目录

[杂物 3](#_Toc488430740)

[字符串的 5](#_Toc488430741)

[KMP 5](#_Toc488430742)

[字典树 6](#_Toc488430743)

[AC自动机 一个匹配多个 7](#_Toc488430744)

[后缀数组 11](#_Toc488430745)

[马拉车 12](#_Toc488430746)

[数据结构 14](#_Toc488430747)

[树状数组 区间max 14](#_Toc488430748)

[树状数组 区间和 15](#_Toc488430749)

[二维树状数组 区间修改单点查询 16](#_Toc488430750)

[不大于k的最大值 17](#_Toc488430751)

[线段树 19](#_Toc488430752)

[最长连续子区间 20](#_Toc488430753)

[暴力 22](#_Toc488430754)

[二维线段树 23](#_Toc488430755)

[扫描线 矩形周长并 24](#_Toc488430756)

[主席树-区间第k大 28](#_Toc488430757)

[往后多少位在哪里 28](#_Toc488430758)

[区间不重复数字个数和第k个是哪位 29](#_Toc488430759)

[图论 32](#_Toc488430760)

[二分图,匈牙利算法 32](#_Toc488430761)

[最短路 34](#_Toc488430762)

[网络流 38](#_Toc488430763)

[上下界网络流 40](#_Toc488430764)

[强连通分量tarjan 42](#_Toc488430765)

[2-sat 44](#_Toc488430766)

[求凸包 47](#_Toc488430767)

[数学相关 48](#_Toc488430768)

[逆元 48](#_Toc488430769)

[数位dp 50](#_Toc488430770)

[博弈：NIM,SG 52](#_Toc488430771)

[莫队 53](#_Toc488430772)

#include <cstdio>

#include <iostream>

#include <algorithm>

#include <vector>

#include <set>

#include <map>

#include <string>

#include <stack>

#include <queue>

#include <cmath>//可能有x,y,x1,y1,x2,y2,注意

using namespace std;

#define REP(I,N) for (I=0;I<N;I++)

#define rREP(I,N) for (I=N-1;I>=0;I--)

#define rep(I,S,N) for (I=S;I<N;I++)

#define rrep(I,S,N) for (I=N-1;I>=S;I--)

#define FOR(I,S,N) for (I=S;I<=N;I++)

typedef unsigned long long ULL;

typedef long long LL;

const int INF=0x3f3f3f3f;

const LL INFF=0x3f3f3f3f3f3f3f3fll;

const LL M=1e9+7;//double可能会报错，强转一下

const LL maxn=1e6+7;

const double eps=0.00000001;

const double pi=acos(-1.0);

LL gcd(LL a){return b?gcd(b,a%b):a;}

template<typename T>

inline T abs(T a) {return a>0?a:-a;}

头文件在上面~~~~~~

其他有用的东西：

struct node{

int x,y;

node(int xx=0,int yy=0):x(xx),y(yy){};

bool operator<(const node &a) const{

if (x<a.x) return 1;

if (x>a.x) return 0;

return y<a.y;

}

};

杂物

int ans;

void fqsort(int l,int r)//第k大

{

int le=l,ri=r,m;

m=a[le];

while (le<ri)

{

while (le<ri&&a[ri]<=m) ri--;

a[le]=a[ri];

while (le<ri&&a[le]>=m) le++;

a[ri]=a[le];

}

if (le==k) printf("%d\n",m);

else if (le>k) fqsort(l,le-1);

else fqsort(le+1,r);

}

void msort(int le,int ri)//逆序对

{

if (le==ri) return;

int mid=(le+ri)>>1,l1=le,r1=mid+1,k1=l1;

msort(le,mid); msort(r1,ri);

while (l1<=mid||r1<=ri)

{

if (l1==mid+1) {b[k1++]=a[r1++]; ans+=mid-l1+1;}

else if (r1==ri+1) b[k1++]=a[l1++];

else if (a[l1]<=a[r1]) b[k1++]=a[l1++];

else {b[k1++]=a[r1++]; ans+=mid-l1+1;}

}

for (l1=le;l1<=ri;l1++) a[l1]=b[l1];

}

**输入挂**

int n,m;

char s[maxn],str[maxn];

int len1,len2,p[maxn],ans;

template<class T>

bool read\_d(T &num){

char in;bool IsN=false;

in=getchar();

if (in==EOF) return false;

while (in!= '-'&&(in<'0'||in>'9')) in=getchar();

if (in=='-') {IsN=1;num=0;}

else num=in-'0';

while (in=getchar(),in>='0'&&in<='9') num=num\*10+in-'0';

if (IsN) num=-num;

return 1;

}

template<class T>

bool read\_f(T &num){

char in;bool IsN=false,IsD=false;

T Dec=0.1;

in=getchar();

if (in==EOF) return false;

while (in!='-'&&in!='.'&&(in<'0'||in>'9')) in=getchar();

if (in=='-') {IsN=1;num=0;}

else if (in=='.') {IsD=1;num=0;}

else num=in-'0';

if (!IsD)

while (in=getchar(),in>='0'&&in<='9') num=num\*10+in-'0';

if (in=='.')

while (in=getchar(),in>='0'&&in<='9') {num+=Dec\*(in-'0');Dec\*=0.1;}

if (IsN) num=-num;

return 1;

}

LL d;

double c;

int main(){

int i;

while (read\_f(c)){

printf("%lf\n",c);

}

}

# 字符串的

KMP

LL n,m;

char s[M],a[N];

LL Next[N];

LL i,j,k,t;

void init(char \*a,LL \*Next){

Next[0]=-1;

int len=strlen(a);

register int i,j;

FOR(i,1,len-1){

j=Next[i-1];

while (j>=0&&a[j+1]!=a[i]) j=Next[j];

if (a[i]==a[j+1]) Next[i]=j+1;

else Next[i]=-1;

}

}

int kmp(char \*s,char \*a,LL \*Next){

int Len=strlen(s),len=strlen(a);

register int i,j=-1;

REP(i,Len){

while (j>=0&&a[j+1]!=s[i]) j=Next[j];

if (s[i]==a[j+1]) j++;

if (j==len-1) return i-len+1;

}

return -1;

}

int main(){

while (~scanf("%s%s",&s,&a)){

init(a,Next);

n=strlen(a);

// REP(i,n) printf("%d ",Next[i]);

t=kmp(s,a,Next);

if (~t) printf("%d",t+1);

else printf("Not Found!");

puts("");

}

}

字典树

LL n,m;

LL a[N][27],f[N],ff[N];//ff[N]:num

LL i,j,k;

int cnt;

string s;

inline void insert(string str){

int len=str.length(),now=0;

int i;

REP(i,len){

if (!a[now][str[i]-'a']) a[now][str[i]-'a']=++cnt;

now=a[now][str[i]-'a'];

++f[now];//表示小于等于这个的有多少

}

ff[now]++;//==的

}

int calc(string str){//小于str的

int len=str.length(),now=0,ans=0;

int i,j;

REP(i,len){

REP(j,str[i]-'a')

ans+=f[a[now][j]];

// if (i!=len-1)//等于的也加

ans+=ff[a[now][str[i]-'a']];

now=a[now][str[i]-'a'];

if (now==0) break;

}

return ans;//求大的要再加上后面的

}

int findstr(string str){//等于的

int len=str.length(),now=0,ans=0,i;

