提高算法班

二叉堆、堆排序

Mas

二叉堆

堆(Heap) 是一类数据结构，它们拥有树状结构，且能够保证父节点比子节点大(或小)

当根节点保存堆中最大值时，称为大根堆; 反之，则称为小根堆

二叉堆(Binary Heap) 是最简单、常用的堆，是一棵符合堆的性质的完全二叉树

它可实现 $O(\log n)$ 插入或删除某个值，并且 $O(1)$ 查询最大/小值

作为完全二叉树二叉堆可用数组存储

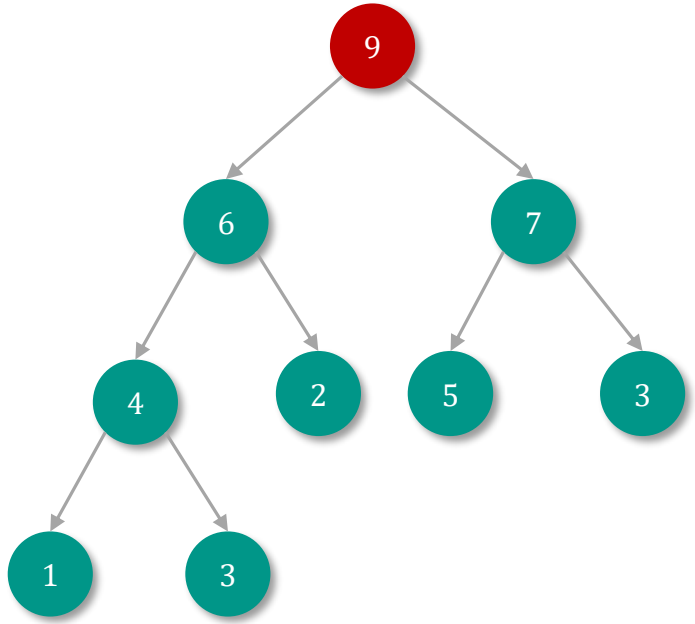
对于节点 p，其左儿子编号为 $2p$ 其儿子编号为 $2p+1$

设节点 p 位于第 x 层，那么其左右儿子位于第 $x+1$ 层

在第 x 层节点 p 前有 $p-2^x$ 个节点，p 之后存在 $2^{x+1}-1-p$ 个节点

节点 p 前的节点孩子(必有两个孩子)编号都小于 节点 p 的左孩子编号

其左孩子编号为 $1+p+2(p-2^x)+2^{x+1}-1-p=2p$



1	2	3	4	5	6	7	8	9
9	6	7	4	2	5	3	1	3

插入操作

二叉堆中插入一个元素，要保证插入后也是一棵完全二叉树

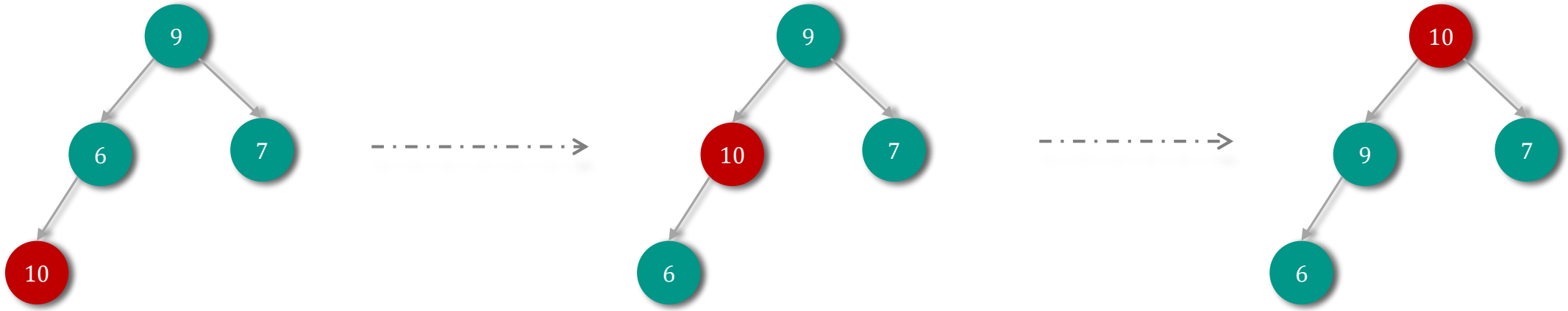
在最下一层最右边的叶子之后插入(若最下层已满则新增一层)

