头文件：

#include "cstdio"

#include "map"

#include "set"

#include "cmath"

#include "queue"

#include "vector"

#include "string"

#include "cstring"

#include "time.h"

#include "iostream"

#include "stdlib.h"

#include "algorithm"

#define db double

#define ll long long

#define vec vector<ll>

#define Mt vector<vec>

#define ci(x) scanf("%d",&x)

#define cd(x) scanf("%lf",&x)

#define cl(x) scanf("%lld",&x)

#define pi(x) printf("%d\n",x)

#define pd(x) printf("%f\n",x)

#define pl(x) printf("%lld\n",x)

#define rep(i, x, y) for(int i=x;i<=y;i++)

const int N = 1e6 + 5;

const int mod = 1e9 + 7;

const int MOD = mod - 1;

const db eps = 1e-18;

const db PI = acos(-1.0);

using namespace std;

快速读入：

ll R()

{

ll 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;

}

构造函数：

struct P

{

int u,v,w;

P(int x,int y,int z):u(x),v(y),w(z){};

P(){};

};

离散化：

void init(int \*x) {

for(int i=0;i<n;i++) A[i]=x[i];

sort(A,A+n);

int \_=unique(A,A+n)-A;

for(int i=0;i<n;i++) x[i]=lower\_bound(A,A+\_,x[i])-A+1;//从1开始

}

highest bit one(i):

int F(int i) {

i |= (i >> 1);

i |= (i >> 2);

i |= (i >> 4);

i |= (i >> 8);

i |= (i >> 16);

return i - (((unsigned)i)>>1);

}

正则表达式的匹配：

#include<bits/stdc++.h>

#include<regex>

#define db double

#define ll long long

using namespace std;

const int N=1e6+5;

using namespace std;

const int MAXN=2600;

string s,p;

int main()

{

int T;

cin>>T;

while(T--)

{

cin>>s>>p;

string s1=".\*";

string s2="(.)\\1\*";

int len=s2.length();

auto pos=p.find(s1);

while(pos!=string::npos)

{

p.replace(pos,2,s2);//replace two letters after the pos that we have found

pos=p.find(s1,pos+len);//find the string we need after POS+LEN

}

regex pat(p);

if(regex\_match(s,pat)) cout<<"yes"<<endl;

else cout<<"no"<<endl;

}

return 0;

}

LIS(n\*logn):

(1):

#include"bits/stdc++.h"

using namespace std;

const int N = 1e6 + 5;

int A[N],B[N],MaxV[N];

int l;

int BS(int \*MaxV, int size, int x) {

int l = 0, r = size - 1;

while (l <= r) {

int mid = (l + r) / 2;

if (MaxV[mid] <= x) l = mid + 1;

else r = mid - 1;

}

return l;

}

int LIS(int \*S, int size) {

MaxV[0] = S[0];

l = 1;

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

if (S[i] > MaxV[l - 1]) MaxV[l++] = S[i];

else {

int pos = BS(MaxV, l, S[i]);

MaxV[pos] = S[i];

}

}

return l;

}

int main() {

int T;

cin >> T;

while (T--){

int n, k;

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

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

scanf("%d", &A[i]), B[i] = -A[i];

int lup = LIS(A, n);

int ldown = LIS(B, n);

if ((n - lup <= k) || (n - ldown <= k))

printf("A is a magic Say.\n");

else

printf("A is not a magic Say.\n");

}

return 0;

}

(2):

void solve

{

fill(dp, dp + n, INF);

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

\*lower\_bound(dp, dp + n, a[i]) = a[i];

int ans=lower\_bound(dp, dp+n, INF)-dp;

}

LCS(n\*logn):

#include"bits/stdc++.h"

#define ci(x) scanf("%d",&x)

#define pi(x) printf("%d\n",x)

using namespace std;

const int N = 1e6 + 5;

map<int, int> v;

int a[N], b[N], c[N];

int n;

int main() {

ci(n);

v.clear();

for (int i = 0; i < n; i++) ci(a[i]),v[a[i]] = i;

int nn = 0, x;

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

scanf("%d", &x);

if (v.find(x) != v.end()) b[nn++] = v[x];

}

int len = 0, p;

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

p = lower\_bound(c, c + len, b[i]) - c, c[p] = b[i];

if (p == len) len++;

}

pi(len);

return 0;

}

单调栈:

*//*求*l*，*r*内最大矩形面积*:*

*//h[i]:[L[i],R[i])* 左开右闭*,h[i]*为区间的最小值

*//L[i]:*对于*h[i]*将栈内大于它的数都去掉，这些数不可能做为左端点，若栈为空则*L[i]=0,*否则*L[i]=s.top()+1;*

*//R[i]:*对于*h[i]*将栈内大于它都数都去掉，这些数不可能做为右端点*,*若栈为空则*R[i]=n,*否则*R[i]=s.top();*

#include**"stack"**

**int** h[N];

**int** L[N],R[N];

**int** main()

{

**int** n;

**while**(scanf(**"%d"**,&n)==1&&n!=0)

{

**for**(**int** i=0;i<n;i++) ci(h[i]);

stack<**int**> s;

