算法基础 2022 春　　　　　　　　　　　　　　　　　　　　　　Homework 2

任课教师：陈雪　　　　　　　　　　　　　　　　　　　　　　due: Mar 10, 15:30

（1）只准讨论思路，严禁抄袭

（2）只能阅读 bb 上的材料和教材算法导论。严禁网上搜寻任何材料，答案或者帮助

问题 1 (30 分). 分析程序，证明下面的问题：

Algorithm 1 Quick Sort algorithm

```
1:  function QSort($l, r$)
2:      if $l \geq r$ then
3:          exit
4:      pick a random element in $A[l], ..., A[r]$ as $pivot$
5:      $i = l, j = r$
6:      while $i \leq j$ do
7:          while $A[i] < pivot$ do $i = i + 1$
8:          while $A[j] > pivot$ do $j = j - 1$
9:          if $i \leq j$ then
10:             Swap($A[i], A[j]$)
11:             $i = i + 1$
12:             $j = j - 1$
13:     QSort($l, j$)
14:     QSort($i, r$)
```

(a)（10 分）每次调用 QSort(l, r) 时，if 块内的语句至少执行一次。

(b)（10 分）外层循环结束后，恒有 $A[l], ..., A[i-1] \leq pivot, A[j+1], ..., A[r] \geq pivot$ 成立。

(c)（10 分）外层循环结束时，$i \neq l$ 且 $j \neq r$。故程序可以终止。

Answer:

(a) The first case $l < r$: Because $pivot$ is in $A[l], \ldots, A[r]$, i and j will quit their while loops (separately) at least once. Moreover, i and j satisfy the **if** condition in line 9, $i \leq j$, when they quit the while at the 1st time.

The second case $l \geq r$: l and r satisfy the **if** condition in line 2.

Therefore, conditional statements would be true at least once each time QSORT(l, r) is called.

(b) Let A_0 be the input of the function and A be the output. Assuming that $A[k]$ is the first element which is greater than *pivot* between $A[l]$ with $A[i-1]$ after the big while loop, and $A[k]$ was $A_0[u]$ when the function started.

If $u = k$, The statement, '**While** $A[i] < pivot$ $i = i+1$' would swap it with $A_0[k]$ as $k < i$ and $A[k] > pivot$. So $u \neq k$, which indicates $A_0[k]$ was swapped with $A_0[u]$ in the big loop. But it contradicts the statement, '**While** $A[j] > pivot$ $j = j+1$', which means $A[k]$ wouldn't be swapped because of $A[k] > pivot$.

Therefore, none of $A[l], ..., A[i-1]$ greater than *pivot*.

And the other side is symmetric.

(c) proposition (b) tells us that **if** is executed at least once, Then $i = i+1 > l$ and $j = j-1 < r$. Therefore, the input length strictly decreases with increasing depth. At last $l \geq r$ and function exits.

问题 2 (40 分)**.** 以下是快速排序的找第 k 大元素的变形，主过程会直接调用 RANDOMSELECT$(A, 1, n, k)$。修改课上使用比较次数期望的分析方法，证明此程序的期望时间复杂度是 $O(n)$。

提示：程序的功能分析可参考算法导论第 9 章关于中位数的内容。为了得到本题所有的分数，只能使用基于比较次数概率与期望的方法。

Answer:

As analysis in the course, The time complexity of algorithms like QuickSort mainly consist of its frequency of comparison.

Pairwise comparison only happens when one of them is chosen as pivot. And current pivot wouldn't compare with previous pivot. Therefore, comparison between two element occurs at most once.

Let Let $z_1, ..., z_n$ be sorted order of A and $X_{i,j} \in 0, 1$ denotes whether it compares z_i and z_j or not. Elements between z_i and z_j shouldn't be picked as pivot before, otherwise z_i and z_j

would be separated. What's more, Elements between z_i and z_k also shouldn't be picked as pivot before, otherwise z_i would be abandoned. Similarly, Elements between z_j and z_k also shouldn't be picked as pivot before. In general, $X_{i,j} = 1$ if and only if z_i or z_j is chosen as pivot fisrt.

without loss of generality, Let $k \leq \frac{n+1}{2}$.

$$\begin{aligned}
\mathbb{E}[\sum_{i<j} X_{i,j}] &= \sum_{i<j} \mathbb{E}[X_{i,j}] \\
&= \sum_{k\leq i<j} \mathbb{E}[X_{i,j}] + \sum_{i<k<j} \mathbb{E}[X_{i,j}] + \sum_{i<j\leq k} \mathbb{E}[X_{i,j}] \\
&= \sum_{k\leq i<j} \frac{2}{j-k+1} + \sum_{i<k<j} \frac{2}{j-i+1} + \sum_{i<j\leq k} \frac{2}{k-i+1} \\
&= \sum_{t=1}^{n-k} t \cdot \frac{2}{t+1} + \sum_{t=2}^{k-1} (t-1) \cdot \frac{2}{t+1} + \sum_{t=k}^{n-k+1} (k-1) \cdot \frac{2}{t+1} + \sum_{t=n-k+2}^{n-1} (n-t+1) \cdot \frac{2}{t+1} + \sum_{t=1}^{k-1} t \cdot \frac{2}{t+} \\
&= O(n)
\end{aligned}$$

So algorithm's time complexity is $O(n)$.

