二叉堆

南开中学信息学竞赛教练组

PART

01

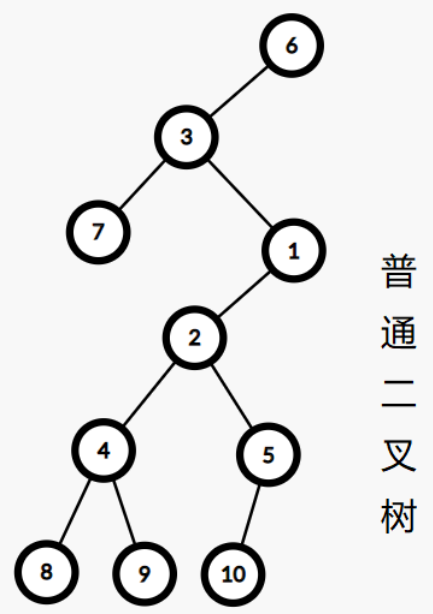
概念

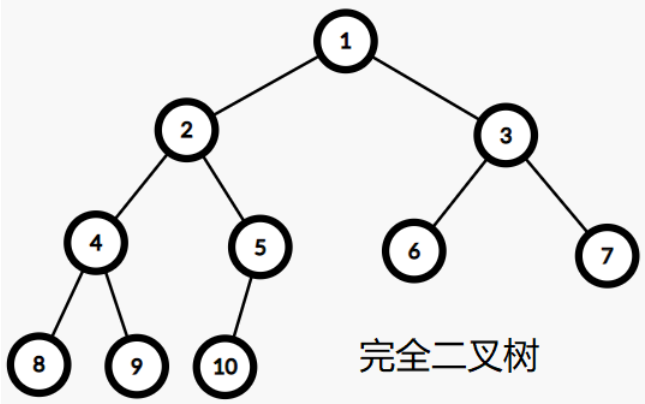
- 二叉堆

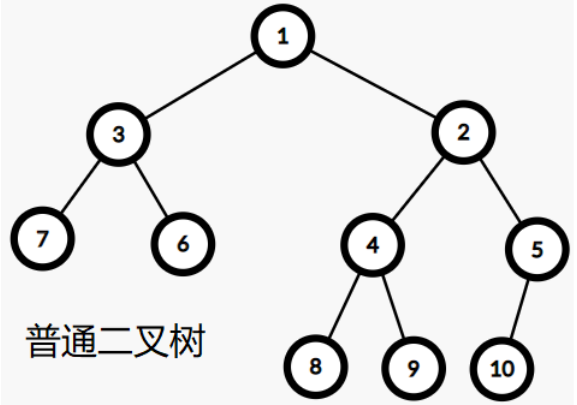
二叉树

完全二叉树：

- 除了最底层，其他层全满，且最底层的节点全部靠左排布的二叉树。



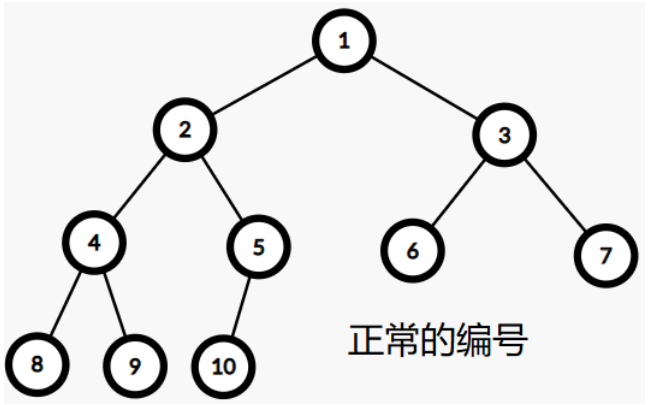


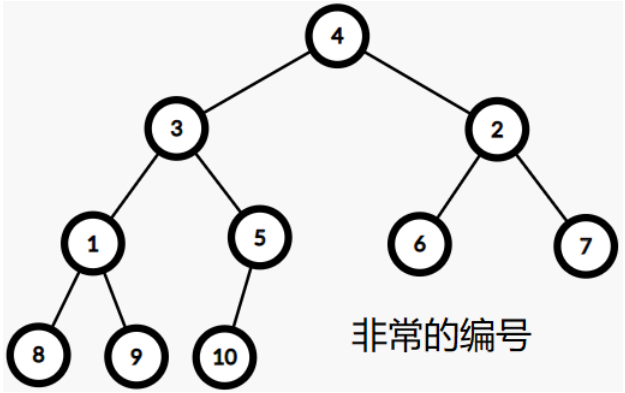


二叉树

完全二叉树：

- 除了最底层，其他层全满，且最底层的节点全部靠左排布的二叉树。
- 一般按照“从上到下，从左到右”的顺序编号。

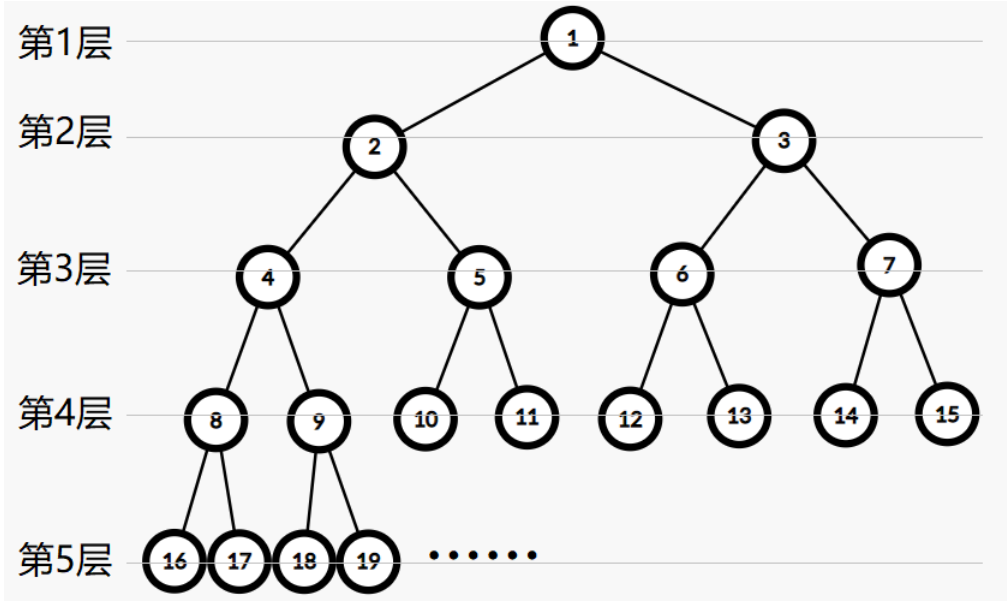




二叉树

完全二叉树：

