算法基础HW2

PB19071535徐昊天

问题1

答：

(a)证明如下：

①$l \geq r$时

将执行第2、3行的if语句如下：

```
1 if l≥r then
2    exit
```

故满足**if语句至少执行一次**的条件。

②$l < r$时

用反证法，假设在外层循环进行中，不执行一次**if语句**，即第一次循环经过第7、8行的代码后满足$i > j$。

令执行第9行时$i = i_1, j = j_1$，根据如下第7、8行代码：

```
1 while A[i] < pivot do i = i + 1
2 while A[j] > pivot do j = j - 1
```

可得已知条件：

$$A[l]、A[l+1]、\dots A[i_1-1] < pivot \quad \dots(1)$$
$$A[j_1+1]、A[j_1+2]、\dots A[r] > pivot \quad \dots(2)$$
$$A[i_1] \geq pivot \quad \dots(3)$$
$$A[j_1] \leq pivot \quad \dots(4)$$
$$i_1 > j_1 \quad \dots(5)$$

由以上式子可推断：

$$\text{由}(5)\text{式可知：}\ l \leq j_1 \leq i_1 - 1$$
$$\text{根据}(1)\text{式可知，}j\text{循环到}i_1-1\text{便会终止}$$
$$\text{故必有}j_1 = i_1 - 1$$
$$\text{同理，由}(5)\text{式可知：}\ j_1 + 1 \leq i_1 \leq r$$
$$\text{根据}(2)\text{式可知，}i\text{循环到}j_1+1\text{便会终止}$$
$$\text{故必有}i_1 = j_1 + 1$$
$$\text{根据}(1)(2)\text{式可知：}\ A[j_1] = A[i_1-1] < pivot, A[i_1] = A[j_1+1] > pivot$$
$$\text{故对}x \in [l,r]\text{：}\ x \in [l, j_1]\text{时，}\ A[x] < pivot; x \in [j_1+1, r]\text{时，}\ A[x] > pivot$$

然而根据实际情况，A数组中必须存在等于 pivot 的元素，这与以上推论得到的结果相矛盾，故假设不成立，外层循环执行中至少执行一次**if语句**。

综上，原命题得证，每次调用 QSort(l, r) 时，**if语句**至少执行一次。

(b)根据外循环代码可知，i的初始值为l，j的初始值为r，假设循环第n次后i、j的值分别为i_n、j_n，外层循环结束后i、j的值分别为i'、j'。

用数学归纳法证明：

> 外层循环进行n次后，恒有 $A[l],\dots,A[i_n-1]\le pivot,A[j_n+1],\dots,A[r]\ge pivot$成立。

①第一次循环后

由(a)问可知，**if语句**在第一次循环中一定执行，即在7、8行代码执行结束后，$i=i_1-1,j=j_1+1$，且满足如下条件：

$$\begin{aligned}A[l]、A[l+1]、\dots A[i_1-2]&<pivot\\A[j_1+2]、A[j_1+3]、\dots A[r]&>pivot\\A[i_1-1]&\ge pivot\\A[j_1+1]&\le pivot\end{aligned}$$

执行**if语句**后，交换$A[i_1-1]$与$A[j_1+1]$，条件变为如下所示：

$$\begin{aligned}A[l]、A[l+1]、\dots A[i_1-2]&<pivot\\A[i_1-1]&\le pivot\\A[j_1+2]、A[j_1+3]、\dots A[r]&>pivot\\A[j_1+1]&\ge pivot\end{aligned}$$

故有$A[l],\dots,A[i_1-1]\le pivot,A[j_1+1],\dots,A[r]\ge pivot$成立。

②证明第n次循环满足以上命题，则第n+1次循环也满足以上命题。

显然第n次循环后有:$A[l],\dots,A[i_n-1]\le pivot,A[j_n+1],\dots,A[r]\ge pivot$成立。

要证第n+1次循环后有:

$A[l],\dots,A[i_{n+1}-1]\le pivot,A[j_{n+1}+1],\dots,A[r]\ge pivot$成立。

证明过程如下：

显然第n+1次循环开始时i、j的值分别为i_n、j_n。

分以下两种情况讨论：

1. **if语句**执行

   在7、8行代码执行结束后，$i=i_{n+1}-1,j=j_{n+1}+1$，且满足如下条件：

   $$\begin{aligned}A[i_n]、A[i_n+1]、\dots A[i_{n+1}-2]&<pivot\\A[j_{n+1}+2]、A[j_{n+1}+3]、\dots A[j_n]&>pivot\\A[i_{n+1}-1]&\ge pivot\\A[j_{n+1}+1]&\le pivot\end{aligned}$$

