

Algorithm - P46

特征序列(streak), 对于掷一枚均匀硬币 n 次

则最长连续正面的序列的期望长度为 $\Theta(\lg n)$, 即 $E(L) \in \Theta(\lg n)$

注意到 $E(L) \in \Theta(\lg n) \leftrightarrow E(L) \in O(\lg n) \wedge E(L) \in \Omega(\lg n)$

则对于 $E(L) \in O(\lg n)$ 的证明

令事件 A_{ik} 为从第 i 次掷开始为连续至少 k 个正面, 其中 $1 \leq i \leq n$, $1 \leq k \leq n-k+1$

由于掷硬币间相互独立, 则有 $\Pr(A_{ik}) = 1/2^k$

且 $k = 2\lceil \lg n \rceil$, 即长度至少为 $2\lceil \lg n \rceil$ 的正面特征序列

$$\Pr(A_{ik}) = \Pr(A_{i, 2\lceil \lg n \rceil}) = 1/2^{2\lceil \lg n \rceil} \leq 1/2^{2\lg n} = 1/n^2$$

由于使 $k = 2\lceil \lg n \rceil$ 成立的 i 值至多有 $n - 2\lceil \lg n \rceil + 1$ 个

$$\begin{aligned} \text{于是 } \Pr(U_{i=1}^{n-2\lceil \lg n \rceil+1} A_{i, 2\lceil \lg n \rceil}) &\leq \sum_{i=1}^{n-2\lceil \lg n \rceil+1} \Pr(A_{i, 2\lceil \lg n \rceil}) \\ &\leq \sum_{i=1}^{n-2\lceil \lg n \rceil+1} (1/n^2) < \sum_{i=1}^n (1/n^2) = 1/n \end{aligned}$$

令事件 L_j 为最长特征序列的长度为 j , 其中 $0 \leq j \leq n$

于是有 $E(L) = \sum_{j=0}^n j \Pr(L_j)$

注意到当 $j < 2\lceil \lg n \rceil$ 时, j 相对较小, 而 $j \geq 2\lceil \lg n \rceil$ 时, $\Pr(L_j)$ 相对较小,

且有对于 $j = 0, 1, \dots, n$, L_j 是互不相交的

于是 $\sum_{j=2\lceil \lg n \rceil}^n \Pr(L_j) = \Pr(U_{i=1}^{n-2\lceil \lg n \rceil+1} A_{i, 2\lceil \lg n \rceil}) < 1/n$

$$\begin{aligned} \text{则 } E(L) &= \sum_{j=0}^n j \Pr(L_j) = \sum_{j=0}^{2\lceil \lg n \rceil-1} j \Pr(L_j) + \sum_{j=2\lceil \lg n \rceil}^n j \Pr(L_j) \\ &< \sum_{j=0}^{2\lceil \lg n \rceil-1} (2\lceil \lg n \rceil) \Pr(L_j) + \sum_{j=2\lceil \lg n \rceil}^n n \Pr(L_j) \\ &= (2\lceil \lg n \rceil) \sum_{j=0}^{2\lceil \lg n \rceil-1} \Pr(L_j) + n \sum_{j=2\lceil \lg n \rceil}^n \Pr(L_j) \\ &< (2\lceil \lg n \rceil) \cdot 1 + n \cdot (1/n) = 2\lceil \lg n \rceil + 1 \end{aligned}$$

于是有 $E(L) \in O(\lg n)$

对于 $E(L) \in \Omega(\lg n)$ 的证明

考虑长度为 $S = \lfloor L(\lg n)/2 \rfloor$ 的特征序列, 将 n 次硬币划分为至多 $\lfloor n/L(\lg n)/2 \rfloor$ 组

如果 \exists 可能存在至少一组全是正面, 则特征序列最长至少为 $S = \lfloor L(\lg n)/2 \rfloor$

注意到 $\Pr(A_{i, \lfloor L(\lg n)/2 \rfloor}) = 1/2^{\lfloor L(\lg n)/2 \rfloor} \leq 1/\sqrt{n}$