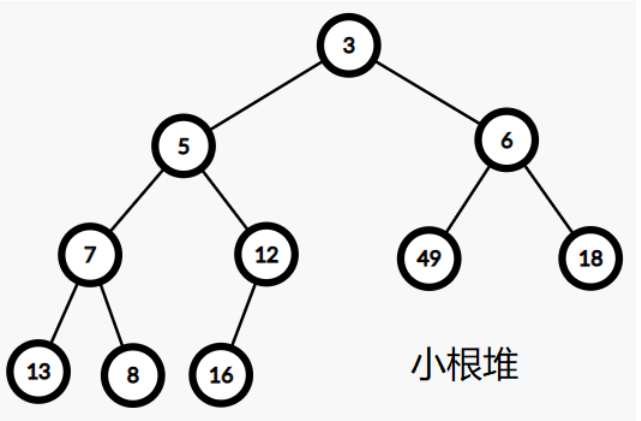
- 除了最底层，其他层全满，且最底层的节点全部靠左排布的二叉树。
- 一般按照“从上到下，从左到右”的顺序编号。
- 第 k 层有 2^{k-1} 个节点。
- 第 k 层编号从 2^{k-1} 到 $2^k - 1$ 。
- 共 n 个节点时，有 $\lfloor \log_2 n \rfloor + 1$ 层。
- 节点 x 的左右儿子分别是 $2x, 2x+1$，父亲是 $\lfloor x/2 \rfloor$。
- 可以用一维数组存储。

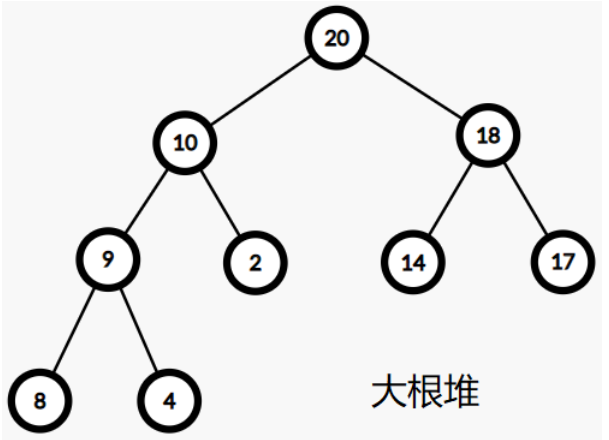


二叉堆

二叉堆：

- 简称“堆”
- 在完全二叉树上存储一些数据，满足“堆性质”：
  - 每个节点的数据都小于(或大于)父亲的数据。
- 分为小根堆和大根堆。
- 根节点称为“堆顶”，它的数据最小(或最大)。





