**算法训练 会议中心**

时间限制：2.0s   内存限制：512.0MB

　　会议中心　　Siruseri政府建造了一座新的会议中心。许多公司对租借会议中心的会堂很感兴趣，他们希望能够在里面举行会议。  
　　对于一个客户而言，仅当在开会时能够独自占用整个会堂，他才会租借会堂。会议中心的销售主管认为：最好的策略应该是将会堂租借给尽可能多的客户。显然，有可能存在不止一种满足要求的策略。  
　　例如下面的例子。总共有4个公司。他们对租借会堂发出了请求，并提出了他们所需占用会堂的起止日期（如下表所示）。

|  |  |  |
| --- | --- | --- |
|  | 开始日期 | 结束日期 |
| 公司1 | 4 | 9 |
| 公司2 | 9 | 11 |
| 公司3 | 13 | 19 |
| 公司4 | 10 | 17 |

　　上例中，最多将会堂租借给两家公司。租借策略分别是租给公司1和公司3，或是公司2和公司3，也可以是公司1和公司4。注意会议中心一天最多租借给一个公司，所以公司1和公司2不能同时租借会议中心，因为他们在第九天重合了。  
　　销售主管为了公平起见，决定按照如下的程序来确定选择何种租借策略：首先，将租借给客户数量最多的策略作为候选，将所有的公司按照他们发出请求的顺序编号。对于候选策略，将策略中的每家公司的编号按升序排列。最后，选出其中字典序最小[1]的候选策略作为最终的策略。  
　　例中，会堂最终将被租借给公司1和公司3：3个候选策略是{(1,3),(2,3),(1,4)}。而在字典序中(1,3)<(1,4)<(2,3)。  
　　你的任务是帮助销售主管确定应该将会堂租借给哪些公司。

输入格式

　　输入的第一行有一个整数*N*，表示发出租借会堂申请的公司的个数。第2到第*N*+1行每行有2个整数。第*i*+1行的整数表示第*i*家公司申请租借的起始和终止日期。对于每个公司的申请，起始日期为不小于1的整数，终止日期为不大于109的整数。

输出格式

　　输出的第一行应有一个整数*M*，表示最多可以租借给多少家公司。第二行应列出*M*个数，表示最终将会堂租借给哪些公司。

数据规模和约定

　　对于50%的输入，*N*≤3000。在所有输入中，*N*≤200000。

样例输入

4  
4 9  
9 11  
13 19  
10 17

样例输出

2  
1 3  
  
[1] 字典序指在字典中排列的顺序，如果序列*l*1是序列*l*2的前缀，或者对于*l*1和*l*2的第一个不同位置*j*，*l*1[*j*]<*l*2[*j*]，则*l*1比*l*2小。

本题的C++参考代码如下：

#include <cstdio>

#include <cstdlib>

#include <algorithm>

#include <string>

#include <set>

const char fi[] = "convention.in";

const char fo[] = "convention.out";

const int maxN = 200010;

const int MAX = 0x3f3f3f3f,MIN = ~MAX;

struct Seg

{

int L,R;

Seg()

{

}

Seg(int L,int R): L(L),R(R)

{

}

bool operator<(const Seg &b) const

{

return L < b.L || L == b.L && R < b.R;

}

};

std::set <Seg> S;

std::set <Seg>::iterator iter;

Seg req[maxN],seg[maxN],tmp[maxN];

int tab[maxN << 1],next[20][maxN << 1];

int n,cnt,Lim = 1,logLim;

void init\_file()

{

return;

}

inline int getint()

{

int res = 0; char tmp;

while(!isdigit(tmp = getchar()));

do res = (res << 3) + (res << 1) + tmp - '0';

while(isdigit(tmp = getchar()));

return res;

}

void readdata()

{

n = getint();

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

{

int L = getint(),R = getint();

req[i] = Seg(L,R);

tab[i << 1] = L;

tab[(i << 1) + 1] = R;

}

return;

}

int plc(int x)

{

for(int L = 0,R = Lim - 1; L < R + 1;)

{

int Mid = (L + R) >> 1;

if(x == tab[Mid]) return Mid + 1;

if(x < tab[Mid]) R = Mid - 1;

else L = Mid + 1;

}

}

bool cmp(const Seg &a,const Seg &b)

{

return a.R < b.R || a.R == b.R && a.L > b.L;