REP(i,len){

now=a[now][str[i]-'a'];

if (now==0) return 0;

}

return ans=ff[now];//可能==0

}

int main(){

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

REP(i,n) {cin>>s;insert(s);}

REP(i,m) {cin>>s;cout<<calc(s)<<'\n';}

}

AC自动机 一个匹配多个

const int maxtot=50\*10007;//个数

const int charnum=26;

int nxt[maxtot][charnum],fail[maxtot],num[maxtot];

int cnt;

queue<int> Q;

void init(){

int i,j;

while (Q.size()) Q.pop();

REP(i,maxtot) {

REP(j,charnum) nxt[i][j]=0;

num[i]=fail[i]=0;

}

cnt=1;

}

inline void insert(char \*str){

int len=strlen(str),now=0,i;

REP(i,len){

int k=str[i]-'a';

if (!nxt[now][k]) nxt[now][k]=cnt++;

now=nxt[now][k];

}

num[now]++;

}

inline void buildAC(){

fail[0]=-1;

Q.push(0);

int i;

while (Q.size()){

int x=Q.front();Q.pop();

REP(i,charnum) if (nxt[x][i]){

if (x==0) fail[nxt[x][i]]=0;

else {

int p=fail[x];

while (p!=-1&&!nxt[p][i]) p=fail[p];//注意这里是nxt[p][i]

if (p!=-1) fail[nxt[x][i]]=nxt[p][i];

else fail[nxt[x][i]]=0;

}

Q.push(nxt[x][i]);

}

}

}

inline int match(char \*str){

int len=strlen(str),now=0;

int i,ret=0;

REP(i,len){

int k=str[i]-'a';

while (now&&!nxt[now][k]) now=fail[now];

now=nxt[now][k];

if (now==-1) now=0;

int tmp=now;

while (tmp){

if (num[tmp]==-1) break;//vis

ret+=num[tmp];

num[tmp]=-1;

tmp=fail[tmp];

}

}

return ret;

}

int T,i,n;

char s[maxn];

int main(){

scanf("%d",&T);

while (T--){

scanf("%d",&n);

init();

REP(i,n){

scanf("%s",s);

insert(s);

}

buildAC();

scanf("%s",s);

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

}

}

//或者~~~~~~~~~~~~~~

int ans[505],num;//标记

const int tot=505; const int maxtot=505\*140; const int charnum=98;

int nxt[maxtot][charnum],fail[maxtot],mark[maxtot];

int cnt;

queue<int> Q;

void init(){

int i,j;

while (Q.size()) Q.pop();

REP(i,maxtot){

REP(j,charnum) nxt[i][j]=0;

mark[i]=fail[i]=0;

}

cnt=1;

}

inline void insert(char \*str,int id){

int len=strlen(str),now=0,i;

REP(i,len){

int k=str[i]-33;

if (!nxt[now][k]) nxt[now][k]=cnt++;

now=nxt[now][k];

}

mark[now]=id;

}

inline void buildAC(){

fail[0]=-1;

Q.push(0);

int i;

while (!Q.empty()){

int x=Q.front();Q.pop();

REP(i,charnum) if (nxt[x][i]){

if (x==0) fail[nxt[x][i]]=0;

else{

int p=fail[x];

while (p!=-1&&!nxt[p][i]) p=fail[p];//这里注意

if (p!=-1) fail[nxt[x][i]]=nxt[p][i];

else fail[nxt[x][i]]=0;

}

Q.push(nxt[x][i]);

}

}

}

inline void match(char \*str){

int len=strlen(str),now=0;

int i;

num=0;

REP(i,tot) ans[i]=0;

REP(i,len){

int k=str[i]-33;

while (now&&!nxt[now][k]) now=fail[now];

now=nxt[now][k];

if (now==-1) now=0;

int tmp=now;

while (tmp&&!ans[mark[tmp]]){

if (mark[tmp]){

ans[mark[tmp]]=1;

num++;

}

tmp=fail[tmp];

if (num>=3) return;

}

}

}

int T,i,j,n,m,total;

char s[maxn];

int main(){

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

total=0;

init();

REP(i,n){

scanf("%s",s);

insert(s,i+1);

}

buildAC();

scanf("%d",&m);

REP(i,m){

scanf("%s",s);

match(s);

if (num==0) continue;

total++;

printf("web %d:",i+1);

REP(j,tot) if (ans[j]) printf(" %d",j);

puts("");

}

printf("total: %d\n",total);

}

}

后缀数组

int wa[maxn],wb[maxn],wv[maxn],ws1[maxn];

int cmp(int \*r,int a,int b,int l){

return r[a]==r[b]&&r[a+l]==r[b+l];

}

//sa->pos(后缀排名->pos)

void da(int \*r,int \*sa,int n,int m){

r[n++]=0;//使rank从1开始(sa[0]=n)

int i,j,p,\*x=wa,\*y=wb,\*t;

REP(i,m) ws1[i]=0;//pre-cmp

REP(i,n) ws1[x[i]=r[i]]++;//r->x

rep(i,1,m) ws1[i]+=ws1[i-1];

rREP(i,n) sa[--ws1[x[i]]]=i;//sort(计数排序)

for (j=1,p=1;p<n;j<<=1,m=p){//j->2^x

p=0;rep(i,n-j,n) y[p++]=i;//最后j个是不用加(显然)

REP(i,n) if (sa[i]>=j) y[p++]=sa[i]-j;//后缀顺序

REP(i,n) wv[i]=x[y[i]];//x+y->wv(由于后缀顺序)

REP(i,m) ws1[i]=0;

REP(i,n) ws1[wv[i]]++;

rep(i,1,m) ws1[i]+=ws1[i-1];

rREP(i,n) sa[--ws1[wv[i]]]=y[i];//sort(计数排序)

t=x,x=y,y=t;

p=1;x[sa[0]]=0;

rep(i,1,n) x[sa[i]]=cmp(y,sa[i-1],sa[i],j)?p-1:p++;

}

}

int rank[maxn],height[maxn];

void calheight(int \*r,int \*sa,int n){

int i,j,k=0;

FOR(i,1,n) rank[sa[i]]=i;

REP(i,n){//线性按照从前往后找,充分利用性质

if (k) k--;

j=sa[rank[i]-1];

while (r[i+k]==r[j+k]) k++;

height[rank[i]]=k;

}

}

char a[maxn],b[maxn],c[maxn];

int sa[maxn],r[maxn];

int i,j,k;

int n,m,t,ans;

int main()

{

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

n=strlen(a);

m=strlen(b);

REP(i,n) c[t++]=a[i];

c[t++]='z'+1;

REP(i,m) c[t++]=b[i];

REP(i,t) r[i]=c[i]-'a'+1;

//REP(i,t) printf("%c",r[i]+'a'-1);

da(r,sa,t,200);

calheight(r,sa,n+m+1);

// FOR(i,1,n) printf("%d\n",sa[i]);

rep(i,1,n+m+1)

if ((sa[i]<n)^(sa[i-1]<n)) ans=max(ans,height[i]);

// FOR(i,1,n+m+1) printf("%s %d %d\n",&c[sa[i]],sa[i],height[i]);

printf("%d",ans);

}

马拉车

int n,m;

char s[maxn],str[maxn];

int len1,len2,p[maxn],ans;

void init(){

ans=0;

int i;

str[0]='+';

str[1]='%';