二叉堆 ？优先队列

优先队列：

- 维护一些数值，需要支持以下基本功能：
  - 插入一个数
  - 查询最小的数
  - 删除最小的数
- 且保持操作的高效性。

- 二叉堆：
  - 满足功能需求
  - 效率较高
  - 简单，容易实现

PART

02

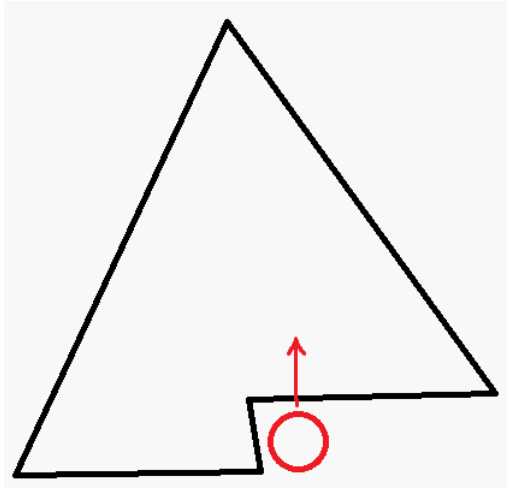
基本操作

- 插入
- 删除堆顶

流程

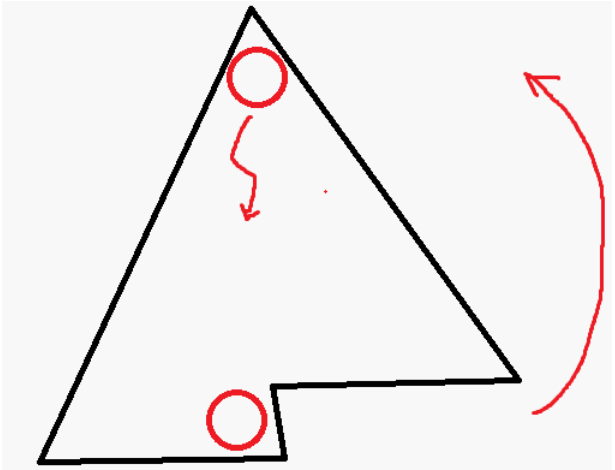