}

void discrete()

{

std::sort(tab,tab + (n << 1));

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

if(tab[i] != tab[i - 1])

tab[Lim++] = tab[i];

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

tmp[i] = req[i] = Seg(plc(req[i].L),

plc(req[i].R));

std::sort(tmp,tmp + n,cmp);

//这里必须要用一个临时数组，

//保证左界右界同时单调递增。

int p = 0; seg[cnt++] = tmp[0];

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

if(tmp[i].L > tmp[p].L)

seg[cnt++] = tmp[p = i];

return;

}

void next\_set()

{

int p = cnt; next[0][Lim + 1] = MAX;

for(int j = Lim; j; --j)

if(p > -1 && j == seg[p - 1].L)

next[0][j] = seg[--p].R + 1;

else next[0][j] = next[0][j + 1];

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

{

bool flag = 0;

next[i + 1][Lim + 1] = MAX;

for(int k = 1; k < Lim + 1; ++k)

{

if(next[i][k] == MAX)

next[i + 1][k] = MAX;

else next[i + 1][k] = next[i][next[i][k]];

if(next[i + 1][k] < MAX) flag = 1;

}

if(!flag)

{

logLim = i; break;

}

}

return;

}

int max\_time(int L,int R)

{

if(L > R++) return 0;

int ans = 0,p = L;

for(int i = logLim; i > -1 && p < R; --i)

if(next[i][p] <= R)

{

p = next[i][p]; ans += 1 << i;

}

return ans;

}

bool query(int i)

{

int L = req[i].L,R = req[i].R;

iter = S.lower\_bound(Seg(L,MAX));

if(iter-- == S.begin()) return 0;

if(iter->L > L || iter->R < R)

return 0;

int L1 = iter->L,R1 = iter->R;

if(max\_time(L1,L - 1)

+ max\_time(R + 1,R1)

+ 1 < max\_time(L1,R1))

//这里要满足放进去过后不影响总的答案。

return 0;

S.erase(iter);

if(L1 < L) S.insert(Seg(L1,L - 1));

if(R < R1) S.insert(Seg(R + 1,R1));

return 1;

}

void work()

{

printf("%d\n",max\_time(1,Lim));

S.insert(Seg(1,Lim));

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

if(query(i))

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

printf("\n");

return;

}

int main()

{

init\_file();

readdata();

discrete();

next\_set();

work();

return 0;

}

本题的C参考代码如下：

本题的Java参考代码如下：

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

public class Main{

static int[] sqqs,sqzz,sqsx,sqcf,cxz,finz;

static int[][] dgb;

static int n=0,fin=0,max=0;

public static void main(String[] args)throws IOException {

BufferedReader reader=new BufferedReader(new InputStreamReader(System.in));

String s=reader.readLine();

n=Integer.parseInt(s);

sqqs=new int[n+1];

sqzz=new int[n+1];

sqcf=new int[n+1];

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

s=reader.readLine();

String[] sa = s.split(" ");

sqqs[i]=Integer.parseInt(sa[0]);

sqzz[i]=Integer.parseInt(sa[1]);

if (max<sqzz[i]) max=sqzz[i];

}

sqsx=new int[max+1];

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

if (sqsx[sqzz[i]]==0){

sqsx[sqzz[i]]=i;

}else{

if (sqqs[i]>sqqs[sqsx[sqzz[i]]]){

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

if (sqcf[j]==0){

sqcf[j]=sqsx[sqzz[i]];

sqsx[sqzz[i]]=i;

break;

}

}

}

}

}

dgb=new int[2][n+1];

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

if (sqsx[i]!=0) {

getcs(sqsx[i]);

}

}

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

if (fin<dgb[0][i]) fin=dgb[0][i];

}

for (int k=max;k>=1;k--){

if (sqsx[k]==0) continue;

int i=sqsx[k];

if (dgb[0][i]==fin) continue;

for (int j=sqzz[i]+1;j<=max;j++){

if (j==max){

dgb[0][i]=0;

dgb[1][i]=0;

continue;

}

if (sqsx[j]==0) continue;

if (dgb[0][sqsx[j]]==0) continue;

if ((dgb[0][sqsx[j]]==dgb[0][i]+1)&(sqqs[sqsx[j]]>sqzz[i])) break;

if (dgb[0][sqsx[j]]>dgb[0][i]+1) {

dgb[0][i]=0;

dgb[1][i]=0;

break;