插入之后可能会不满足堆性质，需要**向上调整**

若该结点权值大于其父节点权值则交换，重复此过程直到不满足或者到根

插入之后向上调整后，没有其它结点违背堆性质

向上调整的时间复杂度是 $O(\log n)$



删除操作

删除操作指删除堆中最大/小的元素，即删除根结点

若直接删除则变成了两个堆，难以处理

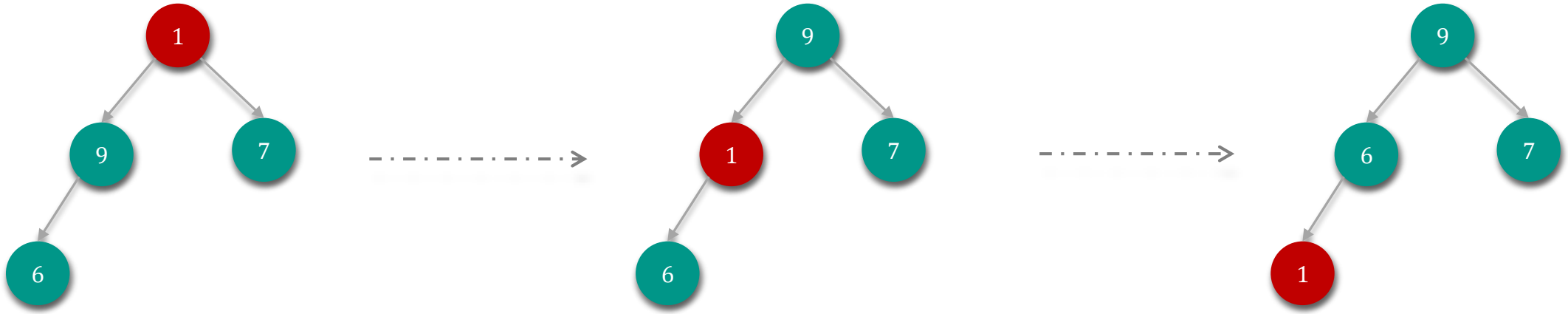
考虑将根结点和最后一个结点直接交换，再删掉 (已换至末尾) 根结点

但新的根结点可能不满足堆性质，考虑**向下调整**

在该结点的儿子中，找一个最大的与该结点交换，重复此过程直到底层

删除并向下调整后，没有其它结点违背堆性质

向下调整的时间复杂度是 $O(\log n)$



自顶向下建堆

从根节点开始,逐一插入堆的末尾,向上进行调整

设堆高度为 H 每层节点数为 n_i , 每层的高度定义为该层到堆的根节点的层数记为 h_i

每层的调整的时间复杂度为 $T(n_i) = n_i \times (h_i - 1)$, 则该建堆方式的时间复杂度

$$T(n) = \sum_{i=1}^{H} T(n_i) = \sum_{i=1}^{H} n_i \times (h_i - 1) = \sum_{i=1}^{H} 2^{i-1} \times (i - 1) = \sum_{i=0}^{H-1} 2^i \times i$$

上述为差比数列之和 , 错位相减法求得结果

$$T(n) = (H - 2) \times 2^H + 2$$

若该堆为满二叉树存在 $H = \log_2(n + 1)$ 即 $n + 1 = 2^H$ 则有

$$T(n) = (\log_2(n + 1) - 2) \times (n + 1) + 2$$

即时间复杂度 $O(n \log n)$

自底向上建堆

从倒数第二层的节点(叶子节点的上一层)开始,从右向左,从下到上的向下进行调整(可理解为每次合并两个合法的堆)