REP(i,len1+1){

str[i\*2+2]=s[i];

str[i\*2+3]='%';

}

len2=len1\*2+2;

// printf("%s",str);

}

void manacher(){//主要是说已经对称匹配过的不用再进行

int id=0,mx=0;

int i;

FOR(i,1,len2-1){

if (mx>i) p[i]=min(p[2\*id-i],mx-i);

else p[i]=1;

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

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

mx=p[i]+i;

id=i;

}

}

}

int main(){

int i;

while (~scanf("%s",s)){

len1=strlen(s);

init();

manacher();

REP(i,len2) ans=max(ans,p[i]);

printf("%d\n",ans-1);

}

}

# 数据结构

树状数组 区间max

LL a[N];int n;int i,j,k;

LL lowbit(LL x){return x&-x;}

/\*区间最大值\*/

LL m[N];

void change(LL r){

m[r]=a[r];

LL i,t=lowbit(r);

for (i=1;i<t;i<<=1) m[r]=max(m[r],m[r-i]);

}

void init(LL n){

LL i;

FOR(i,1,n) c[i]=0;

FOR(i,1,n) change(i);

}

void update(LL x){

LL i;

change(x);

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

}

LL getmax(LL l,LL r){

LL ret=a[r];

while (l!=r){

for (r--;r-lowbit(r)>=l;r-=lowbit(r)) ret=max(ret,m[r]);

ret=max(ret,a[r]);

}

return ret;

}

int main()

{

cin>>n;

FOR(i,1,n) cin>>a[i];

init(n);

FOR(i,1,n) cout<<m[i]<<' ';

cin>>n;

FOR(i,1,n){

cin>>j>>k;

printf("%lld\n",getmax(j,k));

}

}

树状数组 区间和

LL a[N];

int n,m;

int i,j,k;

LL lowbit(LL x){

return x&-x;

}

/\*区间和，单点修改\*/

LL c[N];

LL presum(LL x){

LL ret=0;

while (x){

ret+=c[x];

x-=lowbit(x);//可^=

}

return ret;

}

LL sum(LL l,LL r){

return presum(r)-presum(l-1);

}

void add(LL x,int d){//修改不如add有效

while (x<=n){

c[x]+=d;

x+=lowbit(x);

}

}

void init(LL n){

FOR(i,1,n) c[i]=0;

FOR(i,1,n) add(i,a[i]);

}

int main()

{

cin>>n;

FOR(i,1,n) cin>>a[i];

init(n);

FOR(i,1,n) cout<<c[i]<<' ';

cin>>n;

FOR(i,1,n){

cin>>j>>k;

printf("%lld\n",sum(j,k));

}

}

二维树状数组 区间修改单点查询

int n,m;

int c[maxn][maxn];

int lowbit(int x){return x&-x;}

void update(int x1,int y1){

int x=x1;

while (x<=n){

int y=y1;

while (y<=n){

c[x][y]^=1;

y+=lowbit(y);

}

x+=lowbit(x);

}

}

int sum(int x1,int y1){

int ret=0;

int x=x1;

while (x){

int y=y1;

while (y){

ret^=c[x][y];

y^=lowbit(y);

}

x^=lowbit(x);

}

return ret;

}

void init(){

int i,j;

FOR(i,1,n)

FOR(j,1,n) c[i][j]=0;

}

int T;

char s[10];

int i,j,k;

int x1,x2,y1,y2;

int main()

{

scanf("%d",&T);

while (T--){

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

init();

REP(i,m){

scanf("%s",s);

if (s[0]=='C'){

scanf("%d%d%d%d",&x1,&y1,&x2,&y2);

update(x1,y1);

update(x1,y2+1);

update(x2+1,y1);

update(x2+1,y2+1);

}

else {

scanf("%d%d",&x1,&y1);

printf("%d\n",sum(x1,y1));

}

}

puts("");

}

}

不大于k的最大值

int a[maxn];

int n,i,j;

const int nn=1000000;

inline int lowbit(int x){

return x&-x;

}

inline void insert(int x){

while (x<=nn){

a[x]++;

x+=lowbit(x);

}

}

inline int find(int x){

while (x&&!a[x]) x^=lowbit(x);

if (!x) return 0;

int t=lowbit(x)>>1,y=a[x];

while (t){

if (y-a[x-t]) y-=a[x-t];

else{y=a[x-t];x=x-t;}

t>>=1;

}

return x;

}

int ans;

const int MOD=19260817;

int main()

{

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

{

ans=0;

FOR(i,1,1000000) a[i]=0;

REP(i,n){

scanf("%d",&j);

if (j==0) continue;

ans=ans+find(j);

// printf("%d ",find(j));

insert(j);

ans%=MOD;

}

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

}

}

线段树

int a[maxn];

struct node{

int left,right;

}tree[maxn\*4];

LL sum[maxn\*4],lazy[maxn\*4];

void change(int x,int i){

sum[x]+=1ll\*(tree[x].right-tree[x].left+1)\*i;

lazy[x]+=i;

}

void pushup(int x){

sum[x]=sum[x<<1]+sum[x<<1|1];

}

void pushdown(int x){

if (lazy[x]){

change(x<<1,lazy[x]);

change(x<<1|1,lazy[x]);

lazy[x]=0;

}

}

void build(int x,int l,int r){

tree[x].left=l;tree[x].right=r;

sum[x]=lazy[x]=0;

if (l==r){

sum[x]=a[l];

return;

}

int mid=(l+r)/2;

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

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

pushup(x);

}

void update(int x,int l,int r,LL val){

int L=tree[x].left,R=tree[x].right;

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

change(x,val);

return;

}

pushdown(x);

int mid=(L+R)/2;

if (mid>=l) update(x<<1,l,r,val);

if (r>mid) update(x<<1|1,l,r,val);

pushup(x);

}

LL query(int x,int l,int r){

int L=tree[x].left,R=tree[x].right;

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

return sum[x];

}

pushdown(x);

int mid=(L+R)/2;

LL sum=0;

if (mid>=l) sum+=query(x<<1,l,r);

if (r>mid) sum+=query(x<<1|1,l,r);

pushup(x);

return sum;

}

最长连续子区间

struct node{

int left,right;

}tree[maxn\*4];

int lmax[maxn\*4],rmin[maxn\*4],len[maxn\*4];//free

int mark[maxn\*4];

inline void change(int x){

int &L=tree[x].left,&R=tree[x].right;

if (mark[x]==1){

lmax[x]=L-1; rmin[x]=R+1;

len[x]=0;

}else if (!mark[x]){

lmax[x]=R; rmin[x]=L;

len[x]=R-L+1;

}else{

len[x]=max(len[x<<1],len[x<<1|1]);

len[x]=max(len[x],lmax[x<<1|1]-rmin[x<<1]+1);

if (mark[x<<1|1]==0) rmin[x]=rmin[x<<1];

else rmin[x]=rmin[x<<1|1];

if (mark[x<<1]==0) lmax[x]=lmax[x<<1|1];

else lmax[x]=lmax[x<<1];

if (len[x]==0) mark[x]=1;

if (len[x]==R-L+1) mark[x]=0;

}

}

void pushdown(int x){

if (mark[x]==1){

// printf("-%d %d %d-",tree[x].left,tree[x].right,mark[x]);

mark[x<<1]=1;

mark[x<<1|1]=1;

change(x<<1);

change(x<<1|1);