插入：

- 将需要插入的数据放在最底层末尾位置
- 逐层向上调整堆



删除：

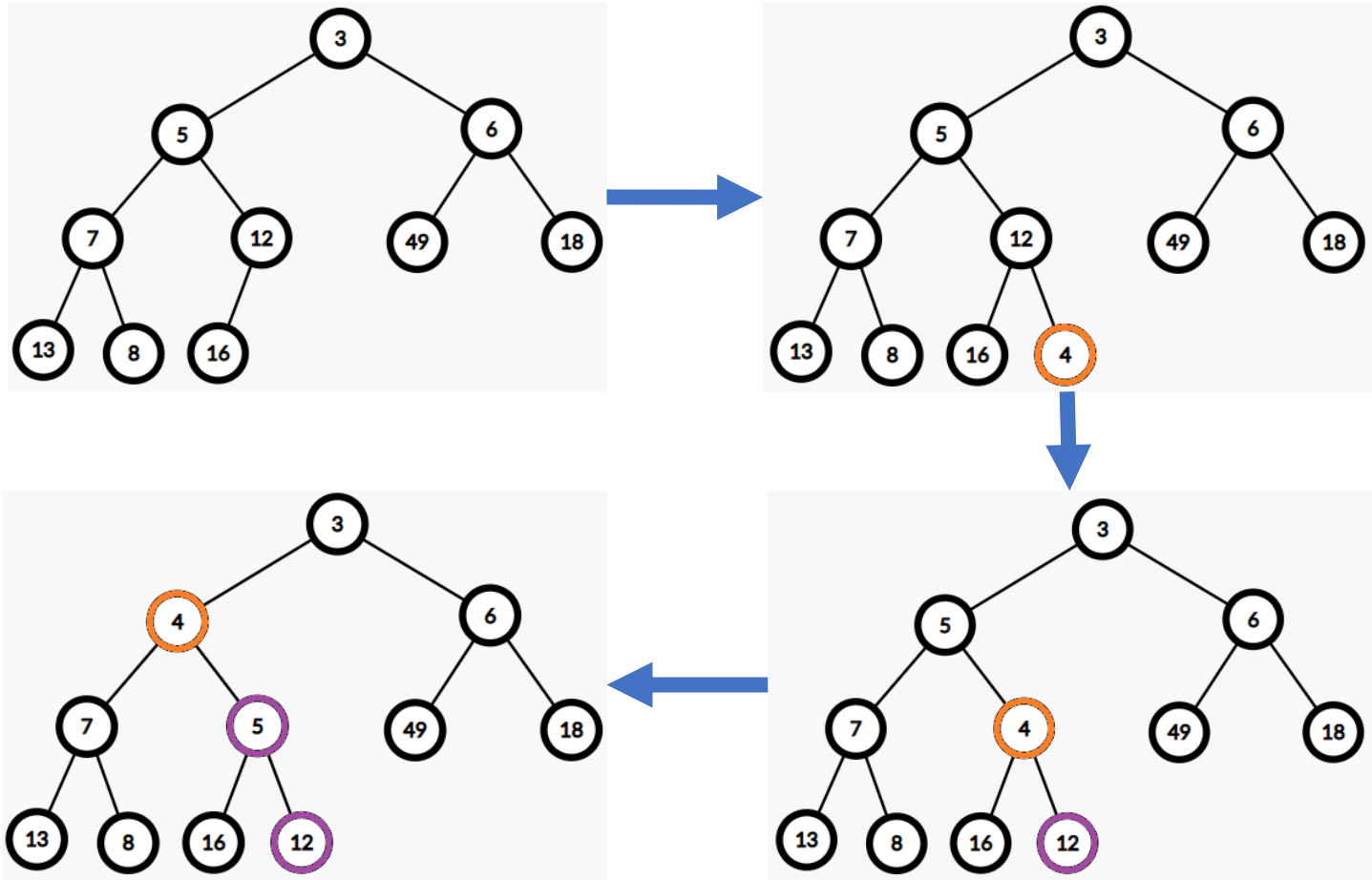
- 将最底层末尾的数放到堆顶
- 逐层向下调整堆



插入操作

流程：

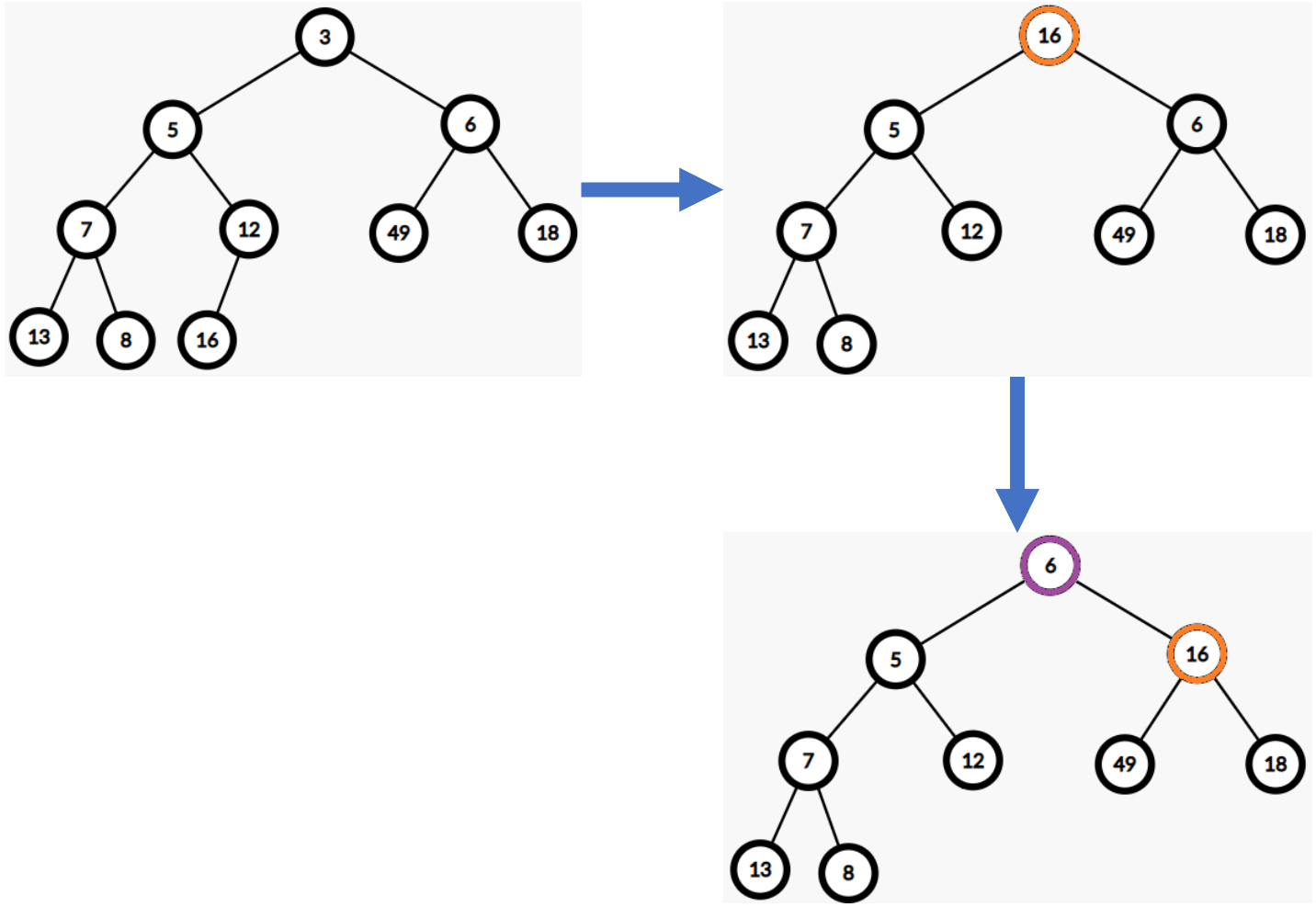
- 小根堆
- 在末尾插入“4”
- 向上调整
- 向上调整
- 4⩾3，不用再调整，停止