设该堆为满二叉树堆高为 H,假设每层高度为h_i,每层结点数为n_i, 则建堆复杂度为

$$T(n)=\sum_{i=1}^{H} n_i \times (H-h_i)=\sum_{i=1}^{H} 2^{i-1} \times (H-h_i)$$

上述为差比数列之和,错位相减法求得结果,即

$$T(n)=2^H-H-1$$

若该堆为满二叉树存在 $H=\log_2(n+1)$, 即 $n+1=2^H$ 则有

$$T(n)=n+1-\log_2(n+1)-1$$

即时间复杂度 $O(n)$

#2564、判断堆

题目描述

在计算机科学中,堆是一种的基于树的专用数据结构,它具有堆属性:

如果 P 是 C 的父结点

则在大顶堆中 P 结点的权值大于或等于 C 结点的权值

在小顶堆中 P 结点的权值小于或等于 C 结点的权值

一种堆的常见实现是二叉堆,它是由完全二叉树来实现的

你的任务是判断给定的完全二叉树是否是堆

输入格式

第一行包含两个整数 M 和 N ,分别表示给定完全二叉树的数量以及每个完全二叉树包含的结点数量

接下来 M 行,每行包含 N 个不同的整数(都在 int 范围内),表示一个完全二叉树的层序遍历序列

输出格式

对于每个给定的二叉树,首先输出一行对它是否是堆的判断结论

如果是大顶堆,则输出 `Max Heap` ,如果是小顶堆,则输出 `Min Heap` ,如果不是堆,则输出 `Not Heap`

然后,再输出一行它的后序遍历序列

输入样例

```
3 8
98 72 86 60 65 12 23 50
8 38 25 58 52 82 70 60
10 28 15 12 34 9 8 56
```

输出样例

```
Max Heap
50 60 65 72 12 23 86 98
Min Heap
60 58 52 38 82 70 25 8
Not Heap
56 12 34 28 9 8 15 10
```

数据范围

对于全部的数据 $1 \leq M \leq 100, 1 < N \leq 1000$

堆排序

堆排序(Heap Sort) 是利用二叉堆所实现的一种排序算法

先将待排序的序列构造成一个大根堆

将堆顶元素和最后一个元素交换，并维持剩余元素大根堆的性质

再将堆顶的元素取出作为次大值，与倒数第二位元素交换，并维持剩余元素大根堆的性质

在第 $n-1$ 次操作后，整个数组完成排序

堆排序可看作借助堆实现的选择排序

同选择排序一样，由于其交换位置的操作，是不稳定的排序算法

堆排序的最优时间复杂度、平均时间复杂度、最坏时间复杂度均为$O(n \log n)$

#2545、Insertion Or Heap

题目描述

根据 Wikipedia 描述:

插入排序(迭代) Insertion Sort

- 每次将一个插入元素插入到排好序的序列中
- 每次迭代插入排序都会从序列中移除一个元素,并在已排好序的序列中找到它所属的位置,然后将其插入
- **直到没有元素剩余为止**