问题 3 (10 分)**.** 参考算法导论第 8.1 章，考虑 n 个未知元素所有的 $n!$ 个排列。

(a)（10 分）对任何的基于比较排序的算法 A，证明对至少 99% 的全排列，A 的运行时间为 $\Omega(n \log n)$。也就是

$$\Pr_{\sigma}\left[Time\Big(A(\sigma)\Big) \geq 0.5 n \log n\right] \geq 0.99.$$

(b)（10 分）**附加题:** 考虑任何基于比较排序的随机化算法 A，证明存在某个输入排列 σ，使得 A 的期望时间为 $\Omega(n \log n)$。也就是

$$\mathop{\mathbb{E}}_{r}\left[Time\big(A(\sigma, r)\big)\right] \geq 0.5 n \log n.$$

提示： 可以假设算法 A 最多使用 n^2 个随机比特。于是 r 的取值范围为 $\{0,1\}^{n^2}$，同时注意 $A(\cdot, r_0)$ 为确定性的排序算法在固定的随机串 $r_0 \in \{0,1\}^{n^2}$ 后。

(c)（10 分）**附加题:** 加强 (b) 的结果成

$$\Pr_{\sigma}\left[\mathop{\mathbb{E}}_{r}\left[Time\big(A(\sigma, r)\big)\right] \geq 0.5 n \log n\right] \geq 0.99.$$

Answer:

(a) To prepare for the proof later, we can prove a stronger proposition:

$$\forall \epsilon > 0, \delta > 0, \exists n_0, \Pr_{\sigma}\left[Time\Big(A(\sigma)\Big) \geq (1-\epsilon)n\log n\right] \geq 1-\delta \ for\ n > n_0.$$

Assuming the inequality isn't true. It infers $2^{\epsilon n\log n} \geq \delta \cdot n! \approx \delta \cdot \sqrt{2\pi n} \cdot (\frac{n}{e})^n$, which is wrong when $n > 4^{1/\epsilon} + \frac{\log\delta}{\epsilon}$. Therefore, the proposition is true. When $\epsilon = 0.5$, $\delta = 0.01$, this proposition becomes the proposition in 3(a).

(b)

$$\begin{aligned}\mathop{\mathbb{E}}_{\sigma}\mathop{\mathbb{E}}_{r} Time\Big(A(\sigma,r)\Big) &= \mathop{\mathbb{E}}_{r}\mathop{\mathbb{E}}_{\sigma} Time\Big(A(\sigma,r)\Big)\\ &\geq \mathop{\mathbb{E}}_{r}(1-\epsilon)(1-\delta)n\log n\\ &= (1-\epsilon)(1-\delta)n\log n \qquad (1)\end{aligned}$$

With $(1-\epsilon)(1-\delta) \geq 0.5$, we get $\mathbb{E}_{\sigma}\mathbb{E}_{r}\, Time\Big(A(\sigma,r)\Big) \geq 0.5n\log n$. So there exists a permutation σ, $\mathbb{E}_{r}\, Time\Big(A(\sigma,r)\Big) \geq 0.5n\log n$.

(c) Assuming $\Pr_{\sigma}\left[\mathbb{E}_r\left[Time\big(A(\sigma,r)\big)\right] < 0.5n\log n\right] \geq 0.01$.

And $\mathop{\mathbb{E}}_{r}\left[Time\big(A(\sigma,r)\big)\right] < 0.5n\log n$ infers $\Pr_{r}\left[Time\big(A(\sigma,r)\big)\right] < 0.9n\log n] \geq 1 - \frac{0.5}{0.9}$.

Therefore, $\Pr_{\sigma,r}\left[Time\big(A(\sigma,r)\big)\right] < 0.9n\log n] \geq \frac{0.01\cdot 0.4}{0.9}$.

However, (a) tells us that $\Pr_{\sigma,r}\left[Time\big(A(\sigma,r)\big)\right] \geq 0.9n\log n] > 1 - \frac{0.01\cdot 0.4}{0.9}$. It contradicts the above results.

Consequently,

$$\Pr_{\sigma}\left[\mathop{\mathbb{E}}_{r}\left[Time\big(A(\sigma,r)\big)\right] \geq 0.5n\log n\right] \geq 0.99.$$

问题 4 (20 分)**.** 考虑插入排序的过程中，$A[1] \le A[2] \le \ldots \le A[j]$ 已经排好顺序，此时插入 $A[j+1]$。

(a)（10 分）设计一个算法（提供伪代码）在 $O(\log n)$ 的时间按内找出 $A[j+1]$ 的正确位置。

(b)（10 分）证明时间复杂度和算法的正确性。

Answer:

Algorithm 2 Binary Search algorithm

```
function BinarySearch($(A, l, r, t)$)
    while $l < r - 1$ do
        if $t \le A[l]$ then
            return $l$
        else if $t \ge A[r]$ then
            return $r + 1$
        $m = \lceil (l + r)/2 \rceil$;
        if $t < A[m]$ then
            $r = m$
        else if $t > A[m]$ then
            $l = m$
        else
            return $m$
    return $r$
```

To get the location of $A[j+1]$, the input of algorithm is BinarySearch$(A, 1, j, A[j+1])$.

As $l - r$ halves each loop, **while** will be executed $O(\log(l - r))$ times at most. And each loop costs O(1) time. Therefore, time complexity of the algorithm is $O(\log(l - r)) = O(\log j)$.

line 3-6 of code deal with boundary conditions. If $A[j+1] \le A[l]$ or $A[j+1] \ge A[r]$, we can find out location directly. Otherwise, $A[l] < A[j+1] < A[r]$. Then line 8-13 compares $A[j+1]$ with $A[m]$ to reduce the scope. As $A[1], ..., A[j]$ have been sorted, $A[l] < A[j+1] < A[m]$ or $A[m] < A[j+1] < A[r]$ if loop continues, which infers the position is between them. At last, loop ends when $r - l \le 1$. So $A[l] \le A[j+1] \le A[r]$.