删除操作

流程：

- 小根堆
- 把末尾放到堆顶
- 向下调整（因为5＜6所以与右儿子交换）
- 16≤49且16≤18，不用再调整，停止



PART

02

基本操作

- 插入
- 删除堆顶
- 建堆

建堆

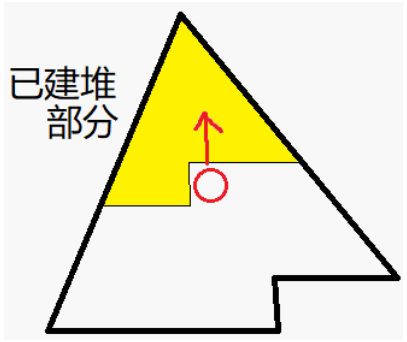
目标：

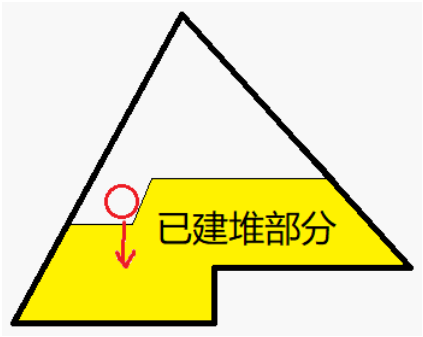
- 数据散乱分布在数组中，下标从 1 到 n
- 如何操作，才能变成一个堆？

- 两种思路
  1. 从上到下的顺序，向上调整

     每时每刻，上半部分是一个堆
  2. 从下往上的顺序，向下调整

     每时每刻，下半部分是很多个堆

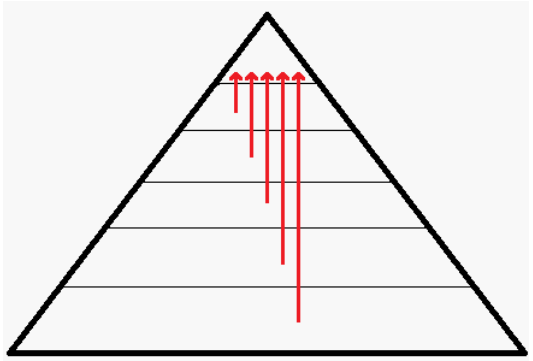




建堆

建堆●向上：

- 假设有$n = 2^k - 1$个节点（满二叉树）
  - 第1层有1个节点，每个节点至多上调0层
  - 第2层有2个节点，每个节点至多上调1层
  - 第3层有4个节点，每个节点至多上调2层
  - ……
  - 第k层有2^{k-1}个节点，每个节点至多上调$k-1$层
- 时间复杂度？
  - $O(n \log n)$
  - 还不如直接sort呢

