A:

题意：给出一组数，选若干个数，每个数的距离大于K，求和的最大值

/\*

分别统计以每个数为结尾时的和的最大值，由前面距离大于K的最大值加本身得到，最后取最大值即为结果

\*/

#include<bits/stdc++.h>

using namespace std;

const int maxn=1e5+9;

int arr[maxn];

int main(){

int t,n,k;

ios::sync\_with\_stdio(0);

cin>>t;

while(t--){

cin>>n>>k;

int maxinum=0;

int ans=0;

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

cin>>arr[i];

if(i<=k+1){

ans=max(ans,arr[i]);

}

else{

maxinum=max(maxinum,arr[i-k-1]);

arr[i]+=maxinum;

ans=max(ans,arr[i]);

}

}

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

}

return 0;

}

B:

题意：在一个环中，每次走i步的概率已知，求从一点到另一点的期望

/\*

F(i)=sum(1-m)p[i+j]\*(f[i+j]+j),然后利用高斯消元求解

\*/

#include<bits/stdc++.h>

using namespace std;

#define MAXN 109

#define EPS 1e-8

int n,m,row;

bool f[MAXN+10];

double a[MAXN+10][MAXN+10],x[MAXN+10];

double re;

void gauss\_jordan(){

int col,i,mxr,j;

for(row=col=1;row<=n&&col<=m;row++,col++){

mxr=row;

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

if(fabs(a[i][col])>fabs(a[mxr][col]))

mxr=i;

if(mxr!=row)

swap(a[row],a[mxr]);

if(fabs(a[row][col])<EPS){

row--;

continue;

}

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

if(i!=row&&fabs(a[i][col])>EPS)

for(j=m+1;j>=col;j--)

a[i][j]-=a[row][j]/a[row][col]\*a[i][col];

}

row--;

}

int pos;

int gcd(int a,int b){

int c;

while(b){

c=a%b;

a=b;

b=c;

}

return a;

}

bool print(){

int i,cnt,ff,j;

if(row==n&&m==n){

re=a[pos][m+1]/a[pos][pos]+EPS;

return 1;

}

return 0;

}

double arr[109];

int main(){

int t,nn,x,y,mm;

cin>>t;

while(t--){

cin>>nn>>mm>>x>>y;

double su=0.0;

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

cin>>arr[i];

arr[i]/=100.0;

su+=arr[i]\*i;

}

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

for(int j=1;j<=nn;j++){

a[i][j]=0.0;

}

if(i==y+1){

a[i][y+1]=1.0;

a[i][nn+1]=0.0;

}

else{

a[i][nn+1]=su;

a[i][i]=1.0;

for(int j=i,ii=1;ii<=mm;ii++,j++){

j%=nn;

a[i][j+1]=-arr[ii];

}

}

}

n=m=nn;

gauss\_jordan();

pos=x+1;

if(print()){

cout<<fixed<<setprecision(2)<<re<<endl;

}

else{

int g=0;

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

if(arr[i]<EPS){

continue;

}

if(!g)g=i;

else g=gcd(g,i);

}

if((nn+y-x)%g){

cout<<-1<<endl;

continue;

}

int tx=(x+1)/g+((x+1)%g>0),ty=(y+1)/g+((y+1)%g>0);

for(int i=1;i<=nn/g;i++){

for(int j=1;j<=nn/g;j++){

a[i][j]=0.0;

}

if(i==ty){

a[i][i]=1.0;

a[i][nn/g+1]=0.0;

}

else{

a[i][nn/g+1]=su;

a[i][i]=1.0;

for(int j=i,ii=g;ii<=mm;ii+=g,j++){

j%=nn/g;

a[i][j+1]=-arr[ii];

}

}

}

n=m=nn/g;

gauss\_jordan();

pos=tx;

if(print()){

cout<<fixed<<setprecision(2)<<re<<endl;

}

else{

cout<<-1<<endl;

}

}

}

}

C:

题意：求每个点的与之距离不超过6的点的数目

/\*

先统计子节点，然后再统计根结点，每次记录与当前距离为i的点的数目，然后求和即可

\*/

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

const int maxn=1e5+9;

int num[maxn][7];

struct edge{

int to,next;

}edges[maxn<<1];

int no;

int head[maxn];

void addedge(int from,int to){

edges[no].to=to;

edges[no].next=head[from];

head[from]=no++;

}

void bfs(int o,int fa){

for(int i=head[o];~i;i=edges[i].next){

int to=edges[i].to;

if(to==fa)continue;

bfs(to,o);

for(int j=1;j<=6;j++){

num[o][j]+=num[to][j-1];

