P1803 凌乱的yyy

题目描述

现在各大 oj 上有 n 个比赛，每个比赛的开始、结束的时间点是知道的。

yyy 认为，参加越多的比赛，noip 就能考的越好（假的）。

所以，他想知道他最多能参加几个比赛。

由于 yyy 是蒟蒻，如果要参加一个比赛必须善始善终，而且不能同时参加 2 个及以上的比赛。

输入格式

第一行是一个整数 n ，接下来 n 行每行是 2 个整数 ai,bi (ai<bi)，表示比赛开始、结束的时间。

输出格式

一个整数最多参加的比赛数目。

输入样例

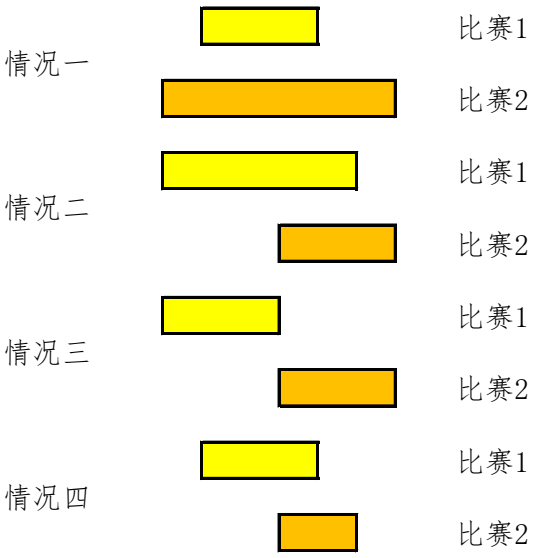
```
3
0 2
2 4
1 3
```

输出样例

```
2
```

解析

本题所使用的方法是数学归纳法。观察下面的图片：



时刻想着我们的问题：如何能参加更多的比赛？

对于情况1来说，显然我们应该选择比赛1，因为比赛1的结束时间更早，使得我们能够参加更多的比赛。

对于情况2来说，显然也应该选择比赛1，道理同上。

对于情况3来说，也是同理。

细细想来，对于两个不同的比赛，无论它们处于何种状态，我们总是应该选择更早结束的那个。因此归纳最终的方案为：优先选择最早结束的比赛，才能使我们参加更多的比赛。

这就是数学归纳方法。罗列出各种可能性，总结归纳出其中的共性，进而解决问题。

编码

```
#include <bits/stdc++.h>

using namespace std;
const int Maxn = 1e6 + 1;
int n, currentEnd, countNum = 1;

//定义比赛的结构体，存储每个比赛的开始时间和结束时间
struct Game {
    int start;
    int end;
};

Game games[Maxn];

//按照结束时间进行排序
```

```
bool Compare(Game a, Game b) {
    return a.end < b.end;
}


int main() {
    //读入比赛的基本信息
    scanf("%d", &n);
    for (int i = 0; i < n; ++i) {
        scanf("%d %d", &games[i].start, &games[i].end);
    }
    //按照比赛结束的先后顺序进行排序
    sort(games, games + n, Compare);
    //记录第一个比赛的结束时间
    currentEnd = games[0].end;
    //遍历剩余比赛，只要开始时间晚于当前比赛的结束时间才能入选
    for (int i = 1; i < n; i++) {
        if (games[i].start >= currentEnd) {
            countNum++;
            //更新最新的比赛结束时间
          currentEnd = games[i].end;
        }
    }
    //输出最终的数量
    cout << countNum;
    return 0;
}
```