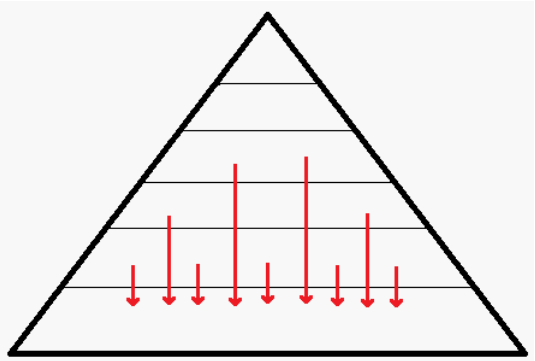


$$
\begin{aligned}
T(n) &= O\left(2^0 \times 2^0 + 2^1 \times 1 + 2^2 \times 2 + \cdots + 2^{k-1} \times (k-1)\right) \\
T(n) &\le O\left((2^0 + 2^1 + \cdots + 2^{k-1}) \times (k-1)\right) \\
&= O(n \times (k-1)) \\
&= O(n \log n) \\
T(n) &\ge 2^{k-1} \times (k-1) \\
&= O(\frac{n}{2} \times (k-1)) \\
&= O(n \log n)
\end{aligned}
$$

建堆

建堆●向下：

- 假设有$n = 2^k - 1$个节点（满二叉树）
  - 第1层有1个节点，每个节点至多上调$k-1$层
  - 第2层有2个节点，每个节点至多上调$k-2$层
  - 第3层有4个节点，每个节点至多上调$k-3$层
  - ……
  - 第k层有2^{k-1}个节点，每个节点至多上调0层
- 时间复杂度？
  - $O(n)$



$$T(n) = O\left(2^0 \times (k-1) + 2^1 \times (k-2) + \cdots + 2^{k-2} \times 1 + 2^{k-1} \times 0\right)$$

$$T(n) = O\begin{pmatrix} +2^{k-2} & & & & & & \\ +2^{k-3} & +2^{k-3} & & & & & \\ +2^{k-4} & +2^{k-4} & +2^{k-4} & & & & \\ \cdots & & & & & & \\ +2^1 & +2^1 & +2^1 & \cdots & +2^1 & & \\ +2^0 & +2^0 & +2^0 & \cdots & +2^0 & +2^0 \end{pmatrix}$$

$$= O\left((2^{k-1}-1) + (2^{k-2}-1) + \cdots + (2^2-1) + (2^1-1)\right)$$

$$= O\left(2^k - (k-1)\right)$$

$$= O(n)$$

PART

02

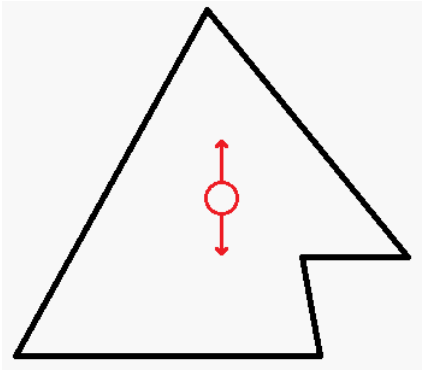
基本操作

- 插入
- 删除堆顶
- 建堆
- 其他操作

改数

操作：

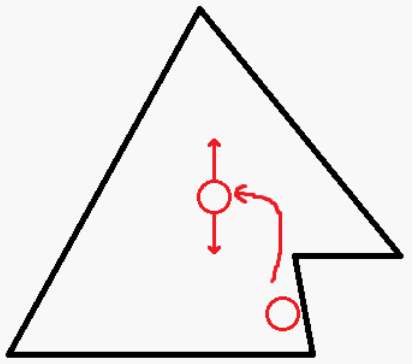
- 要修改一个数，先要知道它在堆中的位置
- 直接修改，然后从这个位置开始，向上调整或者向下调整



删数

操作：

- 要删除一个数，先要知道它在堆中的位置
- 将末尾的数放到这个位置，然后向上调整或者向下调整



参考代码

不需要！

```
int n, heap[233];
//小根堆，向上调整
void shift_up(int i) {
    int t = heap[i], j = i >> 1;
    while (j >= 1) {
        if (t < heap[j]) {
            heap[i] = heap[j];
            i = j;
            j = i >> 1;
        } else {
            break;
        }
    }
    heap[i] = t;
}
```

```
//小根堆，向下调整
void shift_down(int i) {
    int t = heap[i], j = i << 1;
    while (j <= n) {
        if (j < n && heap[j] > heap[j + 1]) {
            j ++;
        }
        if (t > heap[j]) {
            heap[i] = heap[j];
            i = j;
            j = i << 1;
        } else {
            break;
        }
    }
    heap[i] = t;
}
```

```
int main() {
    //建堆
    for (int i = n >> 1; i >= 1; i --) {
        shift_down(i);
    }
    //插入666
    heap[++ n] = 666;
    shift_up(n);
    //删除堆顶
    heap[1] = heap[n --];
    shift_down(1);
    return 0;
}
```