堆排序 Heap Sort

- 将其序列分为**已排序**和**未排序**两个区域
- 并通过提取未排序区域中的最大元素并将其移至已排序的区域来迭代地缩小未排序的区域

> *它通过使用堆(Heap)而非线性时间搜索来找到最大值*

现在给定初始序列,以及经过某种排序方法多次迭代后的序列
请你判断我们使用的哪一种排序方法

数据范围

对于全部的数据 $1 \leq N \leq 100$,序列元素值不超过 100000

输入格式

第一行包含整数 N ,表示序列中整数个数

第二行包含 N 个整数表示初始序列

第三行包含 N 个整数表示经过若干次迭代后的序列

假定排序的目标序列总是递增的

输出格式

第一行输出 `Insertion Sort` 或 `Heap Sort` ,以指明所采用的具体排序方法

运用此方法再进行一次迭代,并在第二行输出本次迭代后的序列

数据保证答案唯一

#230、第K小年龄和

题目描述

今年的 $PION$ 比赛开设了一个特别的赛区,在这个赛区里参赛的选手可以两人一队,进行组队赛

有趣的是组队赛是可以初中和高中的学生共同参加的

因此队伍之间的水平评定就变的很困难, 两个高中学生比两个初中学生组成的队伍要有很大优势,但是也有初中生就很强的选手

同时为了防止强强联手虐场的情况,组委会随机将所有人分成两组,每一个 A 组的选手只能和 B 组的选手组队, B 组也一样

现在作为教练的江哥有一套神奇的计算体系,他发现如果一组组合方式在所有可能的组队情况中,年龄的和从小到大如果排在第 K 位,获胜的概率是最大的

现在江哥想知道,在所有可能的组合中,升序排序后排在第 K 位的年龄和是多少

输入格式

第一行一个整数 N K ,表示 A 和 B 组各有 N 个人,江哥想知道第 K 大的年龄和是多少

接下来两行,每行 N 个数。分别是 A B 组每个人的年龄

每个人的年龄都在 32 位整数范围内

输出格式

输出一个整数, 表示第 K 大的年龄和

输入样例

```
3 3
3 2 3
5 3 4
```

输出样例

```
6
```

数据规模

对于全部的数据 $0 < N \leq 200000, 0 < K \leq 10^6$

#230、第K小年龄和

将 a 和 b 排序，最小和为 $a[1]+b[1]$

$$a[L+2]+b[R] \geq a[L+1]+b[R] \geq a[L]+b[R]$$

$$a[L]+b[R+2] \geq a[L]+b[R+1] \geq a[L]+b[R]$$

使用小根堆根据和维护三元组 $\{L,R,a[L]+b[R]\}$

每次从堆中取出一个节点 $\{L,R,a[L]+b[R]\}$，并加入$\{L+1,R,a[L+1]+b[R]\}$ 和 $\{L,R+1,a[L]+b[R+1]\}$

取出的第 K 个元素即为答案，时间复杂度$O(n\log n)$

如何证明上述做法正确性?

$a[L-1]+b[R]$ 和 $a[L]+b[R-1]$ 都能产生 $a[L]+b[R]$ 需要避免重复入堆

对 $\{L,R\}$ 做哈希?

规定只有当 $R=1$ 的时候才会额外转移 $\{L+1,R\}$ 否则只转移 $\{L,R+1\}$

如何证明不重不漏且顺序正确?

#2501、前缀中位数

题目描述

给出一个整数序列 $a_1, a_2, \cdots, a_n$,计算出前一个数,前三个数,前五个数直到前 n 个数的中位数

所谓一些数的中位数,就是这些数字排序后位置在最中间的数

保证 n 是一个奇数

输入格式

第一行：单个整数表示 n ,保证 n 是奇数

第二行： n 个整数表示 $a_1, a_2, \cdots, a_n$

输出格式

共 $\frac{n+1}{2}$ 行：第 i 行表示前 $2i-1$ 个数字的中位数

数据规模

对于 10% 的数据,满足 $1 \leq n \leq 100$

对于 30% 的数据,满足 $1 \leq n \leq 1000$

对于 前 50% 的数据,数据较为随机,满足 $1 \leq n \leq 30000$

对于 100% 的数据,满足 $1 \leq n \leq 200000$

对于全部的数据 $0 \leq a_i \leq 10^9$

输入样例1

```
7
1 3 5 7 9 11 6
```

输出样例1

```
1
3
5
6
```

#2501、前缀中位数

思路1

问题可抽象成: 动态维护一个序列上第 K 大数(K 值可能会发生变化)

可使用对顶堆维护，对顶堆由一个大根堆与一个小根堆组成

小根堆维护大值即前 K 大的值(包含第 K 个)，大根堆维护小值即比第 K 大数小的其他数

这两个堆构成的数据结构支持以下操作:

- **维护**

  当小根堆的大小小于 K 时不断将大根堆堆顶元素取出并插入小根堆,直到小根堆的大小等于 K

  当小根堆的大小大于 K 时不断将小根堆堆顶元素取出并插入大根堆,直到小根堆的大小等于 K

- **插入元素**

  若插入的元素大于等于小根堆堆顶元素，则将其插入小根堆

  否则将其插入大根堆，然后维护对顶堆

#2501、前缀中位数

- **查询第 K 大元素**

  小根堆堆顶元素即为所求

- **删除第 K 大元素**

  删除小根堆堆顶元素，然后维护对顶堆

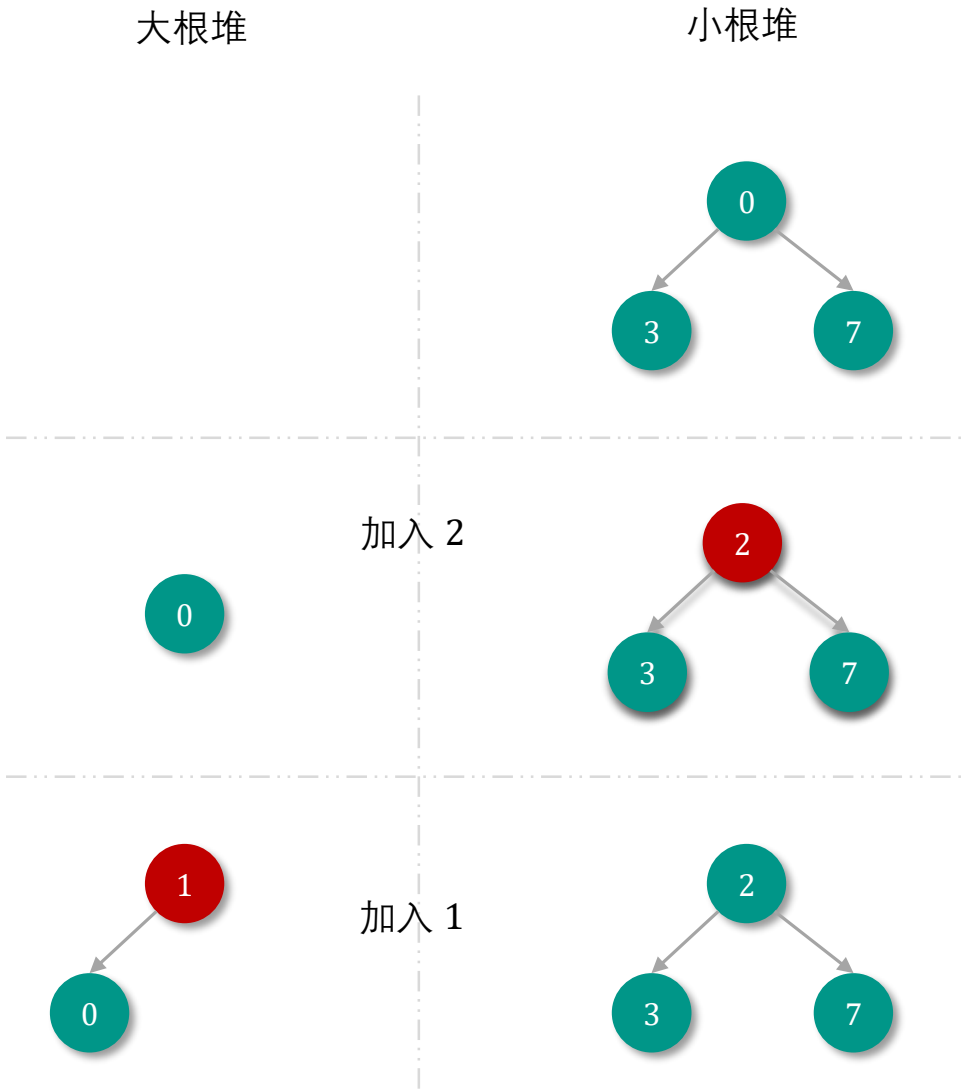
- **K 值 $+1/-1$**

  根据新的 K 值直接维护对顶堆

由于插入、删除或调整 K 值后，小根堆的大小与期望的 K 值最多相差 1

每次维护最多只需对大根堆与小根堆中的元素进行一次调整

总时间复杂度 $O(n \log n)$



#2501、前缀中位数

思路2

对数组 a 排序后的数组记为 S，最后一次询问必然输出 $S_{\frac{1+n}{2}}$

若删去 a_n 与 a_{n-1} 问题规模减为 $n-2$

利用 S 建立双向链表 L，同时令 idx_i 表示 a_i 在链表中的位置

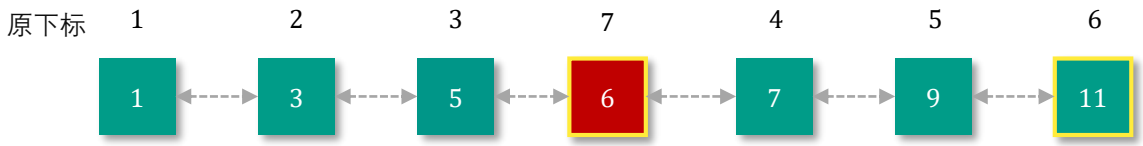
初始时令 $m=\frac{1+n}{2}$ 每次删除下标最大的两个元素

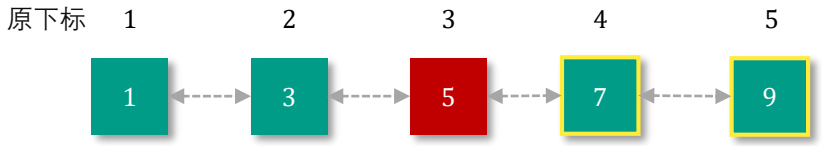
对于 m 需要分情况讨论

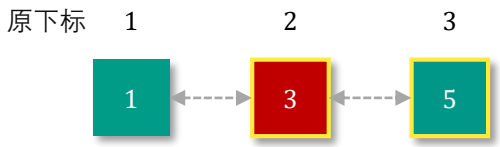
- 两元素在链表中的位置都 $\geq m$，令 $m \leftarrow \mathrm{pre}(m)$
- 两元素在链表中的位置都 $\leq m$，令 $m \leftarrow \mathrm{nxt}(m)$