}

}

}

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

if (sqcf[i]!=0) {

getcs1(sqcf[i]);

} else break;

}

cxz=new int[fin+1];

finz=new int[fin+1];

int xh=1;

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

if (dgb[0][i]==0) continue;

if (cxz[dgb[0][i]]!=0) continue;

if (fincheck(i)==1){

finz[xh]=i;

xh++;

cxz[dgb[0][i]]=i;

dealsq(i);

}

}

System.out.println(fin);

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

System.out.print(""+finz[i]+" ");

}

System.out.println();

}

static void getcs(int a){

int cs=0;

for (int i=sqqs[a]-1;i>0;i--){

if (sqsx[i]!=0){

cs=dgb[0][sqsx[i]];

if (cs!=0) {

break;

}

}

}

for (int i=sqqs[a];i<sqzz[a];i++){

if (sqsx[i]!=0){

if (dgb[0][sqsx[i]]>cs+1) return;

}

}

dgb[0][a]=cs+1;

dgb[1][a]=1;

}

static void getcs1(int a){

int cs=0;

int b=dgb[0][sqsx[sqzz[a]]];

if (b==0) return;

for (int i=sqqs[a]-1;i>0;i--){

if (sqsx[i]!=0){

cs=dgb[0][sqsx[i]];

if (cs!=0) {

break;

}

}

}

if (cs+1<b) return;

dgb[0][a]=cs+1;

dgb[1][a]=1;

}

static int fincheck(int a){

int[] lsst=new int[fin+1];

int b=0,c=0,sc=0,ec=0;

int cs=dgb[0][a];

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

if (cxz[i]!=0) {

b=cxz[i];

sc=i;

break;

}

}

if (b!=0){

lsst[sc]=sqzz[b];

for (int i=sqzz[b]+1;i<sqqs[a];i++){

if (sqsx[i]==0) continue;

if (dgb[0][sqsx[i]]==0) continue;

if (lsst[dgb[0][sqsx[i]]]!=0) continue;

if (sqqs[sqsx[i]]<=lsst[dgb[0][sqsx[i]]-1]) continue;

lsst[dgb[0][sqsx[i]]]=sqzz[sqsx[i]];

}

if (lsst[cs-1]==0) return 0;

if (lsst[cs-1]>=sqqs[a]) return 0;

}

for (int i=cs+1;i<=fin;i++){

if (cxz[i]!=0){

c=cxz[i];

ec=i;

break;

}

}

if (c!=0){

lsst[ec]=sqqs[c];

for (int i=sqqs[c]-1;i>sqzz[a];i--){

if (sqsx[i]==0) continue;

if (dgb[0][sqsx[i]]==0) continue;

if (i>=lsst[dgb[0][sqsx[i]]+1]) continue;

lsst[dgb[0][sqsx[i]]]=Math.max(sqqs[sqsx[i]],lsst[dgb[0][sqsx[i]]]);

}

if (lsst[cs+1]==0) return 0;

if (lsst[cs+1]<=sqzz[a]) return 0;

}

return 1;

}

static void dealsq(int a){

int cs=dgb[0][a];

for (int i=sqzz[a]-1;i>=1;i--){

if (sqsx[i]==0) continue;

if (dgb[0][sqsx[i]]==0) continue;

int dcs=dgb[0][sqsx[i]];

if (dcs==cs) {

dgb[0][sqsx[i]]=0;

dgb[1][sqsx[i]]=0;

continue;

}

if (dcs==cs-1) {

if (i>=sqqs[a]) {

dgb[0][sqsx[i]]=0;

dgb[1][sqsx[i]]=0;

continue;

}

}

if (dcs==cs-2) break;

}

for (int i=sqzz[a]+1;i<=max;i++){

if (sqsx[i]==0) continue;

if (dgb[0][sqsx[i]]==0) continue;

int dcs=dgb[0][sqsx[i]];

if (dcs==cs) {

dgb[0][sqsx[i]]=0;

dgb[1][sqsx[i]]=0;

continue;

}

if (dcs==cs+1) {

if (sqzz[a]>=sqqs[sqsx[i]]) {

dgb[0][sqsx[i]]=0;

dgb[1][sqsx[i]]=0;

continue;

}

}

if (dcs==cs+2) break;

}

}

}