于是所有组均不是全正面的概率率为 $(1 - 1/\sqrt{n})^{\lfloor n/L(\lg n)/2 \rfloor}$

则有 $(1 - 1/\sqrt{n})^{\lfloor n/L(\lg n)/2 \rfloor} \leq (1 - 1/\sqrt{n})^{\lfloor n/L(\lg n)/2 \rfloor - 1}$

$\leq (1 - 1/\sqrt{n})^{2n/\lg n - 1} \leq e^{-(2n/\lg n - 1)/\sqrt{n}} \in O(e^{-\lg n}) = O(1/n)$

于是有 $\sum_{j=\lfloor L(\lg n)/2 \rfloor}^n \Pr(L_j) \geq 1 - O(1/n)$

则 $E(L) = \sum_{j=0}^n j \Pr(L_j) = \sum_{j=0}^{\lfloor L(\lg n)/2 \rfloor - 1} j \Pr(L_j) + \sum_{j=\lfloor L(\lg n)/2 \rfloor}^n j \Pr(L_j)$

$> \sum_{j=0}^{\lfloor L(\lg n)/2 \rfloor - 1} 0 \cdot \Pr(L_j) + \sum_{j=\lfloor L(\lg n)/2 \rfloor}^n (\lfloor L(\lg n)/2 \rfloor) \Pr(L_j)$

$= \lfloor L(\lg n)/2 \rfloor \cdot \sum_{j=\lfloor L(\lg n)/2 \rfloor}^n \Pr(L_j) \geq \lfloor L(\lg n)/2 \rfloor (1 - O(1/n))$

于是有 $E(L) \in \Omega(\lg n)$

于是最长特征序列的长度期望 $E(L) \in \Theta(\lg n)$

Algorithm - P47

特征序列中使用指示器随机变量进行简单近似以分析

令 $X_{ik} = I\{A_{ik}\}$, 即开始于第 i 次掷硬币特征序列长度至少为 k 的指示器随机变量

则长度为 k 的特征序列总数为 $X = \sum_{i=1}^{n-k+1} X_{ik}$

$$\text{于是有期望 } E(X) = \sum_{i=1}^{n-k+1} E(X_{ik}) = \sum_{i=1}^{n-k+1} \Pr\{A_{ik}\} = \sum_{i=1}^{n-k+1} \frac{1}{2^k} = \frac{n-k+1}{2^k}$$

即对于不同的 k 值, 如 $E(X)$ 较大, 则可以期望有较多长度为 k 的序列出现

如果 $E(X)$ 较小 (< 1), 则可以期望有很长的长度 k 的序列出现

令 $k = c \lg n$, 其中 c 为一个大于 0 的常数

$$\text{则 } E(X) = (n - c \lg n + 1) / 2^{c \lg n} = (n - c \lg n + 1) / n^c \in \Theta(1/n^{c-1})$$

特别地有, 取 $c = \frac{1}{2}$, 则 $E(X) \in \Theta(1/n^{1/2-1}) = \Theta(n^{1/2})$

于是可以期望有大量的 $(\Theta(n^{1/2}))$ 长度为 $\frac{1}{2} \lg n$ 的特征序列

在线雇用 假设在雇用问题中, 只进行一次雇用, 考虑这次雇用是 n 个人中最好的应聘者的概率

ONLINE-MAXIMUM(k, n) 选择一个正整数 $0 < k < n$

best score := -∞ 对于每个应聘者, 必须立刻雇用或拒绝

for $i := 1$ to k 获得前 k 个应聘者中最高的分数 best score

for if score(i) > best score
best score := score(i) 并拒绝前 k 个应聘者

for $i := k+1$ to n 对于第 $k+1$ 个到第 n 个应聘者

if score(i) > best score 雇用遇到的第一个分数高于 best score 的应聘者

return i [如果这样的应聘者不存在, 则雇用第 n 个应聘者]

