华为OD机试 - 比赛的冠亚季军（Java & JS & Python）

题目描述

有N（3 ≤ N < 10000）个运动员，他们的id为0到N-1,他们的实力由一组整数表示。他们之间进行比赛，需要决出冠亚军。比赛的规则是0号和1号比赛，2号和3号比赛，以此类推，每一轮，相邻的运动员进行比赛，获胜的进入下一轮；实力值大的获胜，实力值相等的情况，id小的情况下获胜；轮空的直接进入下一轮。

输入描述

输入一行N个数字代表N的运动员的实力值(0<=实力值<=10000000000)。

输出描述

输出冠亚季军的id，用空格隔开。

用例

输入	2 3 4 5
输出	3 1 2
说明	第一轮比赛， id为0实力值为2的运动员和id为1实力值为3的运动员比赛，1号胜出进入下一轮争夺冠亚军， id为2的运动员和id为3的运动员比赛，3号胜出进入下一轮争夺冠亚军， 冠亚军比赛，3号胜1号， 故冠军为3号，亚军为1号，2号与0号，比赛进行季军的争夺，2号实力值为4，0号实力值2，故2号胜出，得季军。冠亚季军为3 1 2。

题目解析

本题主要考察逻辑分析。

每轮晋级赛，都会将人数砍一半，因此本题不怕大数量级。

在每轮晋级赛中，相邻的运动员组队进行比赛，比如有实力数组：[0,1,2,3,4,5,6,7,8]

其中0,1比赛，2,3比赛，4,5比赛，6,7比赛，其中实力值较大者晋级去竞争冠军组，对于8而言，没有对手，按照题目意思是直接晋级。

按照上面逻辑，得到获胜组为：[1,3,5,7,8]，失败组为：[0,2,4,6]

我们可以创建一个链表用于保存每轮的获胜组和失败组，但是需要保证获胜组在头部

即可得 [1,3,5,7,8] -> [0,2,4,6]

接下来取出链表头部组[1,3,5,7,8]，继续进行晋级赛：

其中1,3比赛，5,7比赛，8没有对手直接晋级，

最后得到获胜组[3,7,8]，失败组[1,5]，将它们压入链表头部

[3,7,8] -> [1,5] -> [0,2,4,6]

接下来取出链表头部组[3,7,8]，继续进行晋级赛：

其中3,7比赛，8没有对手直接晋级，

最后得到获胜组[7,8]，失败组[3]，将它们压入链表头部

[7,8] -> [3] -> [1,5] -> [0,2,4,6]

此时链表长度超过了3，因此链表尾部的组失去了竞争季军的资格，因此弹出尾部

[7,8] -> [3] -> [1,5]

而链表头部组的运动员个数还不为1，即还有多个人竞争冠军

因此，继续取出链表头部组[7,8]，进行晋级赛：

其中7,8比赛，

最后得到获胜组[8]，失败组[7]，将它们压入链表头部

[8] -> [7] -> [3] -> [1,5]

此时链表长度超过了3，因此链表尾部的组失去了竞争季军的资格，因此弹出尾部

[8] -> [7] -> [3]

最后的冠军实力值8，亚军实力值7，季军实力值3

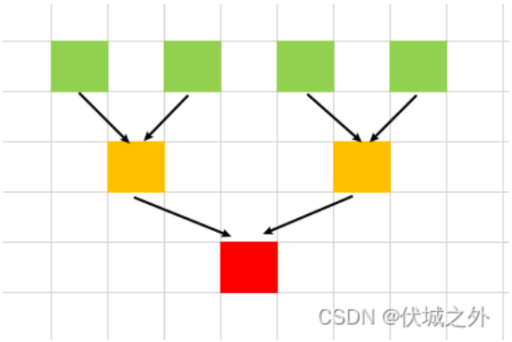
但是题目最后要求输出的是运动员的id，因此我们在一开始的时候可以定义一个运动员类，属性有运动员的id和运动员的实力。

