QuickSort

The Sorting Problem

> **定义 3.3** 排序问题
>
> - 输入：一组各不相同的有序的元素 $< a_1, a_2, \ldots, a_n >$
> - 输出：输出元素的某个排列 $< a'_1, a'_2, \ldots, a'_n >$，满足 $a'_1 < a'_2 < \cdots < a'_n$

Comparison-Based Sorting

不特殊说明时，主要考虑“基于比较的排序”，即模型提供了比较元素大小关系的操作，且只能用该比较操作决定元素间的序。

critical operation：元素间的比较

Insertion Sort

Insertion-Sort (A[1...n]):

```
for j := 2 to n do
    temp := A[j];
    i := j - 1;
    while i > 0 and A[i] > temp do
        A[i+1] := A[i];
        i := i - 1;
    A[i+1] := temp;
```

最坏情况时间复杂度：当待插入元素前有 i 个已排好序的元素时，最多情况下需比较 i 次，则

$$W(n) \leqslant \sum_{i=1}^{n-1} i = \frac{n(n-1)}{2} = \Theta(n^2)$$

平均情况时间复杂度：当待插入元素前有 i 个已排好序的元素时

- 假设输入的各种序列等可能出现
- 假设没有两个 key 相同的元素，

待插入元素有 $i+1$ 个被插入的位置，且插入在各个位置的概率是相等的，此时插入的比较次数期望值为

$$
\begin{aligned}
c_{i+1} &= \frac{1}{i+1}\sum_{j=1}^{i} j + \frac{1}{i+1} i \\
&= \frac{i}{2} + 1 - \frac{1}{i+1}
\end{aligned}
$$

则平均情况时间复杂度为

$$
\begin{aligned}
A(n) &= \sum_{i=1}^{n-1} c_{i+1} \\
&= \frac{n(n-1)}{4} + n - 1 - \sum_{j=2}^{n} \frac{1}{j} \\
&= \frac{n^2}{4} + \frac{3n}{4} - (\ln \gamma + \varepsilon(n) - 1) \\
&= \Theta(n^2)
\end{aligned}
$$

Inversion and Sorting

> 逆序（inversion）：在一个未排序的序列 $E:\{x_1, x_2, x_3 \ldots, x_n\}$ 中
>
> $$< x_i, x_j > \text{ is an inversion } \iff x_i > x_j, i < j$$

排序问题可等价为消除逆序对

最坏情况时间复杂度：在最差情况下，n 个元素的输入最多可包含 $\frac{n(n-1)}{2}$ 个 inversion

故一次比较最多消除一个 inversion 的排序算法的最坏情况时间复杂度一定为 $\Omega(n^2)$，e.g. 选择排序，插入排序，冒泡排序

平均情况时间复杂度：考虑一个输入序列与其逆序

$$
\begin{gathered}
x_1, x_2, x_3, \ldots, x_n \\
x_n, x_{n-1}, \ldots, x_2, x_1
\end{gathered}
$$

一个 inversion $< x_i, x_j >$ 一定出现在其中一个序列之中，而 inversion 总和为 $\frac{n(n-1)}{2}$，故输入序列中 inversion 的个数期望为 $\frac{n(n-1)}{4}$

一次比较最多消除一个 inversion 的排序算法的平均情况时间复杂度一定为 $\Omega(n^2)$，e.g. 选择排序，插入排序，冒泡排序

QuickSort

Strategy

根据一个选定的 pivot 将数组分成两部分，元素都小于 pivot 的部分和元素都不小于 pivot 的部分，然后递归地对两个子数组排序

Divide and Conquer (hard divide, easy combination):

- Divide: Partition the array into two parts "small" and "large"
- Conquer: Sort "small" and "large" recursively
- Combine: Easy to combine sorted arrays

Algorithm

PARTITION (A, p, r):

```
pivot := A[r];
i := p - 1;
for j := p to r - 1 do
    if A[j] < pivot then
        i := i + 1;
        SWAP(A[i], A[j]);
SWAP(A[i + 1], A[r]);
return i + 1;
```

QUICK-SORT (A, p, r):

```
if p < r then
    q := PARTITION(A, p, r);
    QUCIK-SORT(A, p, q-1);
    QUICK-SORT(A, q+1, r);
```

Analysis

Worst-case

Quick Sort 的性能主要受 Partition 影响，设有 k 个元素，则 Partition 的比较次数为 $k-1$，即 Partition 代价为 $\Theta(n)$

最坏情况即为划分后的其中一个子数组为空，问题规模每次减少 1

$$
\begin{aligned}
T(n) &= T(n-1) + T(0) + \Theta(n) \\
&= T(n-1) + \Theta(n)
\end{aligned}
$$

显然 $T(n) = \Theta(n^2)$

Average-case （基于递归方程分析）

首先假设所有可能的输入等可能出现

Partition 将 n 个元素分为三部分（比较 $n-1$ 次），左边部分有 i 个元素（$0 \leqslant i \leqslant n-1$），右边部分有 $n-i-1$ 个元素，$i \in \{0, 1, 2, \ldots, n-1\}$，选取每个值的概率相等，即 $\frac{1}{n}$