}

}

num[o][0]=1;

}

int n;

int res[maxn];

void sol(int o,int fa,int arr[]){

int tmp[7]={0};

for(int i=1;i<7;i++){

res[o]+=arr[i];

res[o]+=num[o][i];

}

for(int i=head[o];~i;i=edges[i].next){

int to=edges[i].to;

if(to==fa)continue;

for(int j=1;j<7;j++){

tmp[j]=arr[j-1];

if(j>1){

tmp[j]+=num[o][j-1]-num[to][j-2];

}

else{

tmp[j]+=1;

}

}

sol(to,o,tmp);

}

}

int main(){

int t,e,f;

ios::sync\_with\_stdio(0);

cin>>t;

for(int ii=1;ii<=t;ii++){

cin>>n;

no=0;

memset(num,0,sizeof(num));

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

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

cin>>e>>f;

addedge(e,f);

addedge(f,e);

}

bfs(1,-1);

memset(res,0,sizeof(res));

int arr[7]={0};

sol(1,-1,arr);

cout<<"Case #"<<ii<<":"<<endl;

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

cout<<res[i]<<endl;

}

}

}

D:

/\*

1+Fib(1)+fib(2)+...+fib(3)+...+fib(n-1)+1=fib(2)+fib(3)+fib(3)+...+fib(n-1)=fib(3)+fib(4)+fib(4)+...+fib(n-1)=...=fib(n-1)+fib(n)=fib(n+1)

Fib(n+1)+fib(n)=fib(n+2)

Gcd(fib(n),fib(m))=fib(gcd(n,m))

然后利用矩阵快速幂计算

Fib(n+1) [1,1] fib(n)

Fib(n) [1,0] fib(n-1)

\*/

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

ll n,m,p;

struct mat{

ll arr[2][2];

};

mat operator\*(mat a,mat b){

mat c;

for(int i=0;i<2;i++){

for(int j=0;j<2;j++){

c.arr[i][j]=0;

for(int k=0;k<2;k++){

c.arr[i][j]+=a.arr[i][k]\*b.arr[k][j];

c.arr[i][j]%=p;

}

}

}

return c;

}

ll gcd(ll a,ll b){

ll c;

while(b){

c=a%b;

a=b;

b=c;

}

return a;

}

mat quick(mat a,ll b){

mat c;

memset(c.arr,0,sizeof(c.arr));

c.arr[0][0]=c.arr[1][1]=1;

while(b){

if(b&1){

c=c\*a;

}

a=a\*a;

b>>=1;

}

return c;

}

ll fib(ll a){

if(a<3){

return 1;

}

mat c;

c.arr[0][0]=1;

c.arr[0][1]=1;

c.arr[1][0]=1;

c.arr[1][1]=0;

c=quick(c,a-2);

return (c.arr[0][1]+c.arr[0][0])%p;

}

int main(){

int t;

ios::sync\_with\_stdio(0);

cin>>t;

while(t--){

cin>>n>>m>>p;

n+=2,m+=2;

n=gcd(n,m);

cout<<fib(n)<<endl;

}

}

E:

题意：完全二叉树每个节点有一个权值，求在权值和不超过k的情况下，最多可以取多少个结点，取得结点需每个结点是另一个的父节点

#include<bits/stdc++.h>

using namespace std;

const int maxn=1e6+9;

int w[maxn];

int su[maxn];

int val[maxn];

int main(){

int t,n,k;

ios::sync\_with\_stdio(0);

cin>>t;

while(t--){

cin>>n>>k;

int ans=-1;

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

cin>>w[i];

if(w[i]>k){

val[i]=su[i]=0;

continue;

}

else{

su[i]=su[i/2]+w[i];

val[i]=val[i/2]+1;

while(su[i]>k){

val[i]--;

su[i]-=w[i>>(val[i])];

}

ans=max(ans,val[i]);

}

}

cout<<ans<<endl;

}

}

F:

题意：求电磁波经若干次反射后从入射点出去所需的反射次数

/\*

不断求出每次反射后的点即可

\*/

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

typedef long double ld;

#define double long double

double aa,bb;

const double eps = 1e-8;

const double PI = acos(-1.0);

int sgn(double x) //判断正负数

{

if(fabs(x) < eps)return 0;

if(x < 0)return -1;

else return 1;

}

struct POint

{

double x,y;

POint(){}

POint(double \_x,double \_y)

{

x = \_x;y = \_y;

}

POint operator -(const POint &b)const

{

return POint(x - b.x,y - b.y);

