclid(b,a%b,x,y);

LL tmp = x;

x = y;

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

}

LL Baby\_Step(LL A,LL B,LL C)

{

LL ret = 1;

for(int i=0; i<=50; i++)

{

if(ret == B) return i;

ret = ret \* A % C;

}

LL ans = 1;

LL tmp,cnt = 0;

while((tmp = gcd(A,C)) != 1)

{

if(B % tmp) return -1;

B /= tmp;

C /= tmp;

ans = ans \* (A / tmp) % C;

cnt++;

}

LL M = ceil(sqrt(1.0\*C));

LL t = 1;

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

{

Insert(i,t);

t = t \* A % C;

}

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

{

LL x,y;

extend\_Euclid(ans,C,x,y);

LL val = x \* B % C;

val = (val % C + C) % C;

LL j = Find(val);

if(j != -1) return i \* M + j + cnt;

ans = ans \* t % C;

}

return -1;

}

int main()

{

LL A,B,C;

while(cin>>A>>C>>B)

{

Init();

if(A + B + C == 0) break;

A %= C; B %= C;

LL ans = Baby\_Step(A,B,C);

if(ans == -1)

{

puts("No Solution");

continue;

}

cout<<ans<<endl;

}

return 0;

}

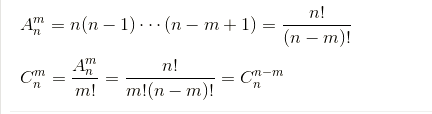
奇奇怪怪的定理猜想

***费马大定理***

又被称为“费马最后的定理”，由17世纪法国数学家皮耶·德·费玛提出。

他断言当整数n >2时，关于x, y, z的方程 x^n + y^n = z^n 没有正整数解。

***排列组合***



***哥德巴赫猜想***

任一大于2的偶数都可写成两个质数之和

***错排公式***

D(n) = n![(-1)^2/2! + … + (-1)^(n-1)/(n-1)! + (-1)^n/n!].

D（n）=（n-1）[D（n-1）+D（n-2）]； D（1）=0； D（2）=1

LightOJ – 1234 调和级数求和公式

//公式

#include <iostream>

#include <cstdio>

#include <cmath>

#include <algorithm>

using namespace std;

const double ER = 0.57721566490153286060651209;

double fn[100005];

void Init() {

fn[0] = 0, fn[1] = 1;

for (int i = 2; i <= 100000; i++)

fn[i] = fn[i - 1] + 1.0 / i;

}

double find(int n) {

return ER + (log(n) + log(n + 1)) / 2;

}

int main() {

Init();

int T, cas = 1;

scanf("%d", &T);

while (T--) {

int n;

scanf("%d", &n);

printf("Case %d: ", cas++);

if (n <= 100000)

printf("%.10lf\n", fn[n]);

else

printf("%.10lf\n", find(n));

}

return 0;

}

快速幂求前k位和后k位 LightOJ – 1282

#include <iostream>

#include <algorithm>

#include <cstdio>

#include <cmath>

using namespace std;

typedef long long ll;

const ll MOD=1000;

ll qpow(ll a,ll b){

ll ans=1;

while(b){

if(b&1)

ans=ans\*a%MOD;

a=a\*a%MOD;

b>>=1;

}

return ans;

}

int main(){

int T,cas=1;

scanf("%d",&T);

while (T--){

ll n,k;

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

ll ans=(ll)pow(10.0,2.0+fmod(k\*log10(n),1.0));

printf("Case %d: %03lld %03lld\n",cas++,ans,qpow(n,k));

}

return 0;

}

整除分块 LightOJ – 1245

#include <iostream>

#include <algorithm>

#include <vector>

#include <cstdio>

using namespace std;

typedef long long ll;

int main() {

int T,cas=1;

scanf("%d", &T);

while (T--) {

ll n, ans = 0;

scanf("%lld", &n);

for (ll l = 1, r; l <= n; l = r + 1) {

r = n / (n / l);

ans += (r - l + 1) \* (n / l);

}

printf("Case %d: %lld\n",cas++,ans);

}

return 0;

}

二分 阶乘末尾0的数量 LightOJ – 1138

//不想写了 粘题解 这题很简单 wa那一次是因为想当然了

#include <math.h>

#include <vector>

#include <queue>

#include <string>

#include <string.h>

#include <stdlib.h>

#include <iostream>

#include <algorithm>

using namespace std;

typedef long long LL;

LL solve(LL n)

{

LL num=0;

while(n){

num+=n/5;

n/=5;

}

return num;

}

LL er(LL n)

{

LL x=1;

LL y=1844674407370;

LL mid;

LL res=-1;

while(y>=x){

mid=(x+y)/2;

LL ans=solve(mid);

if(ans==n){

res=mid;

y=mid-1;

//return mid;

}

else if(ans>n){

y=mid-1;

}

else if(ans<n){

x=mid+1;

}

}

return res;

}

int main()

{

int t;

scanf("%d",&t);

int xp=1;

while(t--){

LL n;

scanf("%lld",&n);

LL ans=er(n);

if(ans==-1) printf("Case %d: impossible\n",xp++);

else printf("Case %d: %d\n",xp++,ans);

}

return 0;

}