- 否则 m 不变

倒序删除记录 a_m 的值即为答案

时间复杂度 $O(n \log n)$







#1053、工作调度

题目描述

FJ 有太多的工作要做啊!
为了让农场高效运转,他必须靠他的工作赚钱,每项工作花一个单位时间

他的工作日从 0 时刻开始,有 10^9 个单位时间

在任一时刻,他都可以选择编号 $1 \sim N$ 的 项工作中的任意一项工作来完成

因为他在每个单位时间里只能做一个工作,而每项工作又有一个截止日期
所以他很难有时间完成所有 N 个工作,虽然还是有可能

对于第 i 个工作,有一个截止时间 D_i ,如果他可以完成这个工作,那么他可以获利 P_i

在给定的工作利润和截止时间下, FJ 能够获得的利润最大为多少呢?

输入格式

第 1 行输入一个整数 N
第 $2 \sim N+1$ 行,第 $i+1$ 行有两个用空格分开的整数 D_i, P_i

输出格式

输出一行一个整数,表示最大获利值

数据规模

对于全部的数据 $1 \leq N \leq 10^5, 1 \leq D_i, P_i \leq 10^9$

若以 DDL 为第一关键字

利润为第二关键字排序

顺序向后贪心选取任务，该策略显然错误

当前的最优解可能不是全局最优解

```
4
1 2
1 8
2 100
2 76
```

#1053、工作调度

思路1

若发现最新任务因为 DDL 无法选取，需要取消(反悔)之前一个任务的选取

显然反悔最小价值的任务

按 DDL 升序排序，用小根堆维护选取过的任务价值

- 当前任务 DDL 大于堆大小，说明当前任务可选取

  直接累加任务价值，并将当前任务价值入队

- 当前任务截至时间不大于堆大小，说明选取当前任务必须舍弃之前选取的某个任务

  若堆内最小价值小于当前任务价值

  减去堆内最小价值累加当前任务价值(实现反悔)