}else if (mark[x]==0){

mark[x<<1]=0;

mark[x<<1|1]=0;

change(x<<1);

change(x<<1|1);

}

}

void build(int x,int l,int r){

tree[x].left=l;tree[x].right=r;

change(x);

if (l==r) return;

int mid=(l+r)/2;

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

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

}

void add(int x,int l,int r){

int L=tree[x].left,R=tree[x].right;

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

mark[x]=1;

change(x);

return;

}

pushdown(x);

int mid=(L+R)/2;

if (mid>=l) add(x<<1,l,r);

if (r>mid) add(x<<1|1,l,r);

mark[x]=-1;

change(x);

}

void del(int x,int l,int r){

int L=tree[x].left,R=tree[x].right;

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

mark[x]=0;

change(x);

return;

}

pushdown(x);

int mid=(L+R)/2;

if (mid>=l) del(x<<1,l,r);

if (r>mid) del(x<<1|1,l,r);

mark[x]=-1;

change(x);

}

暴力

int a[maxn];

struct node{

int left,right;

int m;

LL sm;

}tree[maxn\*4];

inline pushup(int x){

tree[x].sm=tree[x<<1].sm+tree[x<<1|1].sm;

tree[x].m=max(tree[x<<1].m,tree[x<<1|1].m);

}

void build(int x,int l,int r){

tree[x].left=l;tree[x].right=r;

if (l==r){

tree[x].sm=tree[x].m=a[l];

}else {

int mid=(l+r)/2;

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

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

pushup(x);

}

}

void mod(int x,int l,int r,int mm){//暴力

if (tree[x].m<mm) return;

int L=tree[x].left,R=tree[x].right;

if (L==R){

a[L]%=mm;

tree[x].sm=tree[x].m=a[L];

}else{

int mid=(L+R)/2;

if (l<=mid) mod(x<<1,l,r,mm);

if (mid<r) mod(x<<1|1,l,r,mm);

pushup(x);

}

}

LL query(int x,int l,int r){

int L=tree[x].left,R=tree[x].right;

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

return tree[x].sm;

}else{

int mid=(L+R)/2;

LL sum=0;

if (l<=mid) sum+=query(x<<1,l,r);

if (mid<r) sum+=query(x<<1|1,l,r);

return sum;

}

}

void change(int x,int pos,int nm){

int l=tree[x].left,r=tree[x].right;

if (l==r){

a[l]=nm;

tree[x].m=tree[x].sm=a[l];

}else{

int mid=(l+r)/2;

if (pos<=mid) change(x<<1,pos,nm);

else change(x<<1|1,pos,nm);

pushup(x);

}

}

二维线段树

struct Tnode{

int left,right;

}treeY[maxn\*4],treeX[maxn\*4];

bool mark[maxn\*4][maxn\*4];

int locx[maxn],locy[maxn];

void buildY(int x,int y,int yl,int yr){

treeY[y].left=yl;treeY[y].right=yr;

mark[x][y]=0;

if (yl==yr){

locy[yl]=y;

return;

}

int mid=(yl+yr)/2;

buildY(x,y<<1,yl,mid);

buildY(x,y<<1|1,mid+1,yr);

}

void buildX(int x,int n,int xl,int xr){

treeX[x].left=xl;treeX[x].right=xr;

if (xl==xr){

locx[xl]=x;

buildY(x,1,1,n);

return;

}

int mid=(xl+xr)/2;

buildX(x<<1,n,xl,mid);

buildX(x<<1|1,n,mid+1,xr);

buildY(x,1,1,n);

}

void updateY(int x,int y,int yl,int yr){

int L=treeY[y].left,R=treeY[y].right;

if (yl<=L&&R<=yr){

mark[x][y]^=1;

return;

}

int mid=(L+R)/2;

if (mid>=yl) updateY(x,y<<1,yl,yr);

if (yr>mid) updateY(x,y<<1|1,yl,yr);

}

void updateX(int x,int xl,int xr,int yl,int yr){

int L=treeX[x].left,R=treeX[x].right;

// printf("%d %d %d\n",x,L,R);

if (xl<=L&&R<=xr){

updateY(x,1,yl,yr);

return;

}

int mid=(L+R)/2;

if (mid>=xl) updateX(x<<1,xl,xr,yl,yr);

if (xr>mid) updateX(x<<1|1,xl,xr,yl,yr);

}

bool calc(int x,int y){

int ret=0,i,j;

for (i=locx[x];i;i>>=1)

for (j=locy[y];j;j>>=1) ret^=mark[i][j];

return ret;

}

扫描线 矩形周长并

int size;

int len[maxn\*2];

int n,m;

int i,j,k;

struct Seg{

struct node{

int left,right;

int len,num;

bool cl,cr;//iff

int lazy;

void update(int x){

lazy+=x;

}

}tree[maxn\*4];

void pushup(int x){

if (tree[x].lazy){

tree[x].len=len[tree[x].right+1]-len[tree[x].left];

tree[x].cl=tree[x].cr=1;tree[x].num=2;

}else if (tree[x].left==tree[x].right){

tree[x].len=0;

tree[x].cl=tree[x].cr=0;tree[x].num=0;

}else{

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

tree[x].num=tree[x<<1].num+tree[x<<1|1].num;

if (tree[x<<1].cr&&tree[x<<1|1].cl) tree[x].num-=2;

tree[x].cl=tree[x<<1].cl;

tree[x].cr=tree[x<<1|1].cr;

}

};

void build(int x,int l,int r){

tree[x].left=l;tree[x].right=r;

tree[x].len=tree[x].lazy=0;

if (l==r){

}else{

int mid=(l+r)/2;

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

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

pushup(x);

}

}

void update(int x,int l,int r,LL val){

int L=tree[x].left,R=tree[x].right;

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

tree[x].update(val);

pushup(x);

}else{

int mid=(L+R)/2;

if (mid>=l) update(x<<1,l,r,val);

if (r>mid) update(x<<1|1,l,r,val);

pushup(x);

}

}

int query(int x,int l,int r){//num

int L=tree[x].left,R=tree[x].right;

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

return tree[x].len;

}else{

int mid=(L+R)/2;

int ans;

if (mid>=l) ans+=query(x<<1,l,r);

if (r>mid) ans+=query(x<<1|1,l,r);

pushup(x);

return ans;

}

}

}T;

struct point{

int x1,x2,h;

int n;

bool operator <(const point&a)const{

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

return n>a.n;

}

}a[maxn];

map<int,int> hash;

int x1,x2,y1,y2;

int ans;

int len1,len2,num;

int main()

{

int TT=0;

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

if (n==0) break;

FOR(i,1,n){

scanf("%d%d%d%d",&x1,&y1,&x2,&y2);

len[i\*2-1]=x1; len[i\*2]=x2;

a[i\*2-1].x1=x1;a[i\*2-1].x2=x2;

a[i\*2-1].n=1; a[i\*2-1].h=y1;

a[i\*2].x1=x1;a[i\*2].x2=x2;

a[i\*2].n=-1; a[i\*2].h=y2;

}

sort(a+1,a+n\*2+1);

sort(len+1,len+n\*2+1);

hash.clear();

FOR(i,1,2\*n) hash[len[i]]=i;

T.build(1,1,n\*2);

ans=0;