}

double operator ^(const POint &b)const //叉积

{

return x\*b.y - y\*b.x;

}

double operator \*(const POint &b)const //点积

{

return x\*b.x + y\*b.y;

}

void transXY(double B,POint a) //绕原点旋转角度B（弧度值），后x,y的变化

{

double tx = x-a.x,ty = y-a.y;

x = tx\*cos(B) - ty\*sin(B)+a.x;

y = tx\*sin(B) + ty\*cos(B)+a.y;

}

};

double dist(POint a,POint b) //\*两点间距离

{

return sqrt((a-b)\*(a-b));

}

struct LIne

{

POint s,e;

LIne(){}

LIne(POint \_s,POint \_e)

{

s = \_s;e = \_e;

}

};

POint PointToLine(POint P,LIne L) //点到直线距离,返回为result,是点到直线最近的点

{

POint result;

double t = ((P-L.s)\*(L.e-L.s))/((L.e-L.s)\*(L.e-L.s));

result.x = L.s.x + (L.e.x-L.s.x)\*t;

result.y = L.s.y + (L.e.y-L.s.y)\*t;

return result;

}

vector<POint> jie(POint p0,POint p1){

vector<POint> ans;

double tb=p0.y-p1.y;

double tc=p0.x\*p1.y-p0.y\*p1.x;

double td=aa\*aa/pow(p0.x-p1.x,2.0);

double tta=bb\*bb+td\*tb\*tb;

double ttb=2.0\*td\*tc\*tb;

double ttc=td\*tc\*tc-aa\*aa\*bb\*bb;

POint tm;

tm.x=(-ttb-sqrt(ttb\*ttb-4.0\*tta\*ttc))/tta\*.5;

tm.y=(p0.x\*p1.y-p1.x\*p0.y+(p0.y-p1.y)\*tm.x)/(p0.x-p1.x);

if(tm.y>0){

tm.y=sqrt(max((1.0-tm.x\*tm.x/aa/aa)\*bb\*bb,(ld)0.0));

}

else{

tm.y=-sqrt(max((1.0-tm.x\*tm.x/aa/aa)\*bb\*bb,(ld)0.0));

}

ans.push\_back(tm);

tm.x=(-ttb+sqrt(ttb\*ttb-4.0\*tta\*ttc))/tta\*.5;

tm.y=(p0.x\*p1.y-p1.x\*p0.y+(p0.y-p1.y)\*tm.x)/(p0.x-p1.x);

if(tm.y>0){

tm.y=sqrt(max((1.0-tm.x\*tm.x/aa/aa)\*bb\*bb,(ld)0.0));

}

else{

tm.y=-sqrt(max((1.0-tm.x\*tm.x/aa/aa)\*bb\*bb,(ld)0.0));

}

ans.push\_back(tm);

return ans;

}

int main(){

int t;

ios::sync\_with\_stdio(0);

cin>>t;

while(t--){

cin>>aa>>bb;

int ans=0;

POint a1(-aa\*0.5,bb\*sqrt(3.0)\*.5),a2(aa\*.5,bb\*sqrt(3.0)\*.5),en,qie,a3;

en=a2;

while(dist(a1,en)>0.01-eps){

ans++;

if(sgn(a1.y)==0){

qie.x=a1.x;

qie.y=a1.y-1.0;

}

else{

qie.x=a1.x-1.0;

qie.y=(1.0-a1.x\*qie.x/aa/aa)\*bb\*bb/a1.y;

}

qie.transXY(PI\*.5,a1);

LIne l1(a1,qie);

qie=PointToLine(a2,l1);

a3.x=2.0\*qie.x-a2.x;

a3.y=2.0\*qie.y-a2.y;

vector<POint>vec = jie(a1,a3);

if(dist(vec[0],a1)<dist(vec[1],a1)){

a1=vec[1];

a2=vec[0];

}

else{

a1=vec[0];

a2=vec[1];

}

}

cout<<ans<<endl;

}

return 0;

}

G:

题意：找到序列中满足i<j&&ai<=aj中j-i最大值

/\*

按序列中的值由最小到最大排列，再统计之前出现最小的位置值，两者先减即为以当前位j时的最大值，然后取最大即可

\*/

#include<bits/stdc++.h>

using namespace std;

const int maxn=1e5+9;

struct num{

int val,pos;

}arr[maxn];

bool cmp(num a,num b){

if(a.val!=b.val){

return a.val<b.val;

}

return a.pos<b.pos;