基本操作

- 插入
- 删除堆顶
- 建堆
- 其他操作
- C++的优先队列

C++的优先队列

使用方法：

- 头文件：
- 声明大根堆：
- 声明小根堆：
- 声明时用数组数据建堆：

```
#include <queue>
priority_queue<int> pq;
priority_queue<int, vector<int>, greater<int> > pq;
int x[10] = {2, 5, 1, 9, 3, 8, 3, 2, 1, 1};
priority_queue<int> pq(x, x + 10);  //参数和sort相同
```

- 插入666：
- 获取并输出堆顶：
- 删除堆顶：
- 当前的元素个数：
- 判断是否为空：

```
pq.push(666);
cout << pq.top();
pq.pop();
cout << pq.size();
if (pq.empty()) ...
```

- 使用 priority_queue<结构体> 时，要重载“<”运算符。

PART

03

例题

- 若干

例题

例题【NKOJ1760 逃亡】：

何老板抢劫了银行，他驾驶一辆卡车逃跑。卡车每行驶一公里就会消耗掉一升油。
何老板需要把车开到最近的一个小镇，那里有同伙接应。从卡车当前位置到小镇的这条路上，分布着N个加油站，何老板可以停下来加油，但是这条路上每个加油站的油量不超过100升。
警察正在追捕何老板，所以，何老板想让停下来加油的次数尽可能的少。幸运的是，卡车的油箱容量无限大。一开始卡车的油箱存有P升油，距离目标小镇L公里远。请计算何老板最少停下来加油几次？

输入格式：

第一行，一个整数N。
接下来N行，每行两个整数，描述一个加油站的信息：第一个数表示加油站到目标小镇的距离，第二个数表示加油站可用的油量。
接下来一行，两个空格间隔的整数L和P。

$0 <= L <= 1,000,000$
$1 <= N <= 10,000$
$1 <= P <= 1,000,000$

样例输入

```
4
4 4
5 2
11 5
15 10
25 10
```

样例输入

```
2
```

例题

例题【NKOJ1760 逃亡】：

解析：

- 思想：
  - 贪心，跑尽量远
  - 油不够了，就选择油最多的加油站。
  - 选择“能到达且未被选择的加油站中油最多的”
- 怎么选择呢？堆！
  - 将加油站排序，将能直接到达的放入大根堆；
  - 如果没到终点，选最大，并从堆中删除；
  - 将新增的可选加油站加入堆；
  - 重复，直到到达终点。
- 时间复杂度$O(n \log n)$。

例题

例题【NKOJ2683 Cow Lineup】：

农夫约翰的N只奶牛排成了一队，每只牛都用编上了一个“血统编号”。血统相同的奶牛有相同的编号，也就是可能有多头奶牛是相同的"血统编号"。
约翰觉得如果连续排列的一段奶牛有相同的血统编号的话，奶牛们看起来会更具有威猛。为了创造这样的连续段，约翰最多能选出K种血统的奶牛，并把他们全部从队列中赶走。请帮助约翰计算这样做能得到的由相同血统编号的牛构成的连续段的长度最大是多少？