令 $M(j) = \max_{1 \leq i \leq j} \{score(i)\}$ 表示前 j 个人中的最高分数

事件 S_i 表示最好的竞争者出现在第 i 个位置, 并且成功雇用

于是有 当 $1 \leq i \leq k$ 时, $\Pr\{S_i\} = 0$

事件 S 表示雇用到最好的竞争者, 则 $\Pr\{S\} = \sum_{i=k+1}^n \Pr\{S_i\}$

可知 S_i 的发生依赖于另外两个独立事件

事件 B_i 表示最好的应聘者出现在位置 i , 则有 $\Pr\{B_i\} = 1/n$

事件 O_i 表示算法没有选择 $k+1 \sim i-1$ 中的应聘者

即对于 $k+1 \leq j \leq i-1$, 有 $score(j) < M(k)$, 即 $\Pr\{O_i\} = \frac{k}{i-1}$

于是有 $\Pr\{S_i\} = \Pr\{B_i \cap O_i\} = \Pr\{B_i\} \Pr\{O_i\} = k/n(i-1)$

则 $\Pr\{S\} = \sum_{i=k+1}^n \Pr\{S_i\} = \sum_{i=k+1}^n k/n(i-1) = \frac{k}{n} \sum_{i=k}^{n-1} \frac{1}{i}$

则有紧约束的上下界 $\int_k^n \frac{1}{x} dx \leq \sum_{i=k}^{n-1} \frac{1}{i} \leq \int_{k-1}^{n-1} \frac{1}{x} dx$

即有 $\frac{k}{n}(\ln n - \ln k) \leq \Pr\{S\} \leq \frac{k}{n} [\ln(n-1) - \ln(k-1)]$

当 $k = n/e$ 时, 下界取得最大值 $1/e$.

Algorithm - P48

排序问题 输入：一个n个数的序列 $\langle a_1, a_2, \dots, a_n \rangle$

通常是一个n元数组，也可以是链表等其他描述方式

输出：输入序列的一个排列（重排，reordering） $\langle a'_1, a'_2, \dots, a'_n \rangle$

使得有 $a'_1 \leq a'_2 \leq \dots \leq a'_n$

记录 (record)，包含待排序的数作为其一部分的数据集

关键字 (key)，每个记录包含一个关键字，即排序问题是需要重排的值

卫星数据 (satellite data)，记录的剩余部分，通常与关键字一同存取

排序算法描述确定有序次序的方法 (method)

无关待排序的是单个的数还是包含卫星数据的记录

于是通常假定输入只由单个的数组成

原地的 (in place)，指在排序过程中仅有常数个元素需要存储在数组之外

期望 (expected)

algorithm

最坏情况 (worst-case) | 平均情况 (average-case)

insertion (插入)

$\Theta(n^2)$

$\Theta(n^2)$

merge (归并)

$\Theta(n \lg n)$

$\Theta(n \lg n)$

heap sort (堆排序)

$\Theta(n \lg n)$

$\Theta(n \lg n)$

quicksort (快速排序)

$\Theta(n^2)$

$\Theta(n \lg n)$ (expected)

counting sort (计数排序)

$\Theta(k+n)$

$\Theta(k+n)$

radix sort (基数排序)

$\Theta(d(n+k))$

$\Theta(d(n+k))$

bucket sort (桶排序)

$\Theta(n^2)$

$\Theta(n)$ (average-case)