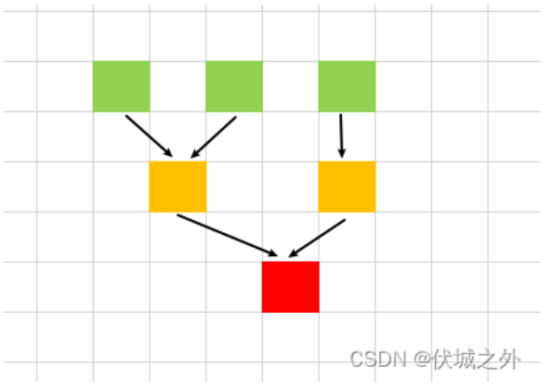
这样最后输出就可以获取到运动员id了。

这里还有一个需要注意的点是：

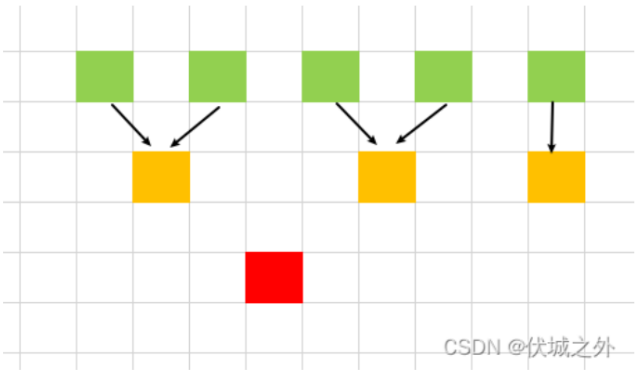
最后一轮晋级赛，必然只有两个人，即分出冠军和亚军

倒数第二轮晋级赛，只可能是4人，或者3人，如下图所示

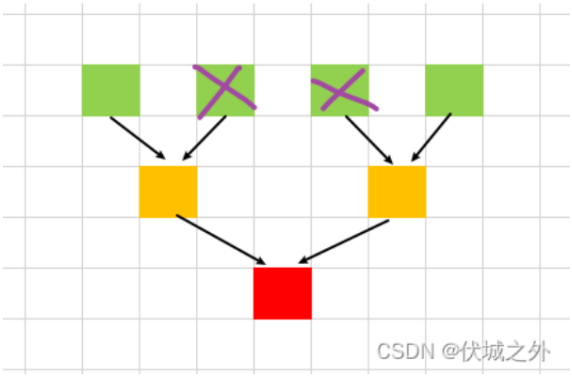


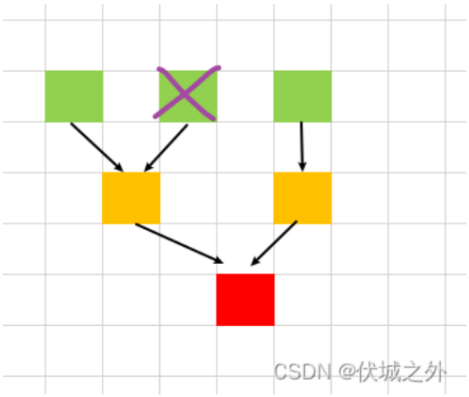


如果倒数第二轮晋级赛有五人的话，是无法在下轮中产生冠军和亚军的



因此，季军争夺组只会有2人或者1人，因为，如下图，倒数第二轮晋级赛的失败者只有两人或者一人





因此，前面定义的链表，最终的形态下：

第一个节点只有一个运动员（冠军），第二个节点只有一个运动员（亚军），第三个节点可能有一个，也可能有两个（季军争夺）

因此针对第三个节点，需要进行季军争夺，直接进行排序取第一个人即可，排序逻辑：

- 实力值大的获胜，如果实力值相同，则id小的获胜

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

public class Main {
  // 运动员类
  static class Sport {
    int id; // 运动员的id
    long strength; // 运动员的实力

    public Sport(int id, long strength) {
      this.id = id;
      this.strength = strength;
    }
  }
```

```java

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

    long[] strengths = Arrays.stream(sc.nextLine().split("
")).mapToLong(Long::parseLong).toArray();

    System.out.println(getResult(strengths));
  }

  public static String getResult(long[] strength) {
    // ans只记录三个组，冠军组，亚军组，季军组
    LinkedList<ArrayList<Sport>> ans = new LinkedList<>();

    // 将输入的实力值，转化为运动员集合
    ArrayList<Sport> sports = new ArrayList<>();
    for (int i = 0; i < strength.length; i++) sports.add(new Sport(i, strength[i]));

    // 晋级赛
    promote(sports, ans);

    // 冠军组如果不是一个人，那么还需要取出冠军组继续进行晋级赛
    while (ans.getFirst().size() > 1) {
      promote(ans.removeFirst(), ans);
    }

    // 冠军
    int first = ans.get(0).get(0).id;

    // 亚军
    int second = ans.get(1).get(0).id;

    // 季军
    ans.get(2)
        .sort(
            (a, b) ->
                a.strength != b.strength ? b.strength - a.strength > 0 ? 1 : -1 : a.id
- b.id);
    int third = ans.get(2).get(0).id;

```

```java
    return first + " " + second + " " + third;
  }

  public static void promote(ArrayList<Sport> sports, LinkedList<ArrayList<Sport>> ans)
  {
    // 记录获胜组
    ArrayList<Sport> win = new ArrayList<>();
    // 记录失败组
    ArrayList<Sport> fail = new ArrayList<>();

    for (int i = 1; i < sports.size(); i += 2) {
      // 序号大的运动员
      Sport major = sports.get(i);
      // 序号小的运动员
      Sport minor = sports.get(i - 1);

      if (major.strength > minor.strength) {
        win.add(major);
        fail.add(minor);
      } else {
        // 如果序号大的运动员的实力 <= 序号小的运动员，则序号小的运动员获胜
        win.add(minor);
        fail.add(major);
      }
    }

    // 如果晋级赛中运动员个数是奇数个，那么最后一个运动员直接晋级
    if (sports.size() % 2 != 0) {
      win.add(sports.get(sports.size() - 1));
    }

    // 依次头部压入失败组，获胜组，保证头部是获胜组
    ans.addFirst(fail);
    ans.addFirst(win);

    // 如果保留组个数超过3个，那么需要将超过部分的组去掉，因为这部分人已经无缘季军
    while (ans.size() > 3) ans.removeLast();
  }
}
```