   执行**if语句**后，交换$A[i_{n+1}-1]$与$A[j_{n+1}+1]$，条件变为如下所示：

   $$\begin{aligned}A[i_n]、A[i_n+1]、\dots A[i_{n+1}-2]&<pivot\\A[i_{n+1}-1]&\le pivot\\A[j_{n+1}+2]、A[j_{n+1}+3]、\dots A[j_n]&>pivot\\A[j_{n+1}+1]&\ge pivot\end{aligned}$$

   故有$A[i_n],\dots,A[i_{n+1}-1]\le pivot,A[j_{n+1}+1],\dots,A[j_n]\ge pivot$成立。

2. **if语句**不执行

   在7、8行代码执行结束后，$i=i_{n+1},j=j_{n+1}$，循环结束且满足如下条件：

$$A[i_n]、A[i_n+1]、\dots A[i_{n+1}-1]<pivot$$
$$A[i_{n+1}]\le pivot$$
$$A[j_{n+1}+1]、A[j_{n+1}+2]、\dots A[j_n]>pivot$$
$$A[j_{n+1}]\ge pivot$$

故有

$A[i_n],\dots,A[i_{n+1}-1]\le pivot, A[j_{n+1}+1],\dots,A[j_n]\ge pivot$成立。

综上可知，第n+1次循环后必有：$A[i_n],\dots,A[i_{n+1}-1]\le pivot, A[j_{n+1}+1],\dots,A[j_n]\ge pivot$成立。

由于已知：$A[l],\dots,A[i_n-1]\le pivot, A[j_n+1],\dots,A[r]\ge pivot$成立。

故有$A[l],\dots,A[i_{n+1}-1]\le pivot, A[j_{n+1}+1],\dots,A[r]\ge pivot$成立。

原命题得证。

由于已证：外层循环进行n次后，恒有 $A[l],\dots,A[i_n-1]\le pivot, A[j_n+1],\dots,A[r]\ge pivot$成立。

故可知外循环结束后，恒有 $A[l],\dots,A[i-1]\le pivot, A[j+1],\dots,A[r]\ge pivot$成立。

原命题得证。

(c)假设外层循环结束时有$i=l$。

由于i的初始值为l，循环结构中对i的值会发生变化的语句皆为 `i=i+1` ，故必须满足 `i=i+1` 的语句从未被执行，即第七行的**while**语句和第九行的**if**语句不可执行。

令第一次循环结束时$i=i_1, j=j_1$。

由于第七行的**while**语句和第九行的**if**语句不可执行，故有以下条件：

$$A[l]\ge pivot$$
$$i_1=l>j_1$$
$$A[j_1]\le pivot$$
$$A[j_1+1]、A[j_1+2]、\dots A[r]>pivot$$

根据以上条件作推断：

$$j_1+1\le i_1<r$$
$$故有A[l]>pivot$$

即满足$A[l]、A[l+1]、\dots、A[r-1]、A[r]>pivot$

根据实际情况，A数组中必须存在等于 `pivot` 的元素，这与以上推论得到的结果相矛盾，故假设不成立，$i\neq l$。

同理，$j=r$同样不可成立，故$j\neq r$。

$\therefore$ 原命题得证，外层循环结束后$i\neq l$且$j\neq r$。

若存在$i=l$或$j=r$,则最后两行的
$QSORT(l,j)$或$QSORT(i,r)$会作为$QSORT(l,r)$再执行，如此便会不停地嵌套执行，程序将不会结束。

由于外层循环结束后$i\neq l$且$j\neq r$，故程序可以终止。

问题2

证:

参考快速排序中基于比较次数概率与期望的方法求解该程序的期望时间复杂度。

将数组A的各个元素重新命名为$z_1, z_2, \ldots, z_n$，其中z_i是数组A中第i小的元素。定义$Z_{ij} = \{z_i, z_{i+1}, \ldots, z_j\}$为$z_i$与$z_j$之间的元素集合。

利用指示器随机变量，定义：$X_{ijk} = \{$查找z_k时z_i与z_j进行比较$\}$

该随机变量的表达式取决于k相对于i,j的位置，由于$i < j$，故存在三种情况：

$$Pr\{\text{查找}z_k\text{时}z_i\text{与}z_j\text{进行比较}\} = E[X_{ijk}] = \begin{cases} \frac{2}{j-i+1} & i \le k \le j \\ \frac{2}{j-k+1} & k \le i < j \\ \frac{2}{k-i+1} & i < j \le k \end{cases}$$

