Tutorial-3

前k大元素，sorted

> “排序是选择最坏的情况”

1. $n \log n$，全排序
2. partition - $O(n)$/construct_heap - $O(n)$/selection - $O(n)$

> “压缩真正问题的空间”

算导 4-5 Chip testing

CLRS Page 109

> Diogenes 教授有n个被认为是完全相同的集成电路芯片，原则上它们是可以互相测试的。教授的测试装置一次可测试二片，当该装置中放有两片芯片时，每一片就对另一片作测试并报告其好坏。一个好的芯片总能够正确的报告另一片的好坏，但一个坏的芯片的结果就是不可靠的。这样,每次的测试的四种可能结果如下:
>
> | A 芯片报告 | B 芯片报告 | 结论 |
> |---|---|---|
> | B 是好的 | A 是好的 | 都是好的，或都是坏的 |
> | B 是好的 | A 是坏的 | 至少一片是坏的 |
> | B 是坏的 | A 是好的 | 至少一片是坏的 |
> | B 是坏的 | A 是坏的 | 至少一片是坏的 |
>
> a) 证明若大于 n/2 的芯片是坏的，在这种成对测试方式下，使用任何策略都不能确定哪些芯片是好的。Hint: 假设坏芯片可以一起愚弄 (conspire to fool) 这个教授。
>
> b) 假设有大于 n/2 的芯片是好的，考虑从 n 片中找出**一片**好芯片的问题。证明 $\lfloor n/2 \rfloor$ 次测试就足以使问题的规模降至近原来的一半。
>
> c) 假设有大于 n/2 的芯片是好的，证明好的芯片可用 O(n) 对测试找出。

解答

> 只要有坏的就全扔

a) 证明不可判只需要考虑最坏情况——坏的芯片永远做出与真相相反的判断 (mirrors the strategy used by the good chips)，即好芯片都被判断为坏的，而坏芯片都被判断为好的。这种情况下，任何策略判断出的“好”芯片可能是真的好芯片，也可能是坏芯片，从而不可判。

b) 任意配对芯片，只要两片芯片的判断里出现坏的就两片全扔，由于鸽巢原理，当好芯片大于n/2时，必定至少有一组两片都是好的，从而剩下。同时，当好芯片数大于n/2时，剩下的"好"芯片中，至少有一半声称对方是"好"芯片的真的就是好芯片。从而，只要从剩下的每对中选一片，就可以构造出规模将近原先一半的子问题。

c) 一旦我们找到了一个好芯片，我们就可以用它来判断别的芯片的好坏。于是，找到一个好芯片需要的的递推式为

$$T(n) \leq T(n/2) + n/2$$

用Master定理可知复杂度为$\Theta(n)$。因此，我们在$O(n)$次的成对测试后就可以解决这个问题。鉴于我们也需要查看至少一半的芯片， 我们知道这个问题也是$\Omega(n)$复杂度的。

中位数最近的k个元素

> "我们把这一半，bong，搬到另一边来"

规约过程

查找√n

折半

折半猜

$$y = x^2$$

$$N = \left(\sqrt{N}\right)^2$$

$y = x^2$是抛物线

牛顿迭代法→梯度下降（机器学习）

求导

> "现在全世界所有人都在用梯度下降做优化问题"

带权中位数，算法导论9-2

CLRS Page 225

9-2 Weighted median

For n distinct elements $x_1, x_2, \ldots, x_n$ with positive weights $w_1, w_2, \ldots, w_n$ such that $\sum_{i=1}^{n} w_i = 1$, the ***weighted (lower) median*** is the element x_k satisfying

$$\sum_{x_i < x_k} w_i < \frac{1}{2}$$

and

$$\sum_{x_i > x_k} w_i \le \frac{1}{2} .$$

For example, if the elements are $0.1, 0.35, 0.05, 0.1, 0.15, 0.05, 0.2$ and each element equals its weight (that is, $w_i = x_i$ for $i = 1, 2, \ldots, 7$), then the median is 0.1, but the weighted median is 0.2.

a. Argue that the median of $x_1, x_2, \ldots, x_n$ is the weighted median of the x_i with weights $w_i = 1/n$ for $i = 1, 2, \ldots, n$.

b. Show how to compute the weighted median of n elements in $O(n \lg n)$ worst-case time using sorting.

c. Show how to compute the weighted median in $\Theta(n)$ worst-case time using a linear-time median algorithm such as SELECT from Section 9.3.

解答

a. 设m_k为小于x_k的x_i的数量。当给每个x_i赋上权重$1/n$时，有$\sum_{x_i<x_k} w_i = m_k/n$以及$\sum_{x_i>x_k} w_i = (n-m_k-1)/2$。能使这两个sum满足$<2$和$\le 2$的唯一一个$m_k$是$\lceil n/2 \rceil - 1$，显然这个时候$x_k$为中位数。

b. 先用`mergesort`对x_i们按值排序，时间复杂度为$O(n\log n)$。设S_i为排完序的数组中前i个元素的权重和，每次更新S_i的时间为$O(1)$。连续计算$S_1, S_2, \cdots$直到第k次使得$S_{k-1} < 1/2$且$S_k \ge 1/2$。从而加权中位数就是x_k。

c. 通过改`SELECT`来达到线性时间解决问题。设x为中位数的中位数。计算$\sum_{x_i<x} w_i$和$\sum_{x_i>x} w_i$然后判断这两个和中的哪一个大于$1/2$，如果都小于$1/2$就中止算法，否则对包含了weighted median的那个子集递归处理。这种改动没有改变`SELECT`时间复杂度，所以仍为$O(n)$。

湖景房问题

东西向排列的房子，比左右房子高的是湖景房

方法给出：一个栈，初始空，

```
for (int i = 0; i < n; i++) {
    while (!s.empty() && A[i] > A[s.top()])
        s.pop();
    s.push(i);
}
```

正确性证明

平摊分析

	total	*amo*	*acc*
push	2	1	1
pop	0	1	-1

平摊分析正确性

$$\sum \text{accounting} \geq 0$$