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

**while**(s.size()>0&&h[i]<=h[s.top()]) s.pop();

**if**(s.empty()==1) L[i]=0;

**else** L[i]=s.top()+1;

s.push(i);

}

**while**(!s.empty()) s.pop();

**for**(**int** i=n-1;i>=0;i--){

**while**(s.size()>0&&h[i]<=h[s.top()]) s.pop();

**if**(s.empty()==1) R[i]=n;

**else** R[i]=s.top();

s.push(i);

}

ll res=0;

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

res=max(res,1ll\*h[i]\*(R[i]-L[i]));

pl(res);

}

**return** 0;

}

*//*单调队列*:*

*//(*求连续区间极值*):*求*a[i~i+k-1]*的 *min* 与 *max.*

#include**"deque"**

**int** main()

{

**int** n,k;

deque<**int**> q;

**while**(scanf(**"%d%d"**,&n,&k)==2)

{

**if**(q.size()>0) q.clear();

**for**(**int** i=0;i<n;i++) ci(a[i]);

**for**(**int** i=0;i<n;i++){*//get min*

**while**(q.size()>0&&a[q.back()]>=a[i]) q.pop\_back();

q.push\_back(i);

**if**(i-k+1>=0){

b[i-k+1]=a[q.front()];

**if**(q.front()==i-k+1) q.pop\_front();

}

}

**if**(q.size()>0) q.clear();

**for**(**int** i=0;i<n;i++){*//get max*

**while**(q.size()>0&&a[q.back()]<=a[i]) q.pop\_back();

q.push\_back(i);

**if**(i-k+1>=0){

c[i-k+1]=a[q.front()];

**if**(q.front()==i-k+1) q.pop\_front();

}

}

**for**(**int** i=0;i<=n-k;i++) printf(**"%d%c"**,b[i],i==n-k?**'\n'**:**' '**);

**for**(**int** i=0;i<=n-k;i++) printf(**"%d%c"**,c[i],i==n-k?**'\n'**:**' '**);

}

**return** 0;

}

凸多边形面积：

for(i=1;i<n;i++) s+=fabs(x[i]\*y[i+1]-y[i]\*x[i+1])\*0.5;//同方向

s+=fabs(x[1]\*y[n]-x[n]\*y[1])\*0.5;

字符串转数字:

char\* : sscanf(str,”%lld”,&ans);

string: atoi(s.data()) (int) | atoll(s.data()) (ll)

数字转字符串:

char\* : sprintf(str,”%lld”,ans);

string: to\_string(x);

字符串反转:

char\* : strrev(s);

string: reverse(s.begin(),s.end())

long double:

定义:long double  a;

输入:scanf(“%Lf",&a);

输出:printf("%.10Lf",a);

常用函数都要在末尾加个 “l”，如fabsl(a),sqrtl(a),cosl(a)…

容斥定理：

求区间内n个数的倍数共有多少个

#include"bits/stdc++.h"

#define db double

#define ll long long

#define ci(x) scanf("%d",&x)

#define cd(x) scanf("%lf",&x)

#define cl(x) scanf("%lld",&x)

#define pi(x) printf("%d\n",x)

#define pd(x) printf("%f\n",x)

#define pl(x) printf("%lld\n",x)

#define rep(i,n) for(int i=0;i<n;i++)

const int N = 1e5 + 5;

const int mod = 1e9 + 7;

const int MOD = mod - 1;

const int inf = 0x3f3f3f3f;

const db PI = acos(-1.0);

const db eps = 1e-10;

using namespace std;

ll a[N],b[N];

bool v[N];

int m=0,n=0;

ll l,r,ans=0;

ll gcd(ll x,ll y){

return y==0?x:gcd(y,x%y);

}

void init(ll x){

if(x>r) return ;

else if(x>0) a[++m]=x;//0不加入

init(x\*10+6);

init(x\*10+8);

}

void dfs(int x,int y,ll z)//表示：前x个数字中选y个数字时的lcm为z

{

if(x==n){

if(y&1) ans+=r/z-(l-1)/z;//奇数加

if(y%2==0&&y!=0) ans-=r/z-(l-1)/z;//偶数减

return;

}

dfs(x+1,y,z);

ll s=z/gcd(a[x+1],z);

if((db(s)\*a[x+1])<=r) dfs(x+1,y+1,a[x+1]\*s);//一个剪枝：lcm<=r

}

int main()

{

cl(l),cl(r);

init(0);

sort(a+1,a+m+1);

for(int i=1;i<=m;i++){//去掉内部倍数情况

if(!v[i]){

b[++n]=a[i];

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

if(a[j]%a[i]==0) v[j]=1;

}

}

reverse(a+1,a+n+1);//数字从大到小排

dfs(0,0,1);

pl(ans);

return 0;

}

全排列：

for(int i=0;i<n;i++) a[i]=i+1;

do{

for(int i=0;i<n;i++) printf("%d ",a[i]);

puts("");

}while(next\_permutation(a,a+n));

求第K大的数：O(n)

nth\_element(a, a+k-1, a+n)

(使第k大元素处于第k位置，且比此元素小的都排在其前，比此元素大的都排在其后，但不能保证有序)

判断是否有序：is\_sorted(a,a+n,cmp);