1236：区间合并

题目描述

给定 n 个闭区间 [ai，bi]，其中i=1,2,...,n。任意两个相邻或相交的闭区间可以合并为一个闭区间。例如，[1，2] 和 [2，3] 可以合并为 [1，3]，[1，3] 和 [2，4] 可以合并为 [1，4]，但是[1，2] 和 [3，4] 不可以合并。

我们的任务是判断这些区间是否可以最终合并为一个闭区间，如果可以，将这个闭区间输出，否则输出no。

输入

第一行为一个整数n，3≤n≤50000。表示输入区间的数量。

之后n行，在第i行上（1≤i≤n），为两个整数 ai 和 bi ，整数之间用一个空格分隔，表示区间 [ai，bi]（其中 1≤ai≤bi≤10000）。

输出

输出一行，如果这些区间最终可以合并为一个闭区间，输出这个闭区间的左右边界，用单个空格隔开；否则输出 no。

输入样例

```
5
5 6
1 5
10 10
6 9
8 10
```

输出样例

```
1 10
```

解析

遇到区间问题，我们首先就要想到排序。对于本题来说，我们需要将所有的区间按照起始区间的大小进行排序。可能得到的结果如下：

情况1：

情况2：

情况3：

当然还存在其他更多的情况，以上仅是两个最特殊的情形。

根据以上三个情况的图片，我们可以大致得出计算步骤：

比较两个相邻区间的右区间和左区间，判断是否有重合部分，即a.右>=b.左。
如果重合，则合并区间：
　a、设两个区间中左边界的最小值为新区间的开始。
　b、设两个区间中右边界的最大值为新区间的结束。
如果不重合，输出“no”。

循环往复

编码

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

using namespace std;
struct a {
    int x;
    int y;
};
a s[50010];

//按照左边界进行排序
int cmp(a p, a q) {
    return p.x < q.x;
}


int main() {
    int n;
    cin >> n;
    //获取输入数据
   for (int i = 1; i <= n; ++i) {
        cin >> s[i].x >> s[i].y;
```

```
    }
    //按照左边界进行排序
    sort(s + 1, s + n + 1, cmp);
    //依次比较相邻的区间
    for (int j = 1; j <= n - 1; ++j) {
        //判断两个区间是否重合
        if (s[j].y >= s[j + 1].x) {
            //标记新区间的左边界和右边界
            s[j + 1].x = min(s[j].x, s[j + 1].x);
            s[j + 1].y = max(s[j].y, s[j + 1].y);
        }
            //存在不重合的区间，返回错误提示
        else {
            cout << "no";
            return 0;
        }
    }
    //输出新区间的左右边界
    cout << s[n].x << " " << s[n].y;
    return 0;
}
```