FOR(i,1,2\*n){

len1=T.tree[1].len;num=T.tree[1].num;

T.update(1,hash[a[i].x1],hash[a[i].x2]-1,a[i].n);

len2=T.tree[1].len;

ans+=abs(len2-len1);

ans+=num\*(a[i].h-a[i-1].h);

}

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

}

}

主席树-区间第k大

vector<int> v;//学到的hash方法

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

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

struct Tnode{

int left,right,sum;

}T[maxn\*40];

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)/2;

if (mid>=pos) update(l,mid,T[x].left,T[y].left,pos);

else update(mid+1,r,T[x].right,T[y].right,pos);

}

int query(int l,int r,int x,int y,int k){

if (l==r) return l;

int mid=(l+r)/2;

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

if (sum>=k) return query(l,mid,T[x].left,T[y].left,k);

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

}

int n,m;

int i,j,k,ii;

int main()

{

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

FOR(i,1,n) scanf("%d",&a[i]),v.push\_back(a[i]);

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

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

REP(ii,m){

scanf("%d%d%d",&i,&j,&k);

printf("%d\n",v[query(1,n,root[i-1],root[j],k)-1]);

}

return 0;

}

往后多少位在哪里

int tot;

int n,i;

int a[maxn],root[maxn],tmp,cnt;

int last[maxn];//去重

struct node{

int left,right,sum;

}T[maxn\*40];

void update(int l,int r,int &x,int y,int pos,int v){

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

if (l==r) return;

int mid=(l+r)/2;

if (mid>=pos) update(l,mid,T[x].left,T[y].left,pos,v);

else update(mid+1,r,T[x].right,T[y].right,pos,v);

}

int query(int l,int r,int x,int k){//只用到了左端点...我是ZZ

if (l==r) return l;//这里return啥看情况

int mid=(l+r)/2;

int sum=T[T[x].left].sum;

if (sum>k) return query(l,mid,T[x].left,k);

else return query(mid+1,r,T[x].right,k-sum);

}

int ask(int i){

int t=1,ret=0;

while (t<=n){

t=query(1,n+1,root[t],i);

ret++;

}

return ret;

}

int main(){

scanf("%d",&n);

FOR(i,1,n) scanf("%d",&a[i]);

rFOR(i,1,n){//这里反着求的原因是上面要从前往后算第k大位置从前往后-1会少

if (!last[a[i]])

update(1,n+1,root[i],root[i+1],i,1);//n+1是为了超出

else{

update(1,n+1,tmp,root[i+1],last[a[i]],-1);

update(1,n+1,root[i],tmp,i,1);//tmp显然没用。。。root就行

}

last[a[i]]=i;

}

FOR(i,1,n) printf("%d ",ask(i));

}

区间不重复数字个数和第k个是哪位

int cnt;

struct node{

int l,r,sum;

}T[maxn\*40];

void update(int l,int r,int &x,int y,int pos,int v){

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

if (l==r) return;

int mid=(l+r)/2;

if (mid>=pos) update(l,mid,T[x].l,T[y].l,pos,v);

else update(mid+1,r,T[x].r,T[y].r,pos,v);

}

int findsum(int l,int r,int x,int L,int R){//每个点记录的都是这个点往后的相同数(前面把后面短路了)

if (L<=l&&r<=R) return T[x].sum;

int mid=(l+r)/2;

int sum=0;

if (mid>=L) sum+=findsum(l,mid,T[x].l,L,R);

if (R>mid) sum+=findsum(mid+1,r,T[x].r,L,R);

return sum;

}

int query(int l,int r,int x,int k){

if (l==r) return l;

int mid=(l+r)/2;

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

if (sum>=k) return query(l,mid,T[x].l,k);

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

}

int n,m;

int i,j,k,pos;

int t,TT;

int ans[maxn],a[maxn];

int last[maxn],root[maxn];

int main()

{

scanf("%d",&TT);

FOR(t,1,TT){

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

FOR(i,1,n) scanf("%d",&a[i]);

FOR(i,1,n) last[a[i]]=0,root[i]=0;

cnt=0;

rFOR(i,1,n){

if (!last[a[i]]) update(1,n,root[i],root[i+1],i,1);

else {

update(1,n,root[i],root[i+1],last[a[i]],-1);

update(1,n,root[i],root[i],i,1);

}

last[a[i]]=i;

}

FOR(i,1,m){

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

j=(j+ans[i-1])%n+1;

k=(k+ans[i-1])%n+1;

if (j>k) swap(j,k);

pos=(findsum(1,n,root[j],j,k)+1)/2;

ans[i]=query(1,n,root[j],pos);

}

printf("Case #%d:",t);

FOR(i,1,m) printf(" %d",ans[i]);

puts("");

}

return 0;

}

# 图论

二分图,匈牙利算法

int n,m,i,j,k,t;

vector<int>edge[N];

int used[N];

int matching[N];

/\*注意数组的标号，必须满足二分图的条件

bool dfs(int u){

int v,i;

REP(i,edge[u].size()){

v=edge[u][i];

if (!used[v]){

used[v]=1;

if (matching[v]==-1||dfs(matching[v])){

matching[v]=u;

matching[u]=v;

return 1;

}

}

}return 0;

}

int DFS(){

int ans=0;

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

int u;

FOR(u,1,n){

if (matching[u]==-1){

memset(used,0,sizeof(used));

if (dfs(u)) ans++;

}

}return ans;

}\*/

/\*注意数组的标号，必须满足二分图的条件

queue<int> Q;

int prev[N];//两格

int check[N];//matchright

int BFS(){

int ans=0;

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

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

FOR(i,1,n){

if (matching[i]==-1){

while (!Q.empty()) Q.pop();

Q.push(i);

prev[i]=-1;

bool flag=false;

while (!Q.empty()&&!flag){

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

for (j=0;!flag&&j<edge[u].size();j++){

int v=edge[u][j];

if (check[v]!=i){

check[v]=i;

Q.push(matching[v]);

if (matching[v]!=-1) prev[matching[v]]=u;

else{

flag=1;

int d=u,e=v;

while (d!=-1){

int t=matching[d];

matching[d]=e;

matching[e]=d;

d=prev[d];

e=t;

}

}

}

}

}

if (matching[i]!=-1) ans++;

}

}return ans;

}\*/

int main(){

int T;

scanf("%d",&T);

while (T--){

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

FOR(i,1,n){

scanf("%d",&k);

edge[i].clear();

REP(j,k) scanf("%d",&t),edge[i].push\_back(t+n);

}

if (BFS()==n) puts("YES");

else puts("NO");

}

}

最短路

Dijkstra（n^2）：

LL n,m,x;

LL a[N+2][N+2];

LL b[N+2];

bool vis[N+2];

LL i,j,k;

LL A,B,T;

int main()

{

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

FOR(i,n)

FOR(j,n) a[i][j]=INF;

FOR(i,m){

scanf("%lld%lld%lld",&A,&B,&T);

a[A][B]=T;

}

FOR(i,n) {b[i]=INF;vis[i]=0;}

b[0]=INF;

b[x]=0;

int pos;

FOR(i,n){

pos=0;

FOR(j,n) if (!vis[j]&&b[j]<b[pos]) pos=j;

vis[pos]=1;

FOR(j,n) if (!vis[j]&&b[pos]+a[pos][j]<b[j]) b[j]=b[pos]+a[pos][j];

