Lab01-AlgorithmAnalysis

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

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

$*$ Name: ShaoxiongLin Student ID: 517010910028 Email: Johnson-Lin@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. The outer for loop will run k times, and in the jth iteration, the inner for loop will run j^2 times. Thus the total number of iterations is

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

Since

$$\lim_{x\to\infty} \frac{2n^{\frac{3}{2}} + 3n + n^{\frac{1}{2}}}{6n^{\frac{3}{2}}} = \frac{1}{3}$$

we can conclude the time complexity of Alg. 1 is $\Theta(n^{\frac{3}{2}})$. $\square$

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. We denote the number of basic operations to sort an array with n elements is $T(n)$, and we have $T(0) = T(1) = 0$. The time complexity of QuickSort depends on where the *pivot* locates in the finally sorted array. To analyze the average time complexity of QuickSort, we assume the probability that the *pivot* locates in each slot of the array is equal. Thus, we have

$$T(n) = (n-1) + \frac{1}{n}\sum_{j=0}^{n-1}(T(j) + T(n-j-1))$$

To get the order of $T(n)$, we multiply n to both of the two sides of the equation and get

$$nT(n) = n(n-1) + 2\sum_{j=0}^{n-1} T(j)$$

Subtracting this from the same equation for $n-1$ gives

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

We denote $\frac{T(n)}{n+1}$ as a_n, and we have

$$a_n - a_{n-1} = \frac{2}{(n+1)} - \frac{2}{n(n+1)}$$

Since $a_0 = 0$ and $a_n = a_n - a_{n-1} + a_{n-1} - a_{n-2} + \cdots + a_1 - a_0$, we have

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

From the above equation, we can easily get that the order of $T(n)$ 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
1 $i \leftarrow 1$;$sorted \leftarrow false$;
2 while $i \le n-1$ 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. The best case appears when the input array is already sorted in nondecreasing order, in this case the outer while loop and the inner for loop will only execute one time, so the best time complexity is $\Omega(n)$.

To sort the array with the improved BubbleSort, the outer while loop will run at least 1 time, at most $n-1$ times. If the outer while loop run k times, the total number of basic operations is $\sum_{i=1}^{k}\sum_{j=i+1}^{n} 1 = \sum_{i=1}^{k}(n-i)$. We assume the probability that k equals to any integer ranging from 1 to $n-1$ is the same. Thus, we have the average time complexity is

$$\frac{1}{n-1}\sum_{k=1}^{n-1}\sum_{i=1}^{k}(n-i) = \frac{1}{n-1}\sum_{k=1}^{n-1}(kn - \frac{k^2}{2} - \frac{k}{2}) = \frac{2n^2-n}{6} = O(n^2)$$

□

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. It is easy to conclude that $\lg\lg n \prec \lg n$. Since $\lim_{n\to\infty}\frac{n}{\lg^2 n} = +\infty$, $\lim_{n\to\infty}\frac{n}{2^{\lg n}} = 1$, $\lim_{n\to\infty}\frac{n^2}{4^{\lg n}} = 1$ and $\lim_{n\to\infty}\frac{(\lg n)^{\lg n}}{n^3} = +\infty$, thus we have

$$\lg\lg n \prec \lg n \prec \lg^2 n \prec 2^{\lg n} = n \prec 4^{\lg n} = n^2 \prec n^3 \prec (\lg n)^{\lg n}$$

Since $\lim_{n\to\infty}\frac{2^n}{(\lg n)^{\lg n}} = +\infty$ and $\lim_{n\to\infty}\frac{e^n}{n\cdot 2^n} = +\infty$ so $2^n = \omega((\lg n)^{\lg n})$. Now we have

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

□

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