**Solution**   
三维偏序关系，CDQ分治即可，比较麻烦的是最小字典序，以dp[i]表示以第i个元素开始的LIS长度，类似一般的求以i结尾的LIS长度可知此处需要倒过来做CDQ，即先CDQ(mid+1,r)，然后根据[mid+1,r]更新[l,mid]，之后再CDQ(l,mid)，那么CDQ(1,n)可以得到所有的dp值，则LIS长度ans=max(dp[i])，之后从前往后扫，每次找dp[i]==ans&&(pre==0||(p[pre].x>=p[i].x&&p[pre].y<=p[i].y))的i，其中pre为LIS上一个元素的编号，初始化为0，每次找到一个i就ans–,pre=i，这样就可以得到字典序最小的LIS了

#include<iostream>

#include<cstdio>

#include<cstring>

#include<string>

#include<algorithm>

#include<cmath>

#include<queue>

#include<map>

#include<vector>

#include<set>

#include<bitset>

#include<stack>

#include<cctype>

#define sc(x) scanf("%d",&x);

#define pr(x) cout<<#x<<": "<<x<<" ";

#define prln(x) cout<<#x<<": "<<x<<endl;

using namespace std;

typedef long long ll;

typedef pair<int,int> p;

const double eps = 1e-4;

const int mod = 1e9+7;

const ll INF = 0x3f3f3f3f3f3f3f3f;

const int inf = 0x3f3f3f3f;

const int maxn = 1e5+7;

const int maxm = 1e6+7;

const double pi = acos(-1.0);

int n,tot;

int dp[maxn];

struct Node{

int id,l,r;

bool operator < ( const Node& a ){

if(r!=a.r)

return r<a.r;

if(l!=l)

return l>a.l;

return id<a.id;

}

}a[maxn];

int id[maxn];

bool cmpid(Node a,Node b){

return a.id<b.id;

}

struct Bit{

int n,treenum[maxn];

void init(){

memset(treenum,0,sizeof treenum);

}

int lowbit(int x){

return x&(-x);

}

void add(int i,int add){

while(i<=tot){

treenum[i]+=add;

i+=lowbit(i);

}

}

int query(int i){

int sum=0;

while(i){

sum=max(sum,treenum[i]);

i-=lowbit(i);

}

return sum;

}

void upadte(int i,int val){

while(i<=tot){

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

i+=lowbit(i);

}

}

void clear(int i){

while(i<=tot){

treenum[i]=0;

i+=lowbit(i);

}

}

}bit;

void cdq(int l,int r){

if(l==r)

return;

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

cdq(mid+1,r);

sort(a+l,a+mid+1);

sort(a+mid+1,a+r+1);

int li=r;

for(int i=mid; i>=l; i--){

for(; li>mid&&a[li].r>=a[i].r; li--)

bit.upadte(a[li].l,dp[a[li].id]);

dp[a[i].id]=max(dp[a[i].id],bit.query(a[i].l)+1);

}

for(int i=mid+1; i<=r; i++)

bit.clear(a[i].l);

sort(a+l,a+mid+1,cmpid);

cdq(l,mid);

}

int tmp[maxn];

int main(){

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

bit.init();

tot=0;

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

scanf("%d",&a[i].l),tmp[++tot]=a[i].l;

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

scanf("%d",&a[i].r),tmp[++tot]=a[i].r;

sort(tmp+1,tmp+tot+1);

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

a[i].l=lower\_bound(tmp+1,tmp+tot+1,a[i].l)-tmp;

a[i].r=lower\_bound(tmp+1,tmp+tot+1,a[i].r)-tmp;

dp[i]=1; a[i].id=i;

// pr(i) pr(a[i].l) prln(a[i].r)

}

cdq(1,n);

sort(a+1,a+n+1,cmpid);

int ans=0,pre=0;

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

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

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

// prln(ans)

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

// prln(dp[i])

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

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

if( dp[i]==ans && ( pre==0 || (a[pre].l>=a[i].l&&a[pre].r<=a[i].r) )){

printf("%d%c",i,(--ans)?' ':'\n');

pre=i;

}

}

return 0;

}

/\*

5

5 4 3 2 1

6 7 8 9 10

2

1 2

3 4

5

1 2 3 4 5

1

1

\*/