Lab01-AlgorithmAnalysis

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

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

∗ Name:Yijia Diao(刁义嘉) Student ID:518030910146 Email: diao_yijia@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. Time complexity =

$$\sum_{j=1}^{k}\sum_{i=1}^{j^2} 1 = \sum_{j=1}^{k} j^2 = \frac{k(k+1)(2k+1)}{6} = \Theta(k^3) = \Theta(n^{1.5})$$

□

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 nondecreasingly
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. Suppose the average time complexity of QuickSort is $T(n)$, when the size of array is n. Between line 1 and line 5, since line 3 is executed in every cycle, the time complexity of this part $\approx n - 1$. Then we have

$$\begin{aligned} T(n) =& \Pr(i=1)(n-1+T(i-1)) + \Pr(i=n)(n-1+T(n-i)) \\ &+ \sum_{k=2}^{n-1} \Pr(i=k)(n-1+T(i-1)+T(n-i)) \end{aligned}$$

Since i is uniformly distributed between 1 and n, $\Pr(i) = \frac{1}{n}$, then we have

$$T(n) = n - 1 + \frac{2}{n}\sum_{k=1}^{n-1} T(k) \tag{1}$$

According to equation 1, we have

$$T(n+1) = n + \frac{2}{n+1}\sum_{k=1}^{n} T(k) \tag{2}$$

Equation 1 $-$ equation 2, we have

$$\frac{T(n+1)}{n+2} = \frac{2n}{(n+1)(n+2)} + \frac{T(n)}{n+1} \tag{3}$$

After iteration, we have

$$\begin{aligned}\frac{T(n)}{n+1} &= \sum_{l=2}^{n} \frac{2(l-1)}{l(l+1)} + \frac{T(1)}{2}\\ &= 2\sum_{l=2}^{n}\left(\frac{1}{l+1} - \frac{1}{l(l+1)}\right) + \frac{T(1)}{2}\end{aligned}$$

Then we get

$$T(n) = 2(n+1)\sum_{l=2}^{n} \frac{1}{n+1} - \frac{n+1}{2} + 2 \tag{4}$$

According to equation 4, we can conclude that

$$T(n) = O(n \log n)$$

which means that the average time complexity of QuickSort is $O(n \log n)$. □

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 $A[1, \dots, n]$
Output: $A[1, \dots, n]$ sorted nondecreasingly
$i \leftarrow 1$;$sorted \leftarrow false$;
while $i \le n-1$ and not $sorted$ do
    $sorted \leftarrow true$;
    for $j \leftarrow n$ downto $i+1$ do
        if $A[j] < A[j-1]$ then
            interchange $A[j]$ and $A[j-1]$;
            $sorted \leftarrow false$;
    $i \leftarrow i+1$;
```

Solution. Suppose that when $i = k$, there are no swaps during an iteration, $k \le n-1$; the average time complexity of Alg. 3 is $T(n)$. Since k is uniformly distributed between 1 and n-1, $\Pr(k) = \frac{1}{n-1}$. Since

$$T(n) \le \sum_{k=1}^{n-1} \Pr(k)(\sum_{i=1}^{k}\sum_{j=n}^{i+1} 3)$$

and

$$\sum_{k=1}^{n-1} \Pr(k)(\sum_{i=1}^{k}\sum_{j=n}^{i+1} 3) = \frac{3}{n-1}\sum_{k=1}^{n-1}(nk - \frac{k(k+1)}{2}) = n(2n-1)$$

, we have

$$T(n) \le n(2n-1) = O(n^2)$$

Then we can conclude that the **average** time complexity of Alg. 3 is $O(n^2)$.

The best case is that the array is already sorted nondecreasingly. So the execution time is

$$\sum_{j=n}^{2} 1 = n - 1 = \Omega(n)$$

Then we can conclude that the **best** time complexity of Alg. 3 is $\Omega(n)$. □

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. Here $\log n$ stands for $\log_2 n$.

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

Solution.

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

Now prove some of the relations above.

① $n^3 \prec (\log n)^{\log n}$.

$$\lim_{n\to\infty} \frac{n^3}{(\log n)^{\log n}} \overset{k=\log n}{=} \lim_{k\to\infty} \frac{8^k}{k^k} = 0$$

② $(\log n)^{\log n} \prec 2^n$. Since

$$\lim_{k\to\infty} \frac{k\log k}{2^k} = 0$$

we have

$$\lim_{n\to\infty} \frac{(\log n)^{\log n}}{2^n} \overset{k=\log n}{=} \lim_{k\to\infty} \frac{k^k}{2^{2^k}} = \lim_{k\to\infty} \frac{2^{k\log k}}{2^{2^k}} = 0$$

③ $e^n \prec n!$

$$0 \le \lim_{n\to\infty} \frac{e^n}{n!} = \lim_{n\to\infty} \frac{e \cdot e \cdot e \cdot \ldots \cdot e}{1 \cdot 2 \cdot 3 \cdot \ldots \cdot n} \le \frac{e^2}{2} \lim_{n\to\infty} \frac{e}{n} = 0$$

④ $(n+1)! \prec 2^{2^n}$. Since

$$0 \le \lim_{n\to\infty} \frac{\sum_{i=1}^{n+1} \log i}{2^n} \le 2 \lim_{n\to\infty} \frac{(n+1)\log(n+1)}{2^{n+1}} = 0$$

we have

$$\lim_{n\to\infty} \frac{(n+1)!}{2^{2^n}} = \lim_{n\to\infty} \frac{2^{\sum_{i=1}^{n+1} \log i}}{2^{2^n}} = 0$$

$\square$

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