平均比较次数为

$$
A(n) = (n-1) + \sum_{i=0}^{n-1} \frac{1}{n}[A(i) + A(n-1-i)] \text{ for } n \geqslant 2
$$

且 $A(1) = A(0) = 0$，又有

$$
\sum_{i=0}^{n-1} A(i) = \sum_{i=0}^{n-1} A[(n-1) - i]
$$

故

$$
A(n) = (n-1) + \frac{2}{n} \sum_{i=1}^{n-1} A(i) \text{ for } n \geqslant 1
$$

> Smart Guess:
>
> $$
> Q(n) \approx 2Q(\frac{n}{2}) + \Theta(n)
> $$
>
> 根据 Master Theorem
>
> $$
> Q(n) = \Theta(n \log n)
> $$

或者是归纳证明该递归方程

> Inductive Proof: $A(n) = O(n \ln n)$
>
> $A(n) \leqslant cn \ln n$ for some constant c
>
> - Base case: n = 1 is trivial
> - Inductive Hypothesis: $A(i) \leqslant ci \ln i$ for $1 \leqslant i < n$
> - Inductive step:

> $$A(n) = (n-1) + \frac{2}{n}\sum_{i=1}^{n-1} A(i) \leqslant (n-1) + \frac{2}{n}\sum_{i=1}^{n-1} ci\ln i$$
>
> $$\text{Note: } \frac{2}{n}\sum_{i=1}^{n-1} ci\ln i \leqslant \frac{2c}{n}\int_1^n x\ln x\mathrm{d}x \approx \frac{2c}{n}\left(\frac{n^2\ln n}{2} - \frac{n^2}{4}\right) = cn\ln n - \frac{cn}{2}$$
>
> $$\text{So, } A(n) \leqslant cn\ln n + n(1-\frac{c}{2}) - 1$$
>
> $$\text{Let } c=2,\ \text{we have } A(n) \leqslant 2n\ln n$$
>
> Inductive Proof: $A(n) = \Omega(n\ln n)$
>
> $A(n) > cn\ln n$ for some constant c , with large n
>
> - Inductive reasoning:
>
> $$A(n) = (n-1) + \frac{2}{n}\sum_{i=1}^{n-1} A(i) > (n-1) + \frac{2}{n}\sum_{i=1}^{n-1} ci\ln i\ (\text{Hypothesis})$$
>
> $$= (n-1) + \frac{2c}{n}\sum_{i=2}^{n} i\ln i - 2c\ln n \geqslant (n-1) + \frac{2c}{n}\int_1^n x\ln x\mathrm{d}x - 2c\ln n$$
>
> $$\approx cn\ln n + \left[(n-1) - c\left(\frac{n}{2} + 2\ln n\right)\right]$$
>
> $$\text{Let } c < \frac{n-1}{\frac{n}{2}+2\ln n},\ \text{then } A(n) > cn\ln n \quad (\text{Note: } \lim_{n\to\infty}\frac{n-1}{\frac{n}{2}+2\ln n} = 2)$$

直接解递归方程:

$$A(n) = (n-1) + \frac{2}{n}\sum_{i=1}^{n-1} A(i)$$

$$A(n-1) = (n-2) + \frac{2}{n-1}\sum_{i=1}^{n-2} A(i)$$

故

$$\begin{aligned} nA(n) - (n-1)A(n-1) &= n(n-1) + 2\sum_{i=1}^{n-1} A(i) - (n-1)(n-2) - 2\sum_{i=1}^{n-2} A(i) \\ &= 2A(n-1) + 2(n-1) \\ nA(n) &= (n+1)A(n-1) + 2(n-1) \\ \frac{A(n)}{n+1} &= \frac{A(n-1)}{n} + \frac{2(n-1)}{n(n+1)} \end{aligned}$$

令 $B(n) = \dfrac{A(n)}{n+1}$

$$
\begin{aligned}
B(n) &= \sum_{i=1}^{n} \frac{2(i-1)}{i(i-1)} \\
&= 4\sum_{i=1}^{n} \frac{1}{i+1} - 2\sum_{i=1}^{n} \frac{1}{i} \\
&= 4\sum_{i=2}^{n+1} \frac{1}{i} - 2\sum_{i=1}^{n} \frac{1}{i} \\
&= 2\sum_{i=1}^{n} \frac{1}{i} - \frac{4n}{n+1} \\
&= O(\log n)
\end{aligned}
$$

故 $A(n) = O(n\log n)$

基于指标随机变量的分析将在 Tutorial 2 中详细讲解

Space Complexity

Partition 是 in-place 的，空间复杂度为 $O(1)$

但在最坏情况递归深度为 $n-1$，递归栈大小为 $\Theta(n)$

More than Sorting

Partition

Bolts and Nuts

k-sorted