Lab01-AlgorithmAnalysis

CS214-Algorithm and Complexity, Xiaofeng Gao, Spring 2020.

$*$ If there is any problem, please contact TA Shuodian Yu.

$*$ Name:Hanzhang Yang Student ID:518030910022 Email: linqinluli@sjtu.edu.cn

1. Please analyze the time complexity of Alg. 1 with brief explanations.

```
Algorithm 1: PSUM
  Input: $n = k^2$, $k$ is a positive integer.
  Output: $\sum_{i=1}^{j} i$ for each perfect square $j$ between 1 and $n$.
1 $k \leftarrow \sqrt{n}$;
2 for $j \leftarrow 1$ to $k$ do
3     $sum[j] \leftarrow 0$;
4     for $i \leftarrow 1$ to $j^2$ do
5         $sum[j] \leftarrow sum[j] + i$;
6 return $sum[1 \cdots k]$;
```

Solution. Define $T(n)$ the time complexity.

$$\begin{aligned}
T(n) &= \sum_{j=1}^{\sqrt{n}} (1 + \sum_{i=1}^{j^2} 1) \\
&= \sum_{j=1}^{\sqrt{n}} (1 + j^2) \\
&= \sqrt{n} + 1^2 + 2^2 + 3^2 + \cdots + \sqrt{n}^2 \\
&= \sqrt{n} + \frac{\sqrt{n}(\sqrt{n}+1)(2\sqrt{n}+1)}{6}
\end{aligned}$$

Thus $T(n) = \Theta(n^{\frac{3}{2}})$ □

2. Analyze the **average** time complexity of QuickSort in Alg. 2.

```
Algorithm 2: QuickSort
  Input: An array $A[1, \cdots, n]$
  Output: $A[1, \cdots, n]$ sorted nonincreasingly
1 $pivot \leftarrow A[n]$; $i \leftarrow 1$;
2 for $j \leftarrow 1$ to $n - 1$ do
3     if $A[j] < pivot$ then
4         swap $A[i]$ and $A[j]$;
5         $i \leftarrow i + 1$;
6 swap $A[i]$ and $A[n]$;
7 if $i > 1$ then QuickSort($A[1, \cdots, i-1]$);
8 if $i < n$ then QuickSort($A[i+1, \cdots, n]$);
```

Solution. Because the main time cost is on the comparsion. Let's think about two elements $A[i], A[j] (j > i)$. There must be such a sequence $\{A[i], A[i+1], \cdots, A[j-1], A[j]\}$

Whether $A[i] \quad A[j]$ will be compared depends on the *pivot*.

If the *pivot* is outside, the sequence still exists and the probability exists. Thus for this sequence the *pivot* should be between $A[i] and A[j]$

If the *pivot* is $A[i]$ or $A[j]$, they will be compared. Otherwise not.

So the probability that $A[i] quad A[j]$ are compared is $\frac{2}{j-i+1}$.

Define $T(n)$ the time cost. We can take the expectation of all number's comparisons as time cost. (Comparsion means 1. Otherwise means 0)

$$
\begin{aligned}
T(n) &= \sum_{i=1}^{n} \sum_{j=i+1}^{n} \frac{2}{j-i+1} \\
&= \sum_{i=1}^{n} \frac{2}{2} + \frac{2}{3} + \frac{2}{4} + \cdots + \frac{2}{n-i+1} \\
&= \sum_{i=1}^{n} O(logn) \\
&= O(nlogn)
\end{aligned}
$$

Thus the average time complexity is $O(nlogn)$.

□

3. The BubbleSort mentioned in class can be improved by stopping in time if there are no swaps during an iteration. An indicator is used thereby to check whether the array is already sorted. Analyze the **average** and **best** time complexity of the improved BubbleSort in Alg. 3.

Algorithm 3: BubbleSort

```
Input: An array $A[1, \ldots, n]$
Output: $A[1, \ldots, n]$ sorted nonincreasingly
1 $i \leftarrow 1$;$sorted \leftarrow false$;
2 while $i \leq n-1$ and not $sorted$ do
3     $sorted \leftarrow true$;
4     for $j \leftarrow n$ downto $i+1$ do
5         if $A[j] < A[j-1]$ then
6             interchange $A[j]$ and $A[j-1]$;
7             $sorted \leftarrow false$;
8     $i \leftarrow i+1$;
```

Solution. Best $\Omega(n)$

$A[1, \cdots, n]$ has been sorted already. After traversing the array, sorted==true and the cycle ends. Traversing costs $\Omega(n)$

Average $\Omega(n^2)$

Define $A_r[n, n-1, \cdots, 1]$ whose elements are in the reverse order of A. Thinking about $(x, y) \quad y > x$. Then this order must exist in one of the two array. Thus there are $\frac{n(n-1)}{2}$ orders in $A_r andA$ totally. The average inversion number of A is $\frac{n(n-1)}{4}$.

Because the BubbleSort exchange only adjacent elements and reduce only one reverse order at a time. It needs $\Omega(n^2)$ exchange. And the worst time complexity is $O(n^2)$. So the average time complexity is $\Omega(n^2)$.

□

4. Rank the following functions by order of growth with brief explanations: that is, find an arrangement $g_1, g_2, \ldots, g_{15}$ of the functions $g_1 = \Omega(g_2), g_2 = \Omega(g_3), \ldots, g_{14} = \Omega(g_{15})$. Partition your list into equivalence classes such that functions $f(n)$ and $g(n)$ are in the same class if and only if $f(n) = \Theta(g(n))$. Use symbols "=" and "$\prec$" to order these functions appropriately.

$2^{\lg n}$	$(\lg n)^{\lg n}$	n^2	$n!$	$(n+1)!$
2^n	n^3	$\lg^2 n$	e^n	2^{2^n}
$\lg \lg n$	$n \cdot 2^n$	n	$\lg n$	$4^{\lg n}$

Solution. L'Hospital's rule

$$lglgn \prec lgn \prec lg^2n \prec n = 2^{lgn} \prec (lgn)^{lgn} \prec 4^{lgn} = n^2 \prec n^3 \prec 2^n \prec n{\cdot}2^n \prec e^n \prec n! \prec (n{+}1)! \prec 2^{2^n}$$

$\lim_{n\to\infty} \frac{lglgn}{lgn} = \frac{1}{lgnln10} = 0(L'Hospital'srule) \to \quad lglgn \prec lgn$

$\lim_{n\to\infty} \frac{lg^2n}{n} = \frac{2lgn}{nln2} = 0(L'Hospital'srule) \to \quad lg^2n \prec n$

$2^{lgn} = n \quad 4^{lgn} = n^2$

$(n{+}1)! < (n{+}1)^{n+1} \quad ln(n{+}1)^{n+1} = (n{+}1)ln(n{+}1) \quad ln2^{2^n} = 2^n ln2 > n^3 > (n{+}1)ln(n{+}1)$

Thus $(n+1)! \prec 2^{2^n}$

□

Remark: You need to include your .pdf and .tex files in your uploaded .rar or .zip file.