二叉堆 (binary heap)，指一个可以看作近似完全二叉树的数组

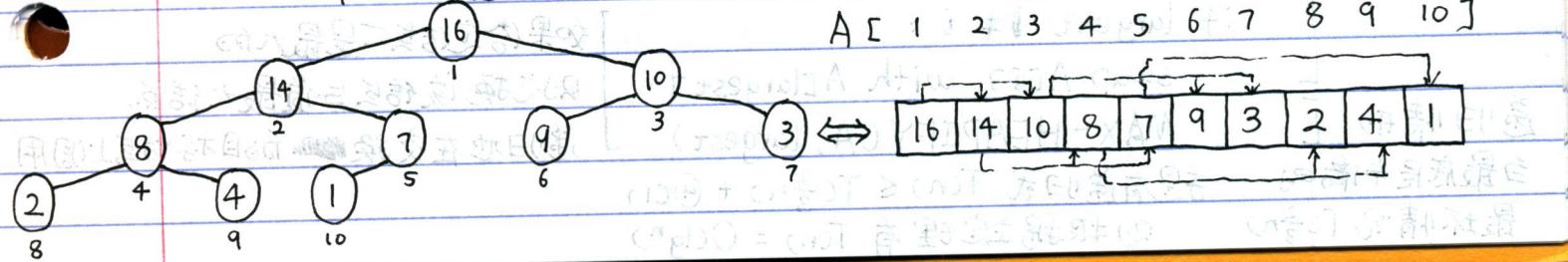
二叉树中的每一个结点，对应数组中的一个元素

除最底层外，二叉树是完全填满的，并且每一层均为从左向右填充

length：表示数组大小的属性。

heap-size：表示存放的堆有效元素的个数，于是有 $0 \leq \text{heap-size} \leq \text{length}$

$A[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]$



Algorithm - P49

对于二叉堆 $A[1..n]$, 对给定的结点下标, 计算父结点, 左子结点, 右子结点,
通常通过宏(macro)或者内联函数(cinline)实现

$\text{PARENT}(i)$ $\text{LEFT}(i)$ $\text{RIGHT}(i)$

$\text{return } i/2$

$\text{return } 2*i$

$\text{return } 2*i + 1$

最大堆 (max-heap), 对于二叉堆 $A[1..n]$, 满足最大堆性质 (max-heap property)

对于结点, $1 < i \leq n$, 即除了根结点, 以外的所有结点,

有结点值不大于父结点的值, 即 $A[\text{PARENT}(i)] \geq A[i]$

于是有, 在最大堆中, 根结点中存放着最大元素

最小堆 (min-heap), 对于二叉堆 $A[1..n]$, 满足最小堆性质 (min-heap property)

对于除根结点, 以外的所有结点, 即结点, $1 < i \leq n$,

有结点值不小于父结点的值, 即 $A[\text{PARENT}(i)] \leq A[i]$

于是有, 在最小堆中, 根结点中存放着最小元素

高度 (height) 在 $A[1..n]$ 二叉堆中, 结点的高度为其到叶结点, 最长简单路径的边数

注意在二叉堆中, 根结点的高度最大, 叶结点, 高度为 0

虽然描述为最长简单路径, 但是默认是只经过以其为根的子树结点,

定义堆高度为根结点的高度, 则有高度为 $O(\lg n)$

堆操作的运行时间与高度成正比, 即时间复杂度为 $O(\lg n)$

堆维护 $\text{MAX-HEAPIFY}(A, i)$

$\begin{cases} l = \text{LEFT}(i) \\ r = \text{RIGHT}(i) \end{cases}$] 对于给定的结点, 考虑结点, 可能不满足最大堆性质
即可能存在其某个子结点大于该结点。

$\text{largest} = i$] 维护使得满足最大堆性质

分解与合并 复杂度 $\Theta(1)$] 找到该结点与其子结点中

$\text{largest} = l$] 值最大的结点下标

$\text{if } r \leq A.\text{heap-size} \text{ and } A[r] > A[\text{largest}]:$

$\text{largest} = r$

$\text{if largest } \neq i:$

$\text{swap } A[i] \text{ with } A[\text{largest}]$

] 如果该结点不是最大的

则交换该结点与值最大结点,

$\text{MAX-HEAPIFY}(A, \text{largest})$] 递归地在交换后的目标结点上调用

当最底层半满时

于是有递归式 $T(n) \leq T(\frac{2}{3}n) + \Theta(1)$

最坏情况 $T(\frac{2}{3}n)$

则根据主定理有 $T(n) = O(\lg n)$

Algorithm - P50

堆维护

对于某些编译器，MAX-HEAPIFY中的递归调用可能产生低效代码

<MAX-HEAPIFY-LOOP(A, i):

 while $i \leq A.\text{heap-size}/2$:

$l = \text{LEFT}(i)$

$r = \text{RIGHT}(i)$

 largest = i

 if $l \leq A.\text{heap-size}$ and $A[l] > A[\text{largest}]$:

 largest = l

 if $r \leq A.\text{heap-size}$ and $A[r] > A[\text{largest}]$:

 largest = r

 if largest $\neq i$:

 swap $A[i]$ with $A[\text{largest}]$

 else:

 break

对于一个大小为 $n \in \mathbb{N}^+$ 的堆，MAX-HEAPIFY 的最坏情况复杂度为 $\Omega(\lg n)$

证明过程有，对于一个大小为 n 的堆，考虑其中最小元素 a_0 的位置

当 a_0 的位置为根结点时，则在 MAX-HEAPIFY 的每次调用中

largest 的值都会从 i 的左右孩子中选择其一

直到 $i > A.\text{heap-size}/2$ ，即到达叶结点

由于 MAX-HEAPIFY 每次调用 a_0 会向下移动一层

则在 a_0 从根结点移动到叶结点的过程中，

需要 $\lfloor \lg n \rfloor$ 或 $\lfloor \lg n \rfloor - 1$ 次 MAX-HEAPIFY 调用

又 MAX-HEAPIFY 的其他步骤复杂度为 $\Theta(1)$

于是 MAX-HEAPIFY 是 $\Omega(\lg n)$ 的

即有 MAX-HEAPIFY 是 $\Theta(\lg n)$ 的

建堆

利用 MAX-HEAPIFY 将大小为 $n = A.\text{length}$ 的数组 $A[1..n]$ 转换为最大堆

BUILD-MAX-HEAP(A):

$A.\text{heap-size} = A.\text{length}$

 for $i = \lfloor A.\text{length}/2 \rfloor$ down to 1:] $A[\lfloor n/2 \rfloor + 1], \dots, A[n]$ 均为叶结点

 MAX-HEAPIFY(A, i)] 叶结点，可看作只包含一个元素的堆

Algorithm - P51

五、堆

建堆

循环不变量：在 $\text{for } i := \lfloor A.length / 2 \rfloor \text{ downto } 1$ 的循环迭代开始前，
结点 $i+1, i+2, \dots, n$ 都是最大堆的根结点。

初始化：在第一次迭代开始前，有 $i = \lfloor n/2 \rfloor$ 。

由于堆是近似完全的二叉树对，即除最下一层全满，最下一层向左填满，则有 $\lfloor n/2 \rfloor + 1, \lfloor n/2 \rfloor + 2, \dots, n$ 都是叶结点。

由于只有根结点的二叉树平凡地是最大堆。

于是结点 $i+1, i+2, \dots, n$ 都是最大堆根结点，平凡地为真。

保持：假设在某一次迭代前， $k+1, k+2, \dots, n$ 都是最大堆的根结点，

可知对于结点 k ，以其为根结点的子树结点，均在 $k+1, k+2, \dots, n$ 中。
如果结点 $k, \text{LEFT}(k), \text{RIGHT}(k)$ 中，最大值为结点 k ，

则对于 $\text{LEFT}(k)$ 和 $\text{RIGHT}(k)$ ，有

$$A[\text{PARENT}(\text{LEFT}(k))] = A[k] \geq A[\text{LEFT}(k)]$$

$$A[\text{PARENT}(\text{RIGHT}(k))] = A[k] \geq A[\text{RIGHT}(k)]$$

于是 $\text{MAX-HEAPIFY}(A, k)$ 终止。

结点 k 为最大堆的根结点。

如果 $\text{LEFT}(k)$ 或 $\text{RIGHT}(k)$ 结点值大于结点 k ，

不失一般性地认为 $\text{LEFT}(k)$ 结点的值最大。

则交换 $\text{LEFT}(k)$ 与 k 结点的值， $\text{RIGHT}(k)$ 结点不变。

继续调用 $\text{MAX-HEAPIFY}(A, \text{LEFT}(k))$ 。

可知这个过程当某一个结点 k' 或者是叶结点，

或者结点 $k', \text{LEFT}(k'), \text{RIGHT}(k')$ 中 $A[k']$ 最大时终止。

于是结点 k 为最大堆的根结点。

则在下一次迭代开始前， $k, k+1, k+2, \dots, n$ 都是最大堆的根结点。

终止：循环终止时， $i=0$ ，则有 $1, 2, \dots, n$ 都是最大堆的根结点。

又 $A[1]$ 是堆 A 的根结点，则堆 A 为最大堆。

特别注意：循环迭代的顺序是 $\text{for } i := \lfloor A.length / 2 \rfloor \text{ downto } 1$

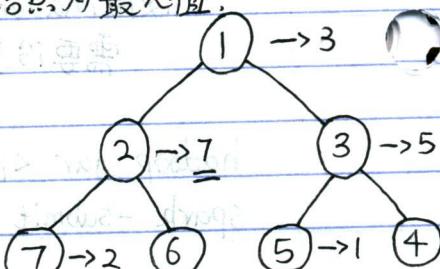
而不能是 $\text{for } i := 1 \text{ to } \lfloor A.length / 2 \rfloor$ 。

