高效的任务规划

题目描述：

你有 n 台机器编号为 1~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 分钟进行设置，然后开始运行，J 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

题目描述：

你有 n 台机器编号为 1~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 分钟进行设置，然后开始运行，J 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

```
// const readline = require("readline");
// async function main() {
//     const rl = readline.createInterface({
//         input: process.stdin,
//         output: process.stdput,
//     });
```

```
//     const T = parseInt(await getInput(rl));
//     const g = [];
//     for (let i = 0; i < T; i++) {
//         const n = parseInt(await getInput(rl));
//         const arr = [];
//         for (let j = 0; j < n; j++) {
//             const line = (await getInput(rl)).split(" ");
//             const a = parseInt(line[0]);
//             const b = parseInt(line[1]);
//             arr.push([a, b]);
//         }
//         g.push(arr);
//     }
//     for (let arr of g) {
//         let res = 0;
//         arr.sort((a, b) => (b[1] = a[1]));
//         let preTime = 0;
//         for (let t of arr) {
//             const a = t[0];
//             const b = t[1];
//             preTime += a;
//             res = Math.max(res, preTime + b);
//         }
//         console.log(res);
//     }
//     rl.close();
// }
// function getInput(rl) {
//     return new Promise((resolve) => {
//         rl.question("", (answer) => {
//             resolve(answer);
//         });
//     });
// }
// main();

// let a = [];
// rl.on("line", (line) => {
//     a.push(line);
//     // 收集数据
// }).on("close", () => {
//     // 处理
// });
```

```
import java.util.*;

public class Main{
   public static void main (String[] args){
      Scanner cin = new Scanner (System.in);
      int T=cin.nextInt();
      int [][][]g=new int[T][][];
      for (int i = 0; i <T; ++i) {
       int n=cin.nextInt();
       int [][]arr=new int[n][2];
       for (int j = 0; j < n; j++) {
       arr[j][0]=cin.nextInt();
       arr[j][1]=cin.nextInt();
       }g[i]=arr;
      }for (int[][]arr:g){
         int res=0;
         Arrays.sort(arr,(a,b)->b[1]-a[1]);
         int preTime=0;
         for(int[]t:arr){
            int a=t[0];
            int b=t[1];
            preTime+=a;
            res=Math.max(res,preTime+b);
         }
         System.out.println(res);
      }
   }
}
```