}

FOR(i,n) printf("%lld ",b[i]);

}

Dijkstra（堆优化）：

struct node{

int n,d;

node(){}

node(int a,int b):n(a),d(b){}

bool operator<(const node&a)const{

if (d==a.d) return n<a.n;

return d>a.d;//注意！！！

}

};

**Dijkstra**

vector<node> edge[maxn];//注意这里priority\_queue是大根堆

int dis[maxn],n,m;

void dij(int s){//DIJKSTRA+HEAP

int i;

FOR(i,1,n) dis[i]=INF;

dis[s]=0;

priority\_queue<node> Q;

Q.push(node(s,dis[s]));

while (!Q.empty()){

node x=Q.top();Q.pop();

REP(i,edge[x.n].size()){

node y=edge[x.n][i];

if (dis[y.n]>x.d+y.d){

dis[y.n]=x.d+y.d;

Q.push(node(y.n,dis[y.n]));

}

}

}

}

**SPFA BFS**

vector<node> edge[maxn];

int dis[maxn],n,m;

bool vis[maxn];

int sumnum[maxn];//judge negative ring

bool spfa(int s){

int i;

FOR(i,1,n) dis[i]=INF;

FOR(i,1,n) vis[i]=0;

FOR(i,1,n) sumnum[i]=0;//judge negative ring

dis[s]=0;

deque<int> Q;//slf need

Q.push\_back(s);

// int sum=0;//lll

while (!Q.empty()){

int u=Q.front();Q.pop\_front();

// if (!Q.empty()&&sum/Q.size()<dis[u]) Q.push\_back(u);//lll

// else {vis[u]=0; sum-=dis[u];}//lll

vis[u]=0;//not lll

REP(i,edge[u].size()){

node v=edge[u][i];

if (dis[u]+v.d<dis[v.n]){

dis[v.n]=dis[u]+v.d;

if (!vis[v.n]){

vis[v.n]=1;

if (Q.empty()||dis[Q.front()]<dis[v.n]) Q.push\_back(v.n);//slf

else Q.push\_front(v.n);//slf

Q.push\_back(v.n);//not slf

// sumnum[v.n]++;//judge negative ring

// if (sumnum[v.n]>=n) return 1;//judge negative ring

// sum+=dis[v.n];//lll

}

}

}

}

// return 0;//judge negative ring

}

**SPFA DFS(只用于判负环)**

vector<node> edge[maxn];

int dis[maxn],n,m;

bool vis[maxn];

bool spfa(int u){

int i;

vis[u]=1;

REP(i,edge[u].size()){

node v=edge[u][i];

if (dis[u]+v.d<dis[v.n]){

dis[v.n]=dis[u]+v.d;

if (vis[v.n]) return 1;

else {

dis[v.n]=dis[u]+v.d;

if (spfa(v.n)) return 1;

}

}

}

vis[u]=0;

return 0;//judge negative ring

}

int s,t;

int u,v,len;

int main(){

int i,j,k;

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

FOR(i,1,n) edge[i].clear();

REP(i,m){

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

edge[u].push\_back(node(v,len));

edge[v].push\_back(node(u,len));

}

dij(1);

FOR(i,2,n) printf("%d ",dis[i]==INF?-1:dis[i]);

puts("");

}

return 0;

}

网络流

int n,m;

int i,j,x;

int from,to;

int u,v,flow;

struct Edge{

int from,to,cap,flow;

};

vector<Edge> edge;

vector<int> G[maxm];

int vis[maxn];

int d[maxn];

int tot=0;

void addflow(int from,int to,int cap){

edge.push\_back((Edge){from,to,cap,0}); G[from].push\_back(tot); tot++;

edge.push\_back((Edge){to,from,0,0}); G[to].push\_back(tot); tot++;

}

int s,t;

bool bfs(){

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

queue<int> Q;

Q.push(s);

d[s]=0;

vis[s]=1;

while (!Q.empty()){

int x=Q.front();

Q.pop();

REP(i,G[x].size()){

Edge &e=edge[G[x][i]];

if (!vis[e.to]&&e.cap>e.flow){

vis[e.to]=1;

d[e.to]=d[x]+1;

Q.push(e.to);

}

}

}

return vis[t];

}

int dfs(int x,int a){

int i;

if (x==t||a==0) return a;

int flow=0,f;

REP(i,G[x].size()){

Edge &e=edge[G[x][i]];

if (d[x]+1==d[e.to]&&(f=dfs(e.to,min(a,e.cap-e.flow)))>0){

e.flow+=f;

edge[G[x][i]^1].flow-=f;

flow+=f;

a-=f;

if (a==0) break;

}

}

return flow;

}

void solve(int t,int m){

int n,i;

s=1; tot=0;

FOR(i,1,t) G[i].clear();

edge.clear();

REP(i,m){

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

addflow(u,v,flow);

}

int flow=0;

bool mark;

while (bfs()) flow+=dfs(s,INF);

REP(i,tot)

if (i&1) edge[i].cap=0;

else edge[i].cap=1,edge[i].flow=0;

if (bfs()) printf("%d\n",flow);

else puts("404 Not Found");

}

int main()

{

while (~scanf("%d%d",&t,&m)){solve(t,m);}

}

上下界网络流

//可二分t->s边的下/上界,即可达到最大最小流

//最大流:t->s连边,ss->tt流,s->t正向最大流,会流掉反向建的边的流量

//最小流:ss->tt流,t->s连边,ss->tt流

int n,m,q;

int i,j,k;

int ss,tt;

struct node{

int to,cap,next;

}edge[maxn\*3];

int tot;

int head[307];

int addedge(int from,int to,int cap){

edge[tot].to=to;

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

edge[tot].cap=cap;

head[from]=tot++;

edge[tot].to=from;

edge[tot].next=head[to];

edge[tot].cap=0;

head[to]=tot++;

return tot-1;//反的边 cap=正的 flow

}

bool vis[307];

int d[307];

queue<int> Q;

bool bfs(int s,int t){

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

while (Q.size()) Q.pop();

Q.push(s);

d[s]=0;vis[s]=1;

int i;

while (Q.size()){

int x=Q.front();Q.pop();

for (i=head[x];i!=-1;i=edge[i].next){

if (!vis[edge[i].to]&&edge[i].cap){

vis[edge[i].to]=1;

d[edge[i].to]=d[x]+1;

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

}

}

}

return vis[t];

}

int cur[307];//当前弧优化

int dfs(int x,int t,int flow){//dinic

if (x==t||flow==0) return flow;

int i,ret=0,f;

for (i=cur[x];i!=-1;i=edge[i].next){

if (d[x]+1==d[edge[i].to]&&(f=dfs(edge[i].to,t,min(flow,edge[i].cap)))>0){

edge[i].cap-=f;

edge[i^1].cap+=f;

ret+=f;

flow-=f;

cur[x]=i;

if (flow==0) break;

}

}

return ret;

}

int in[307],out[307];

int add(int u,int v,int low,int high){

int ret=addedge(u,v,high-low);

out[u]+=low;in[v]+=low;

return ret;

}

int sum,flow,E[maxn],ans[maxn];//E为对应的边位置

int solve(){

memset(head,0xff,sizeof(head));

memset(in,0,sizeof(in));

memset(out,0,sizeof(out));

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

flow=0;sum=0;tot=0;

FOR(i,1,m){

int u,v,low,high;