由于在根结点迭代结束后，无法保证根结点为最大值。

而之后的迭代不会改变根结点的值。

于是使得某一个非根结点包含堆的最大值。

则该结点不满足最大堆性质。



Algorithm - P52

堆 对于任意包含 $n \in \mathbb{Z}^+$ 个元素的堆中，至多有 $\lceil n/2^{h+1} \rceil$ 个高度为 h 的结点，其中 $h \in \mathbb{N}$
证明过程有，对于包含 n 个元素的堆，且由于堆是近似完全的二叉树
所以有堆的高度 $H = \lfloor \log_2 n \rfloor$

相对地 对于高度为 $H \in \mathbb{N}$ 的堆

其包含的元素个数范围是 $2^H \leq n \leq 2^{H+1} - 1$

基础步骤：对于 $h=0$ ，即堆中的叶结点，

由于堆是近似完全的二叉树，叶结点的深度为 H 或 $H-1$
且包括所有深度为 H 的结点，

且没有子结点，的深度 $H-1$ 的结点，

令 x 为深度为 H 的结点，的数量，即在最下一层的结点，

注意到 $n-x$ 的部分是完全二叉树，于是 $n-x$ 是奇数

即当 n 是奇数时 x 为偶数， n 是偶数时 x 为奇数

当 n 为奇数时， x 为偶数，于是所有内点均有两个子结点，

于是有内点数 $i = \text{叶结点数 } l - 1$

即有 $n = i + l = 2l - 1$ ，即有 $l = \frac{n+1}{2} = \lceil n/2 \rceil \leq \lceil n/2^{h+1} \rceil$

当 n 为偶数时， x 为奇数，于是除了一个内点，其余内点均有两个子结点，

则可以在该内点上添加一个子叶点，则此结点是叶结点，

于是有 $l+1 = \frac{(n+1)+1}{2} = \lceil n/2 \rceil + 1$ ，即有 $l = \lceil n/2 \rceil \leq \lceil n/2^{h+1} \rceil$

递归步骤：假设对于高度 $0 \leq h < H$, $P(h)$ 为真，则考虑 $P(h+1)$

令子树 T 为堆中高度不低于 h 的结点，

子树 T' 为堆中高度不低于 $h+1$ 的结点，

则堆中高度为 h 的结点个数为 T 的叶结点，

于是有 $\lceil n_{(T)} / 2^h \rceil = \lceil n_{(T)} \rceil - \lceil n_{(T')} \rceil$

$$\lceil n_{(T)} / 2^h \rceil = n_{(T)} - \lceil n_{(T')} / 2^h \rceil \stackrel{(IH)}{\leq} \lceil n_{(T)} / 2^{h+1} \rceil$$

$$n_{(T')} = n_{(T)} - \lceil n_{(T)} / 2^h \rceil = \lceil n_{(T)} / 2^h \rceil$$

则高度为 $h+1$ 的结点个数为 T' 的叶结点，

$$n_{(T')} = \lceil n_{(T')} / 2^{h+1} \rceil = \lceil n_{(T)} / 2^{h+1} \rceil \leq \lceil n_{(T)} / 2^h \rceil$$

$$\lceil n_{(T)} / 2^h \rceil \leq \lceil n_{(T)} / 2^{h+1} \rceil \leq \lceil n_{(T)} / 2^h \rceil$$

根据数学归纳法，对于任意包含 $n \in \mathbb{Z}^+$ 个元素的堆中

至多有 $\lceil n/2^{h+1} \rceil$ 个高度为 h 的结点