}

int main(){

int n,l;

ios::sync\_with\_stdio(0);

while(cin>>n){

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

arr[i].pos=i;

cin>>arr[i].val;

}

sort(arr,arr+n,cmp);

l=arr[0].pos;

int ans=0;

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

if(arr[i].pos<l){

l=arr[i].pos;

}

else{

ans=max(ans,arr[i].pos-l);

}

}

cout<<ans<<endl;

}

return 0;

}

H:

题意：求连续区间的最大异或值的最左边与最右边的标号，相同取字典序最小的

/\*

用字典树记录前面异或值的和，查询时查找与该位值不同的节点是否出现过，若出现，则访问与当前位值不同的点，否则，访问另一节点

\*/

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

const int maxn=2e6+9;

struct node{

int nex[2];

}dic[maxn];

int no;

int ans,l,r;

void add(int root,int val,int ceng,int noi){

int bi=val&(1<<ceng);

bi=bi>0;

if(ceng==0){

if(dic[root].nex[bi]==-1){

dic[root].nex[bi]=noi;

}

return;

}

if(dic[root].nex[bi]==-1){

dic[root].nex[bi]=no++;

dic[no-1].nex[0]=dic[no-1].nex[1]=-1;

}

add(dic[root].nex[bi],val,ceng-1,noi);

}

void query(int root,int val,int ceng,int xorval,int noi){

int bi=val&(1<<ceng);

bi=bi>0;

bi^=1;

if(ceng==0){

if(dic[root].nex[bi]>-1){

xorval+=1;

}

if(xorval>=ans){

int tl;

if(dic[root].nex[bi]>-1){

tl=dic[root].nex[bi];

}

else{

tl=dic[root].nex[bi^1];

}

if(ans<xorval||(ans==xorval&&tl<l)){

l=tl;

r=noi;

}

ans=xorval;

}

return ;

}

if(dic[root].nex[bi]>-1){

query(dic[root].nex[bi],val,ceng-1,xorval+(1<<ceng),noi);

}

else{

query(dic[root].nex[bi^1],val,ceng-1,xorval,noi);

}

}

int main(){

int t,n;

ios::sync\_with\_stdio(0);

cin>>t;

for(int ii=1;ii<=t;ii++){

cin>>n;

ans=0;

l=0,r=1;

no=1;

dic[0].nex[0]=dic[0].nex[1]=-1;

add(0,0,15,0);

int val=0,tmp;

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

cin>>tmp;

val^=tmp;

query(0,val,15,0,i);

add(0,val,15,i);

}

cout<<"Case #"<<ii<<":"<<endl;

cout<<l+1<<' '<<r<<endl;

}

}

I:

/\*

线段树区间更新,tmp表示当前均为1或2,val表示该区间最大长度,lc表示该区间从左边第一个起的最大长度,rc表示该区间从最右边起连续为B的最大长度,具体操作见代码

\*/

#include<bits/stdc++.h>

using namespace std;

const int maxn=1e6+9;

int lc[maxn<<2];

int rc[maxn<<2];

int val[maxn<<2];

int tmp[maxn<<2];

char s[maxn];

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

tmp[o]=0;

if(l==r){

val[o]=lc[o]=rc[o]=s[l-1]=='B';

return;

}

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

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

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

lc[o]=lc[o<<1];

rc[o]=rc[o<<1|1];

if(lc[o<<1]==mid-l+1){

lc[o]+=lc[o<<1|1];

}

if(rc[o<<1|1]==r-mid){

rc[o]+=rc[o<<1];

}

val[o]=max(val[o<<1],val[o<<1|1]);

val[o]=max(val[o],rc[o<<1]+lc[o<<1|1]);

}

void putdown(int o,int l,int r){

if(l==r||!tmp[o])return;

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

int lz=o<<1,rz=o<<1|1;

if(tmp[o]==1){

val[lz]=val[rz]=lc[lz]=lc[rz]=rc[lz]=rc[rz]=0;

}

else{

val[lz]=lc[lz]=rc[lz]=mid-l+1;

val[rz]=lc[rz]=rc[rz]=r-mid;

}

tmp[lz]=tmp[rz]=tmp[o];

tmp[o]=0;

}

void update(int o,int l,int r,int ul,int ur,int va){

putdown(o,l,r);

if(ul<=l&&ur>=r){

tmp[o]=va;

lc[o]=rc[o]=val[o]=(va>1?(r-l+1):0);

return;

}

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

if(ul<=mid){

update(o<<1,l,mid,ul,ur,va);

}

if(ur>mid){

update(o<<1|1,mid+1,r,ul,ur,va);

