Lab01-AlgorithmAnalysis

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

∗ If there is any problem, please contact TA Shuodian Yu.
∗ Name:Zehao Wang Student ID:518021910976 Email: davidwang200099@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 ← √n;
2 for j ← 1 to k do
3 |   sum[j] ← 0;
4 |   for i ← 1 to j^2 do
5 |   |   sum[j] ← sum[j] + i;
6 return sum[1 ··· k];
```

Solution. The for-loop in Line 2 will be executed for k times.

The for-loop in Line 4 will be executed for j^2 times.

The number of execution in total:

$$\begin{aligned}
&\sum_{j=1}^{k}(1+\sum_{i=1}^{j^2}1)\\
&=\sum_{j=1}^{k}1+\sum_{j=1}^{k}\sum_{i=1}^{j^2}1\\
&=\sqrt{n}+\sum_{j=1}^{k}\frac{j^4+j^2}{2}\\
&=\sqrt{n}+\frac{k(k+1)(2k+1)}{12}+\sum_{j=1}^{k}\frac{j^4}{2}\\
&\approx\sqrt{n}+\frac{k(k+1)(2k+1)}{12}+\int_1^k\frac{x^4}{2}dx\\
&=\sqrt{n}+\frac{k(k+1)(2k+1)}{12}+\frac{k^5}{10}\\
&=\sqrt{n}+\frac{\sqrt{n}(\sqrt{n}+1)(2\sqrt{n}+1)}{12}+\frac{n^{\frac{5}{2}}}{10}
\end{aligned}$$

So the complexity of this algorithm is $O(n^{\frac{5}{2}})$

□

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 ← A[n]; i ← 1;
2 for j ← 1 to n − 1 do
3 |   if A[j] < pivot then
4 |   |   swap A[i] and A[j];
5 |   |   i ← i + 1;
6 swap A[i] and A[n];
7 if i > 1 then QuickSort(A[1, ··· , i − 1]);
8 if i < n then QuickSort(A[i + 1, ··· , n]);
```

Solution. For any i,$j(1 \leq i < j \leq n)$, the probability that A[i] and A[j] are compared is $\frac{1}{j-i+1}$.Any element in A[i,$\cdots$,j] can be selected as a pivot, and the probability is $\frac{1}{j-i+1}$.A[i] and A[j] are compared only when A[j] is selected as a pivot.

Therefore the average comparison time is:

$\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{j-i+1}$

$= \sum_{i=1}^{n-1} \sum_{k=1}^{n-i} \frac{1}{k+1}$

$< \sum_{i=1}^{n-1} \sum_{k=1}^{n} \frac{1}{k} = O(nlogn)$.

Most of the operations are comparison and optional swap.

So the average time complexity of QuickSort 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 ← 1;sorted ← false;
2 while i ≤ n − 1 and not sorted do
3 |   sorted ← true;
4 |   for j ← n downto i + 1 do
5 |   |   if A[j] < A[j − 1] then
6 |   |   |   interchange A[j] and A[j − 1];
7 |   |   |   sorted ← false;
8 |   i ← i + 1;
```

Solution. It has been proved that the worst time complexity of BubbleSort is $O(n^2)$.

The best case happens when all the elements are sorted.Then we only need to traverse the array once. The best time complexity of BubbleSort is $\Theta(n)$.

As for average case,we first pay attention to the number of swap operation.

The number of swap operation depends on the inversion number of the array.If the inversion number is k, there will be k swaps.

The maximum of inversion number is $\frac{n(n-1)}{2}$, and the minimum of inversion number is 0.

For an array with an inversion number of X, we can find a unique corresponding array with an inversion number of $\frac{n(n-1)}{2} - X$ by swaping the k-th largest element and the $(n-k)$-th largest element.($k = 1, 2, \cdots, \frac{n}{2}$)

So the mathematical expectation of inversion number as well as the number of swap operation is $\frac{n(n-1)}{4}$.

So the average time complexity of improved BubbleSort is at least $\Omega(n^2)$.

Taking the comparison operation into consideration, the time complexity is also $\Omega(n^2)$.

Because $O(n^2) \leq$ average complexity $\leq$ worst complexity$= O(n^2)$

Therefore the average time complexity of improved BubbleSort is $O(n^2)$ and the best average time complexity of improved BubbleSort is $\Theta(n)$. □

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.

$2^{\lg n}$	$(\lg n)^{\lg n}$	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.

$$\lg \lg n \prec \lg n \prec \lg^2 n \prec n = 2^{\lg n} \prec n^2 = 4^{\lg n} \prec n^3 \prec 2^n \prec n2^n \prec e^n \prec n! \prec (n+1)! \prec 2^{2^n}$$

The reason for the rank above is as follows.

(a) $n < \lg n, \lg n < \lg^2 n, \lg^2 n < n$ So $\lg \lg n \prec \lg n$, so $\lg \lg n \prec \lg n \prec \lg^2 n \prec n$

(b) It is obvious that $2^{\lg n} = n$, $n < n^2$, $4^{\lg n} = 2^{2 \lg n} = 2^{\lg n^2} = n^2$

(c) It is obvious that $n^2 < n^3 < 2^n < e^n$.

(d) Because $\frac{e}{2} > 1$,therefore $(\frac{e}{2})^n > n$. Therefore $n2^n < e^n$

(e) It can be proved that $e^n < 3^n < (n-1)!$ when $n >= 7$

(f) It is obvious that $(n+1)! > n!$

(g) $\lg 2^{2^n} = 2^n \approx \int_1^n 2^x \ln 2 dx$, $\lg(n+1)! = \sum_{i=1}^{n+1} \lg i \approx \int_1^n \lg(x+1) dx$
Because $2^x \ln 2 > \lg(n+1)$, therefore $2^{2^n} > (n+1)!$

□

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