易得找到z_k时，算法的总比较次数$X_k = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} X_{ijk}$，对该式两边取期望，利用期望值的线性特性可以得到：

$E(X_k) = E[\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} X_{ijk}] = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} E[X_{ijk}] = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} Pr\{\text{查找}z_k\text{时}z_i\text{与}z_j\text{进行比较}\}$

对上式进一步简化可得：

$$\begin{aligned}
&E(X_k) \\
&= \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} E[X_{ijk}] \\
&= \sum_{i=1}^{k}\sum_{j=i+1}^{n} E[X_{ijk}] + \sum_{i=k+1}^{n-1}\sum_{j=i+1}^{n} E[X_{ijk}] \\
&= \sum_{i=1}^{k}\sum_{j=i+1}^{k-1} E[X_{ijk}] + \sum_{i=1}^{k}\sum_{j=k}^{n} E[X_{ijk}] + \sum_{i=k+1}^{n-1}\sum_{j=i+1}^{n} E[X_{ijk}] \\
&= \sum_{i=1}^{k-2}\sum_{j=i+1}^{k-1} E[X_{ijk}] + \sum_{i=1}^{k}\sum_{j=k}^{n} E[X_{ijk}] + \sum_{i=k+1}^{n-1}\sum_{j=i+1}^{n} E[X_{ijk}] \\
&= \sum_{i=1}^{k-2}\sum_{j=i+1}^{k-1} \frac{2}{k-i+1} + \sum_{i=1}^{k}\sum_{j=k}^{n} \frac{2}{j-i+1} + \sum_{i=k+1}^{n-1}\sum_{j=i+1}^{n} \frac{2}{j-k+1} \\
&= \sum_{i=1}^{k-2}\sum_{j=i+1}^{k-1} \frac{2}{k-i+1} + \sum_{i=1}^{k}\sum_{j=k}^{n} \frac{2}{j-i+1} + \sum_{j=k+2}^{n}\sum_{i=k+1}^{j-1} \frac{2}{j-k+1} \\
&= \sum_{i=1}^{k-2} \frac{2(k-i-1)}{k-i+1} + \sum_{i=1}^{k}\sum_{j=k}^{n} \frac{2}{j-i+1} + \sum_{j=k+2}^{n} \frac{2(j-k-1)}{j-k+1} \\
&\le \sum_{i=1}^{k-2} \frac{2(k-i-1)}{k-i+1} + \sum_{i=1}^{k}\sum_{j=k}^{n} \frac{2}{j-i+1} + \sum_{j=k+1}^{n} \frac{2(j-k-1)}{j-k+1}
\end{aligned}$$

对以上得到的$E[X_k]$的三项进行分析如下：

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

对$\sum_{i=1}^{k}\sum_{j=k}^{n} \frac{1}{j-i+1}$分析：

该求和公式中的每一项皆为$\frac{1}{m}$的形式，$i = j = k$时$\frac{1}{m}$有最大值1；$i = 1, j = n$时$\frac{1}{m}$有最小值$\frac{1}{n}$。

根据该求和公式的性质可知：值为$\frac{1}{m}$的项最多有m个，即满足$j - i + 1 = m$的(i, j)最多存在m个，故这些项相加得到的和最大为1。

故$\sum_{i=1}^{k}\sum_{j=k}^{n} \frac{1}{j-i+1} \le \sum_{m=1}^{n} 1 = n$

则$E[X_k] \le \sum_{i=1}^{k-2} \frac{2(k-i-1)}{k-i+1} + \sum_{i=1}^{k}\sum_{j=k}^{n} \frac{2}{j-i+1} + \sum_{j=k+1}^{n} \frac{2(j-k-1)}{j-k+1} \le 2n + 2n = 4n$

由于已证$E[X_k] \le 4n$，故可以得出结论：
使用$RANDOMPARTITION$,在输入元素互异的情况下，该程序的期望时间复杂度为$O(n)$。

问题3

答:

(a)假设对于1%的全排列，对基于比较排序的算法A，能达到线性运行时间。

令比较排序算法对应的决策树高度为h，最后一层结点数为$\frac{n!}{100}$，可得到：

$$2^h \geq \frac{n!}{100}$$

$$h \geq log(\frac{n!}{100}) = log(n!) - log(100) = \Omega(nlgn)$$

∴原假设不成立，故至少**99%**的全排列在算法A中运行时间为Ω(nlogn)。

即$Pr_\sigma[Time(A(\sigma)) \geq 0.5nlgn] \geq 0.99$，原命题得证。

(b)附加题

可以假设算法A最多使用n^2个随机比特。则对于随机化算法，对应的**random string r**取值范围为$\{0,1\}^{n^2}$，即存在2^{n^2}种不同的**random string r**。

