# 题目描述

有5台打印机打印文件,每台打印机有自己的待打印队列。

因为打印的文件内容有轻重缓急之分,所以队列中的文件有1~10不同的代先级,其中数字越大优先级越高。

打印机会从自己的待打印队列中选择*优先级最高*的文件来打印。

如果存在两个优先级一样的文件,则选择*最早进入队列*的那个文件。

现在请你来模拟这5台打印机的打印过程。

## 输入描述

每个输入包含1个测试用例,

每个测试用例第一行给出发生事件的数量N(0<N<1000)。

接下来有 N 行, 分别表示发生的事件。共有如下两种事件:

- "IN P NUM",表示有一个拥有优先级 NUM 的文件放到了打印机 P 的待打印队列中。(0< P <= 5, 0 < NUM <= 10);</li>
- 2. "OUT P",表示打印机 P进行了一次文件打印,同时该文件从待打印队列中取出。(0 < P <= 5)。

## 输出描述

- 对于每个测试用例,每次"OUT P"事件,请在一行中输出文件的编号。
- 如果此时没有文件可以打印,请输出"NULL"。
- 文件的编号定义为"IN P NUM"事件发生第 x 次,此处待打印文件的编号为x。编号从1开始。

## 用例

输入	7 IN 11 IN 12 IN 13 IN 21 OUT 1 OUT 2 OUT 2
输出	3 4 NULL
说明	无

输入	5 IN 11 IN 13 IN 11 IN 13 OUT 1
输出	2
说明	无

# 题目解析

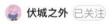
本题可以基于优先队列Q实现打印机总是打印优先级最高的文件。

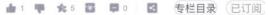
优先队列,如果想简单一点的话,则可以基于  $\frac{6}{6}$  有序数组  $\frac{6}{2}$  实现,但是有序数组是整体有序,每次有新任务入队,都需要 $\frac{6}{6}$  需要 $\frac{6}{6}$  的证明包杂度维持。

优先队列最好是基于堆结构实现,所谓堆结构,即一颗 $\frac{n}{n}$ 完全二叉树 $\frac{n}{n}$ 。本题是优先级数值越大,优先级越高,因此我们可以使用大顶堆。

关于基于堆结构实现优先队列, 可以参考

LeetCode\_1705 吃带里的是十颗日 化械之体的铺皮\_CSDNI铺皮





#### 题目解析

本题可以基于优先队列Q实现打印机总是打印优先级最高的文件。

优先队列,如果想简单一点的话,则可以基于有序数组Q实现,但是有序数组是整体有序,每次有新任务入队,都需要O(n)时间复杂度维持。

优先队列最好是基于堆结构实现,所谓堆结构,即一颗 完全 工 取 。本题是优先级数值越大,优先级越高,因此我们可以使用大顶堆。

关于基于堆结构实现优先队列, 可以参考

LeetCode - 1705 吃苹果的最大数目\_伏城之外的博客-CSDN博客

## JavaScript算法源码

基于有序数组实现优先队列

# 基于堆结构实现优先队列

```
if (lines.length === 1) {
   n = parseInt(lines[0]);
}
 swap(a, b) {
  const tmp = this.queue[a];
  this.queue[a] = this.queue[b];
  this.queue[b] = tmp;
          this.swap(c, f);
c = f;
} else {
break;
}
```

# Java算法源码

Java已经有优先队列实现类PriorityQueue, 因此可以直接使用它。

Java已经有优先队列实现类PriorityQueue, 因此可以直接使用它。

# Python算法源码

```
import queue

import queu
```