  否则不做处理

时间复杂度$O(n \log n)$

#1053、工作调度

思路2

也可不考虑反悔，直接用 贪心+堆 完成该题

按 DDL 升序排序后逆序枚举时间，维护一个初始为空的堆

- 当枚举到时刻 T 时，把所有 DDL 等于 T 的物品一起加入考虑(一起放入堆中)
- 时刻 T 只能做一个任务，那肯定是做价值最大的任务，查询并删除堆顶

当 T 枚举至 0 时算法结束，此时堆中的物品只能放弃

T 的范围虽然很大，但有效的 T 至多只有 n 个

时间复杂度 $O(n \log n)$

#3003、种树

题目描述

A城市有一个巨大的圆形广场,为了绿化环境和净化空气,市政府决定沿圆形广场外圈种一圈树

园林部门得到指令后,初步规划出 n 个种树的位置,顺时针编号 1 到 n

并且每个位置都有一个美观度 A_i ,如果在这里种树就可以得到这 A_i 的美观度

但由于 A 城市土壤肥力欠佳,两棵树决不能种在相邻的位置(i 号位置和 $i+1$ 号位置叫相邻位置

值得注意的是 1 号和 n 号也算相邻位置)

最终市政府给园林部门提供了 m 棵树苗并要求全部种上,请你帮忙设计种树方案使得美观度总和最大

如果无法将 m 棵树苗全部种上,给出无解信息

输入格式

输入的第一行包含两个正整数 n , m

第二行 n 个整数,第 i 个代表 A_i

输出格式

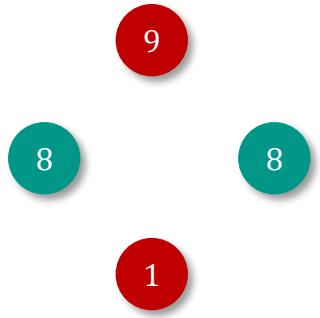
输出一个整数,表示最佳植树方案可以得到的美观度

如果无解输出 `Error!`

数据规模

对于全部数据: $1 \leq n \leq 200000, m \leq n, -1000 \leq A_i \leq 1000$

当 $m > \frac{n}{2}$ 时显然无解



考虑贪心

利用大根堆根据价值维护所有树坑

每次取出堆顶且未标记的节点

累加价值，并标记其左右节点

不难发现上述策略显然错误

#3003、种树

考虑dp

设 $dp[i][j]$ 表示前 i 个坑种了 j 棵树的最大价值

$$dp[i][j] = \max_{0 \le j \le i}\{dp[i-1][j], dp[i-2][j-1] + a[i]\}$$

由于树坑为环形

- 最后一个树坑不选择

  从 $1 \sim n-1$ 求 , 此时待选答案为 $dp[n-1][m]$

- 最后一个树坑选择

  从 $2 \sim n$ 求 ,此时待选答案为 $dp[n][m]$

两遍中的 $\max(dp[n-1][m], dp[n][m])$ 即为最终答案

时/空间复杂度 $O(nm)$ 无法通过

#3003、种树

考虑反悔贪心

依然利用大根堆维护，取出堆顶未标记的节点

记当前取出节点权值为 val_x，左侧未选取节点权值为 val_{pre}，右侧未选取节点权值为 val_{nxt}

在累加 val_x 时，同时加入权值为 $val_{pre}+val_{nxt}-val_x$ 的节点

若下次能选中 $val_{pre}+val_{nxt}-val_x$ 则相当于取消选择 val_x 并选择 $val_{pre}+val_{nxt}$

由于堆内节点可能不是一个树坑(可能为替代的树坑组合)

选取当前节点后需要找到其 左/右 第一个未被标记的树坑标记

可建立双向链表优化

在选取当前节点后仅需将前驱和后继节点从链表中删除并标记

时间复杂度 $O(n\log n)$

谢谢观看