根据(a)中可知，对于任何基于比较排序的算法A，对99%的全排列，A的运行时间为Ω(nlogn)。

故对2^{n^2}个**random string r**中任意一种**r**，对至少99%的全排列，运行时间为Ω(nlogn)。

要证对于$n!$个全排列存在一种全排列，使得对于2^{n^2}种**random string r**对应的随机化算法的期望时间为Ω(nlogn)。

假设对于$\frac{1}{2^n}$的全排列，对基于比较排序的算法A，能达到线性运行时间。

同(a)，有

$$2^h \geq \frac{n!}{2^n}$$

$$h \geq log(\frac{n!}{2^n}) = log(n!) - n = \Omega(nlgn)$$

故原假设不成立，至少$1 - \frac{1}{2^n}$的全排列在算法A中运行时间为Ω(nlogn)。

故$E(T) \geq \frac{1}{2^{n^2}} \sum_{i=1}^{2^{n^2}} [(1 - \frac{1}{2^n}) * nlogn + \frac{1}{2^n} * n] = \frac{1}{2^{n^2}} \sum_{i=1}^{2^{n^2}} [nlogn - \frac{nlogn - n}{2^n}] = \Omega(nlgn)$

故A的期望时间为Ω(nlogn)，即$E_r[Time(A(\sigma, r))] \geq 0.5nlogn$，原命题得证。

(c)附加题

利用马尔科夫不等式：$P(X \geq a) \leq \frac{EX}{a}$

则有$Pr_\sigma[E_r[Time(A(\sigma, r))] \geq 0.5nlogn] \leq \frac{E_\sigma[E_r[Time(A(\sigma,r))]}{0.5nlogn}$

问题4

答:

(a)设计算法伪代码如下：

```
function Search(A, key)    //key为A[j+1]的值
    low=1
    high=n
    while low≤high do
        mid=(low+high)/2    //折半查找
        if A[mid]==key then    //查找到了一样大的元素，直接插入该位置
            return mid
        else if A[mid]>key then    //向左折半
            high=mid-1
        else       //向右折半
            low=mid+1
    return low    //返回插入位置
```

(b)

时间复杂度证明如下：

初始数组长度为high-low+1=n，经过一次循环后由于查找了数组的中间元素A[mid]并选择通过中间元素重新对上界high或下界low赋值，故查找的数组长度变为$\frac{n}{2}$。同理，第二次循环后查找数组的长度为$\frac{n}{2^2}$，第三次循环后查找数组的长度为

$\frac{n}{2^3}$，第k次循环后查找数组的长度为$\frac{n}{2^k}$。

故有：$\frac{n}{2^k} \geq 1 => k \leq logn$。

即可证得该算法的时间复杂度为$O(logn)$。

算法正确性证明如下：

循环不变式：

在每次循环中，key的插入位置在[low, high+1]范围内。

初始化：

第一轮循环前，low=1、high+1=j+1，key的插入位置必然在[1, j+1]范围内。

保持：

某次循环迭代之前，满足`key的插入位置在[low, high+1]范围内`，开始循环后：

1. 若A[mid]=key，则在数组中查找到了与将要插入的key相同的元素，可直接在此位置插入，显然插入后依旧满足排序正确性，程序返回。
2. 若A[mid]>key，则要插入的元素key小于A[mid]，即key必须插在A[mid]的左边，即插在[low,mid]范围内。由于执行了`high=mid-1`，故此轮循环结束后key应插在[low, high+1]范围内。
3. 若A[mid]<key，则要插入的元素key大于A[mid]，即key必须插在A[mid]的左边，即插在(mid,high+1]范围内。由于执行了`low=mid+1`，故此轮循环结束后key应插在(low-1, high+1]范围内，即[low, high+1]范围内。

故证得，某次迭代之前满足`key的插入位置在[low, high+1]范围内`，这次迭代结束后依旧满足。

终止：

循环终止时满足`low>high`，由于最后一轮迭代之前满足`low≤high`，且只执行了`high=mid-1`、`low=mid+1`中的一句。故循环终止时必有`low=high+1`，故[low,high+1]范围内仅包含**low**一个元素，即返回**low**。

由以上可证，该算法正确。