高效的任务规划题目描述：

你有 n 台机器编号为 $1\sim n$，每台都需要完成完成一项工作，机器经过配置后都能完成独立完成一项工作。假设第 i 台机器你需要花 B_i 分钟进行设置，然后开始运行，J_i 分钟后完成任务。现在，你需要选择布置工作的顺序，使得用最短的时间完成所有工作。注意，不能同时对两台进行配置，但配置完成的机器们可以同时执行他们各自的工作。

输入描述：

第一行输入代表总共有 M 组任务数据（$1 < M <= 10$）。

每组数第一行为一个整数指定机器的数量 N（$0 < N <= 1000$）。随后的 N 行每行两个整数，第一个表示 B（$0 <= B <= 10000$），第二个表示 J（$0 <= J <= 10000$）。

每组数据连续输入，不会用空行分隔。各组任务单独计时。

输出描述：

对于每组任务，输出最短完成时间，且每组的结果独占一行。例如，两组任务就应该有两行输出。

补充说明：

示例 1

输入：

```
1

1

2 2
```

输出：

```
4
```

说明：

输入共 3 行数据，第 1 行代表只有 1 组任务；第 2 行代表本组任务只有 1 台机器；第 3 行代表本机器：配置需要 2 分钟，执行任务需要 2 分钟。输出共 1 行数据，代表执行结果为 4 分钟。

示例 2

输入：

```
2
2
1 1
2 2
3
1 1
2 2
3 3
```

输出：

```
4
7
```

说明：

第一行 2 代表输入共 2 组数据，2-4 行代表第 1 组数据，为 2 台机器的配置、执行信息（第 1 台机器：配置需要 1 分钟，执行需要 1 分钟；第 2 台机器：配置需要 2 分钟，执行需要 2 分钟）。5-8 行代表第 2 组数据，为 3 台机器的配置、执行信息（意义同上）。输出共 2 行，分别代表第 1 组任务共需要 4 分钟和第 2 组任务共需要 7 分钟（先配置 3，再配置 2，最后配置 1，执行 1 分钟，共 7 分钟）。

```
import java.util.Arrays;
import java.util.Scanner;

/**
 *
 * @since 2023/07/09 10:39
 * @author Myo
 */
public class Main {

    static int m;
```

```
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        m = sc.nextInt();
        while (m -- > 0) {
            int n = sc.nextInt();
            int[][] time = new int[n][2];
            for (int i = 0; i < n; i++) {
                time[i][0] = sc.nextInt();
                time[i][1] = sc.nextInt();
            }

            Arrays.sort(time, (o1, o2) -> {
                if (o1[1] != o2[1]) {
                    return o2[1] - o1[1];
                }
                return o1[0] - o2[0];
            });

            // 记录每一个任务的开始时间
            int start = 0;
            // 记录每一个任务的最后结束时间
            int end = 0;
            for (int i = 0; i < n; i++) {
                start += time[i][0];
                end = Math.max(end, start + time[i][1]);
            }

            System.out.println(end);
        }
    }
}
```