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

ans[i]=low;

E[i]=add(u,v,low,high);//E[i]很有用

}

ss=n+1;tt=n+2;

FOR(i,1,n){

sum+=max(in[i]-out[i],0);

if (in[i]>out[i]) addedge(ss,i,in[i]-out[i]);

if (in[i]<out[i]) addedge(i,tt,out[i]-in[i]);

}

while (bfs(ss,tt)){

int f;

memcpy(cur,head,sizeof(head));

while (f=dfs(ss,tt,INF)) flow+=f;

}

if (flow!=sum) return 0\*puts("NO");

else {

puts("YES");

FOR(i,1,m){

ans[i]+=edge[E[i]].cap;

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

}

}

}

int main()

{

int T;

scanf("%d",&T);

while (T--){

solve();

}

}

强连通分量tarjan

vector<int> E[maxn];

int dfn[maxn],low[maxn],tot,n,ans=INF,cnt;

bool vis[maxn];

stack<int> S;

vector<int> V[maxn];

//u割点:lowlink[u]>=dfn[v];

//uv割边:lowlink[u]>dfn[v];

//块:lowlink[u]==dfn[v];

void tarjin(int x){

low[x]=dfn[x]=++tot;

S.push(x);vis[i]=1;

for (int i=0;i<E[x].size();i++){

int v=E[x][i];

if (!dfn[v]){

tarjin(x);

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

}else if (vis[v]){

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

}

}

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

cnt++;

while (1){

int now=S.top();

vis[now]=0;

V[cnt].push\_back(now);

if (now==x) break;

}

}

}

2-sat

struct Tsat{

vector<int> edge[maxn\*2];

stack<int> S;

int belong[maxn\*2];

int dfn[maxn\*2],low[maxn\*2];

bool vis[maxn\*2];

int tot,cnt;

bool mark;

void init(int n){

tot=cnt=0;

int i;

REP(i,n\*2) edge[i].clear();

REP(i,n\*2) dfn[i]=vis[i]=low[i]=belong[i]=0;

}

void dfs(int u){

int i;

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

S.push(u);vis[u]=1;

REP(i,edge[u].size()){

int v=edge[u][i];

if (!dfn[v]){

dfs(v);

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

}else if (vis[v]){

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

}

}

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

cnt++;

while (1){

int now=S.top();S.pop();

vis[now]=0;

belong[now]=cnt;

if (now==u) break;

}

}

}

inline void addedge(int u,int v){

edge[u].push\_back(v);

}

bool solve(int n){

int i;

REP(i,n\*2) if (!dfn[i]) dfs(i);

REP(i,n) if (belong[i]==belong[i+n]) return 0;

return 1;

}

}sat;

int n,m,t;

int numA,numB;

int A[maxn][2],B[maxn][2];

int i,j;

int tot;

struct node{

int x,y;

}S1,S2,a[maxn];

inline int dist(node A,node B){

return abs(A.x-B.x)+abs(A.y-B.y);

}

void preadd(){

int i,u,v;

REP(i,numA){

u=A[i][0];v=A[i][1];

sat.addedge(u,v+n);sat.addedge(u+n,v);

sat.addedge(v,u+n);sat.addedge(v+n,u);

}

REP(i,numB){

u=B[i][0];v=B[i][1];

sat.addedge(u,v);sat.addedge(u+n,v+n);

sat.addedge(v,u);sat.addedge(v+n,u+n);

}

}

bool solve(int x){

sat.init(n);

preadd();

int i,j;

REP(i,n)

rep(j,i+1,n){

if (dist(a[i],S1)+dist(a[j],S1)>x) {sat.addedge(i,j+n);sat.addedge(j,i+n);}

if (dist(a[i],S2)+dist(a[j],S2)>x) {sat.addedge(i+n,j);sat.addedge(j+n,i);}

if (dist(a[i],S1)+dist(a[j],S2)+dist(S1,S2)>x) {sat.addedge(i,j);sat.addedge(j+n,i+n);}

if (dist(a[i],S2)+dist(a[j],S1)+dist(S1,S2)>x) {sat.addedge(i+n,j+n);sat.addedge(j,i);}

}

return sat.solve(n);

}

int l,r,mid;

int main(){

int t,m;

while (~scanf("%d%d%d",&n,&numA,&numB)){

scanf("%d%d%d%d",&S1.x,&S1.y,&S2.x,&S2.y);

// printf("%d\n",dist(S1,S2));

REP(i,n) scanf("%d%d",&a[i].x,&a[i].y);

REP(i,numA) {scanf("%d%d",&A[i][0],&A[i][1]);A[i][0]--;A[i][1]--;}/\*careful!!!\*/

REP(i,numB) {scanf("%d%d",&B[i][0],&B[i][1]);B[i][0]--;B[i][1]--;}/\*careful!!!\*/

l=-1;r=5000000;

while (l+1<r){

mid=(r+l)/2;

if (!solve(mid)) l=mid;

else r=mid;

// printf("%d %d\n",mid,solve(mid));

}

if (l<4500000) printf("%d\n",l+1);

else printf("-1\n");

}

}

求凸包

struct node{

double x,y;

bool operator <(const node &a) const{

if (y<a.y) return 1; if (y>a.y) return 0;

return x<a.x;

}

}p[maxn],P[maxn];

inline double X(node A,node B,node C){ return (B.x-A.x)\*(C.y-A.y)-(B.y-A.y)\*(C.x-A.x); }

inline double len(node A,node B){ return sqrt((A.x-B.x)\*(A.x-B.x)+(A.y-B.y)\*(A.y-B.y)); }

bool cmp(node A,node B){

double cp=X(p[0],A,B);

if (cp>0) return 1;if (cp<0) return 0;

return len(p[0],A)<len(p[0],B);

}

int n,m;

double t;

int tot;

int i,j,k;

double ans;//求长度的

int main(){

while (~scanf("%d%lf",&n,&t)){

REP(i,n) scanf("%lf%lf",&p[i].x,&p[i].y);

// ans=2\*pi\*t;//没啥用//=0

if (n==1) printf("%.0lf",ans);

else if (n==2) printf("%.0lf",ans+len(p[0],p[1]));

else {

REP(i,n) if (p[i]<p[0]) swap(p[0],p[i]);

sort(p+1,p+n,cmp);

P[0]=p[0];

P[1]=p[1];

tot=1;

rep(i,2,n){

while (tot>0&&X(P[tot-1],P[tot],p[i])<=0) tot--;

P[++tot]=p[i];

}

REP(i,tot) ans+=len(P[i],P[i+1]);

ans+=len(P[0],P[tot]);

printf("%.0lf",ans);

}puts("");

}

}

# 数学相关

void getPrim(){//线性的筛法求素数

int o=0;

register int i,j;

FOR(i,2,Nmax){

if (!prim[i]) prim[++prim[0]]=i;

FOR(j,1,prim[0]){

if (i\*prim[j]>Nmax) break;

prim[i\*prim[j]]=1;

if (i%prim[j]==0) break;

}

}

}

逆元

int n,m;

int i,j,k;

//d==1时存在逆元 //(x+p)%p为逆元//d!=1可用num\*a/d来代替逆元(num|d)

void exgcd(LL a,LL b,LL &d,LL &x,LL &y){

if (!b) {d=a;x=1;y=0;}

else {exgcd(b,a%b,d,y,x);y-=a/b\*x;}

}

int getinv(int n){

if (n==1) return 1;

return (M-M/n)\*(getinv(M%n))%M;