输入格式：

第一行，两个空格间隔的整数N和K
接下来N行， 每行一个整数，表示对应奶牛的血统编号

1<=N<=100000
K<=N
1<=编号<=10^9

样例输入
9 1
2
7
3
7
7
3
7
5
7

样例输入
4

例题

例题【NKOJ2683 Cow Lineup】：

解析：

- 赶走K种牛，剩下1种牛连续最长
- 这个连续段中，一开始不超过K+1种牛

- 找一个不超过K+1种牛的连续段　　——双指针扫描
- 留下哪一种呢？肯定是最多的一种　　——堆

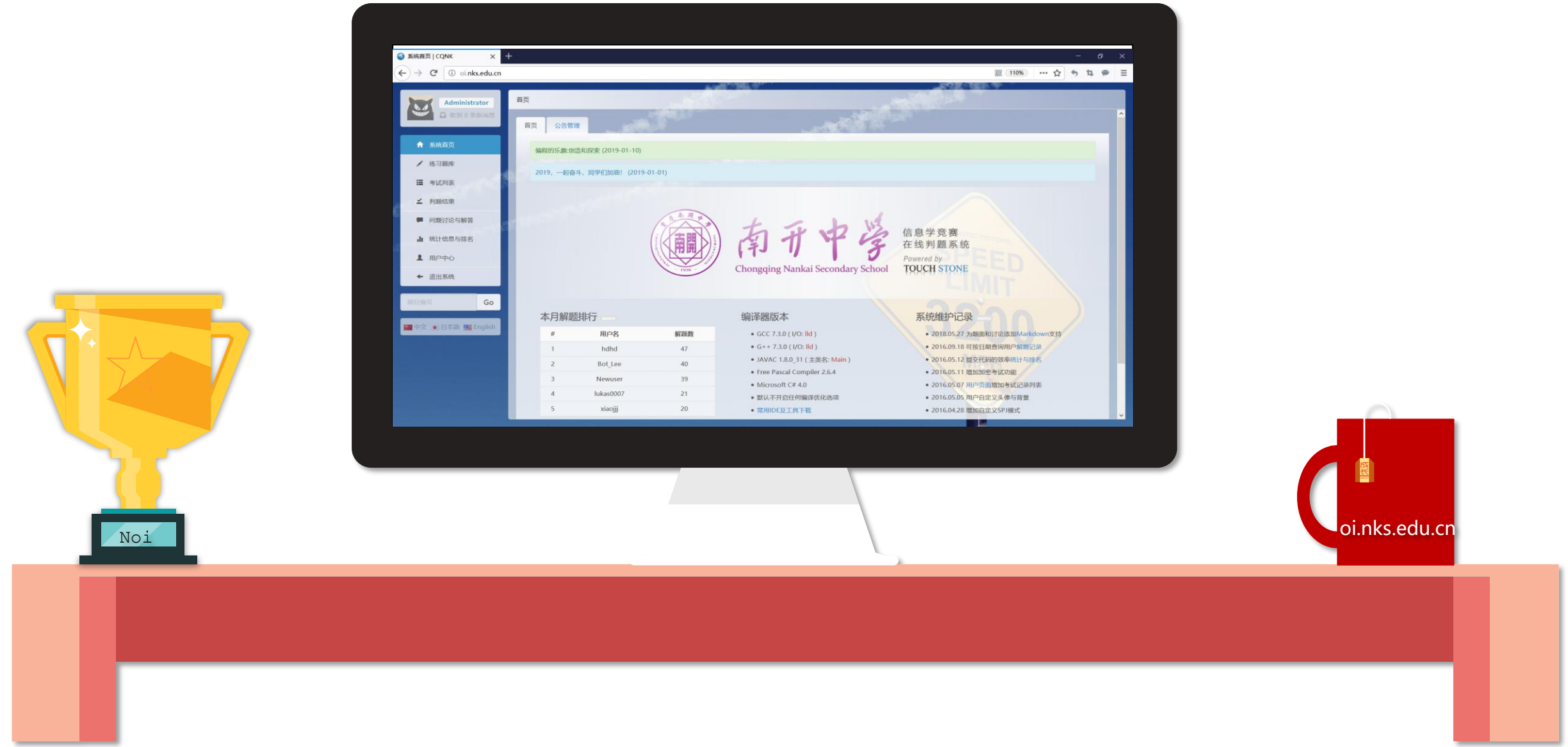
- 时间复杂度$O(n \log n)$。

课后习题

课后习题：

考试名称：愉快の夏季2

考试密码：SUMMERNK


系统首页 | CQNK
oi.nks.edu.cn
Administrator
首页
公告管理
系统首页
练习题库
考试列表
判题结果
问题讨论与解答
统计信息与排名
用户中心
退出系统
Go
编程的乐趣:创造和探索 (2019-01-10)
2019，一起奋斗，同学们加油！ (2019-01-01)
南开中学
Chongqing Nankai Secondary School
信息学竞赛
在线判题系统
Powered by
TOUCH STONE
本月解题排行
用户名 解题数
1 hdhd 47
2 Bot_Lee 40
3 Newuser 39
4 lukas0007 21
5 xiaojjj 20
编译器版本
GCC 7.3.0 (I/O: lld)
G++ 7.3.0 (I/O: lld)
JAVAC 1.8.0_31 (主类名: Main)
Free Pascal Compiler 2.6.4
Microsoft C# 4.0
默认不开启任何编译优化选项
常用IDE及工具下载
系统维护记录
2018.05.27 为题面和讨论添加Markdown支持
2016.09.18 可按日期查询用户解题记录
2016.05.12 提交代码的效率统计与排名
2016.05.11 增加加密考试功能
2016.05.07 用户页面增加考试记录列表
2016.05.05 用户自定义头像与背景
2016.04.28 增加自定义SPJ模式
Noi
oi.nks.edu.cn