}

lc[o]=lc[o<<1];

rc[o]=rc[o<<1|1];

if(lc[o]==mid-l+1){

lc[o]+=lc[o<<1|1];

}

if(rc[o]==r-mid){

rc[o]+=rc[o<<1];

}

val[o]=max(val[o<<1],val[o<<1|1]);

val[o]=max(val[o],rc[o<<1]+lc[o<<1|1]);

}

int query(int o,int l,int r,int ql,int qr){

putdown(o,l,r);

if(ql<=l&&qr>=r){

return val[o];

}

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

int lz=o<<1,rz=o<<1|1;

if(qr<=mid){

return query(o<<1,l,mid,ql,qr);

}

else if(ql>mid){

return query(o<<1|1,mid+1,r,ql,qr);

}

else{

int ans=max(query(o<<1,l,mid,ql,qr),query(o<<1|1,mid+1,r,ql,qr));

ans=max(ans,min(mid-ql+1,rc[o<<1])+min(qr-mid,lc[o<<1|1]));

return ans;

}

}

int main(){

int t,n,m,l,r,va,ch;

ios::sync\_with\_stdio(0);

cin>>t;

for(int ii=1;ii<=t;ii++){

cin>>n>>m;

cin>>s;

build(1,1,n);

cout<<"Case #"<<ii<<":\n";

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

cin>>ch>>l>>r;

if(ch==1){

cin>>va;

update(1,1,n,l,r,va);

}

else{

cout<<query(1,1,n,l,r)<<'\n';

}

}

}

}

J:

题意：判断是否存在一个环，每个环中位置相邻，颜色相同，环中数量大于3

/\*

Bfs:搜索没有搜过的点，然后计录与当起始搜索点的距离，最后统计每个点，若该点距离大于2，且有1个以上相邻点的距离小于等于该值，则符合题意

\*/

#include<bits/stdc++.h>

using namespace std;

char ma[55][55];

int n,m;

int step[4][2]={1,0,0,1,-1,0,0,-1};

bool vis[55][55];

int num[55][55];

void sol(int xx,int yy,int k){

vis[xx][yy]=1;

num[xx][yy]=k;

for(int i=0;i<4;i++){

int tx=xx+step[i][0],ty=yy+step[i][1];

if(tx>=0&&tx<n&&ty>=0&&ty<m&&ma[xx][yy]==ma[tx][ty]&&num[tx][ty]>num[xx][yy]+1){

sol(tx,ty,k+1);

}

}

}

int main(){

ios::sync\_with\_stdio(0);

while(cin>>n>>m){

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

cin>>ma[i];

}

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

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

for(int j=0;j<m;j++){

num[i][j]=3000;

}

}

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

for(int j=0;j<m;j++){

if(!vis[i][j])sol(i,j,0);

}

}

bool ans=0;

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

for(int j=0;j<m;j++){

if(num[i][j]<2)continue;

int t=0;

for(int k=0;k<4;k++){

int ti=i+step[k][0],tj=j+step[k][1];

if(ti>=0&&ti<n&&tj>=0&&tj<m&&num[ti][tj]<=num[i][j]&&ma[ti][tj]==ma[i][j])t++;

}

if(t>1){

ans=1;goto loop;

}

}

}

loop:

if(ans){

cout<<"Yes"<<endl;

}

else{

cout<<"No"<<endl;

}

}

return 0;

}

K:

题意：每天每个人可以修复一个bug,给出n个bug的最后期限，问最少需多少人使每个bug均在最后期限内完成

/\*

计数排序,大于n不用考虑

\*/

#include<bits/stdc++.h>

using namespace std;

const int maxn=1e6+9;

int mark[maxn];

int main(){

int n,t;

ios::sync\_with\_stdio(0);

while(cin>>n){

memset(mark,0,sizeof(mark));

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

cin>>t;

if(t>n)continue;

mark[t]++;

}

int ans=1;

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

mark[i]+=mark[i-1];

ans=max(ans,mark[i]/i+(mark[i]%i>0));

}

cout<<ans<<endl;

}

return 0;

}

L:

题意：求”AB”出现多少次

#include<bits/stdc++.h>

using namespace std;

const int maxn=1e6+9;

char s[maxn];

int main(){

int t;

ios::sync\_with\_stdio(0);

cin>>t;

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

int ans=0;

cin>>s;

for(int j=0;s[j];j++){

if(s[j]=='A'&&s[j+1]=='B')ans++;

}

cout<<"Case #"<<i<<":"<<endl<<ans<<endl;

}

return 0;

}