}

LL inv1[1000002];

LL inv2[1000002];

LL inv3[1000002];

int main()

{

LL d,x,y;

// FOR(i,1,1000000) {exgcd(i,M,d,inv[i],y); inv1[i]=(inv[i]+M)%M;}

// FOR(i,1,1000000) inv2[i]=getinv(i);

inv3[0]=inv3[1]=1;

FOR(i,2,1000000) inv3[i]=(M-M/i)\*inv3[M%i]%M;

// FOR(i,1,1000000) printf("%lld ",inv3[i]\*i%M);

}

C(n,n)

int n,m;

int i,j,k;

LL inv[1000002];//inverse

LL fac[1000002];//Factorial

void init(){

int i;

fac[0]=1;

FOR(i,1,1000000) fac[i]=i\*fac[i-1]%M;

inv[0]=inv[1]=1;

FOR(i,2,1000000) inv[i]=(M-M/i)\*inv[M%i]%M;

FOR(i,1,1000000) inv[i]=inv[i]\*inv[i-1]%M;

}

LL C(int n,int m){

return fac[n]\*inv[m]%M\*inv[n-m]%M; }

int main()

{

LL d,x,y;

init();

printf("%d",C(10,3));

}

Lucas Cnn

int n,m;

int i,j,k;

LL inv[1000002];//inverse

LL fac[1000002];//Factorial

void init(){

int i;

fac[0]=1;

FOR(i,1,1000000) fac[i]=i\*fac[i-1]%MOD;

inv[0]=inv[1]=1;

FOR(i,2,1000000) inv[i]=(MOD-MOD/i)\*inv[MOD%i]%MOD;

FOR(i,1,1000000) inv[i]=inv[i]\*inv[i-1]%MOD;

}

LL C(int n,int m){

return fac[n]\*inv[m]%MOD\*inv[n-m]%MOD;

}

LL lucas(LL n,LL m){//注意MOD不能太大=\_=!

return m==0?1:1ll\*C(n%MOD,m%MOD)\*lucas(n/MOD,m/MOD)%MOD;

}

int main()

{

LL d,x,y;

init();

printf("%d",lucas(10,3));

}

数位dp

对于某一个问题，f[i][j][k][l]表示i位,第一位j,k=0/1(表示是否满足条件),余数或者其他为l时的情况个数

LL n,m;

LL dp[20][3];//0:

LL i,j,k;

void init(){

memset(dp,0,sizeof(dp));

dp[0][0]=1;

FOR(i,1,10){

dp[i][0]=dp[i-1][0]\*9-dp[i-1][1];//okay

dp[i][1]=dp[i-1][0];//2......

dp[i][2]=dp[i-1][0]+dp[i-1][1]+dp[i-1][2]\*10;//not okay

}

}

int A[20];

int calc(int a){

int sum=a;

int m=0;

int ans=0;

bool flag=false;

while(a){

A[++m]=a%10;

a/=10;

}

A[m+1]=0;

for (int i=m;i>=1;i--){

ans+=dp[i-1][2]\*A[i];

if (flag){

ans+=dp[i-1][0]\*A[i];

}else{

if (A[i]>4) ans+=dp[i-1][0];

if (A[i+1]==6&&A[i]>2) ans+=dp[i][1];

if (A[i]>6) ans+=dp[i-1][1];

if (A[i]==4||A[i]==2&&A[i+1]==6) flag=1;

}

}

if (flag) ans++;

return sum-ans;

}

int main(){

int a,b;

int l,r;

init();

while (~scanf("%d%d",&l,&r)&&(l||r)) printf("%d\n",calc(r)-calc(l-1));

}

LL n,m;

LL dp[25][3];

LL i,j,k;

void init(){

memset(dp,0,sizeof(dp));

dp[0][0]=1;

FOR(i,1,25){

dp[i][0]=dp[i-1][0]\*10-dp[i-1][1];//okay(有1的)

dp[i][1]=dp[i-1][0];//9......

dp[i][2]=dp[i-1][1]+dp[i-1][2]\*10;//not okay

}

}

int A[25];

LL calc(LL a){

int m=0;

LL ans=0;

bool flag=false;

while(a){

A[++m]=a%10;

a/=10;

}

A[m+1]=0;

for (int i=m;i>=1;i--){

ans+=dp[i-1][2]\*A[i];

if (flag){

ans+=dp[i-1][0]\*A[i];

}else{

if (A[i]>4) ans+=dp[i-1][1];

// if (A[i+1]==4&&A[i]>9) ans+=dp[i][1];

if (A[i+1]==4&&A[i]==9) flag=1;

}

}

if (flag) ans++;

return ans;

}

int main(){

LL l,r;

init();

scanf("%d",&n);

while (~scanf("%lld",&r)) printf("%lld\n",calc(r));

}

博弈：NIM,SG

选择的最多次数,main中为异或!=0

int sg[maxm+2];//打表~~~

/\*这个是状态和剩余个数有关的

map<int,int> Hash;

int SG(int mask){

if (Hash.count(mask)) return Hash[mask];

set<int> mex;

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

if (!((mask>>i)&1)) continue;//continue

int tp=mask;

for (int j=i;j<maxm;j+=i+1)//change

if ((mask>>j)&1) tp^=1<<j;

mex.insert(SG(tp));//dfs

}

int ret=0;

for (;mex.count(ret);++ret);

return Hash[mask]=ret;

}\*/

/\*这个是状态和剩余个数无关的

map<LL,int> Hash[62];

int SG(int x,LL mask){

// printf("%d %d\n",x,mask);

if (Hash[x].count(mask)) return Hash[x][mask];

set<int> mex;

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

if ((mask>>(i-1))&1) continue;//continue

int tp=mask;

tp^=1<<(i-1);//change

mex.insert(SG(x-i,tp));//dfs

}

int ret=0;

for (;mex.count(ret);++ret);

return Hash[x][mask]=ret;

}\*/

int main(){

sg[0]=0;

// FOR(i,1,maxm) printf("%d,",sg[i]=SG(i,0));

}

莫队

struct node{int l,r,id;}Q[maxn];//new direction

int pos[maxn];

LL ans[maxn],flag[maxn];

int a[maxn];

bool cmp(node a,node b){

if (pos[a.l]==pos[b.l]) return a.r<b.r;

return pos[a.l]<pos[b.l];

}

int n,m,k; int i,j;

LL Ans;

int L=1,R=0;

void add(int x){

Ans+=flag[a[x]^k];

flag[a[x]]++; }

void del(int x){

flag[a[x]]--;

Ans-=flag[a[x]^k]; }

int main(){

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

int sz=sqrt(n);

FOR(i,1,n){

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

a[i]^=a[i-1];

pos[i]=i/sz;

}

FOR(i,1,m){

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

Q[i].id=i;

}

sort(Q+1,Q+1+m,cmp);

flag[0]=1;

FOR(i,1,m){

while (L<Q[i].l){del(L-1);L++;}

while (L>Q[i].l){L--;add(L-1);}

while (R<Q[i].r){R++;add(R);}

while (R>Q[i].r){del(R);R--;}

ans[Q[i].id]=Ans;

}

FOR(i,1,m) printf("%I64d\n",ans[i]);

}