Lab01-AlgorithmAnalysis

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

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

∗ Name: Futao Wei   Student ID: 518021910750   Email: weifutao@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.

$k \leftarrow \sqrt{n}$;
for $j \leftarrow 1$ **to** k **do**
  $sum[j] \leftarrow 0$;
  for $i \leftarrow 1$ **to** j^2 **do**
    $sum[j] \leftarrow sum[j] + i$;
return $sum[1 \cdots k]$;

Solution.
$\Theta(n^{\frac{3}{2}})$. The outer loop iterates k times and the inner loop iterates j^2 times. Hence we have

$$\begin{aligned} \text{total addition} &\sim \sum_{j=1}^{k} j^2 \\ &= \frac{k(k+1)(2k+1)}{6} \\ &\sim k^3 \\ &= n^{\frac{3}{2}} \end{aligned}$$

□

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

$pivot \leftarrow A[n]$; $i \leftarrow 1$;
for $j \leftarrow 1$ **to** $n - 1$ **do**
  if $A[j] < pivot$ **then**
    swap $A[i]$ and $A[j]$;
    $i \leftarrow i + 1$;
swap $A[i]$ and $A[n]$;
if $i > 1$ **then** QuickSort$(A[1, \cdots, i-1])$;
if $i < n$ **then** QuickSort$(A[i+1, \cdots, n])$;

Solution.
$O(n \log n)$.

Let T_n be the expected number of comparisons. $T_0 = 0$, $T_1 = 1$.
Assume that the position where the pivot settles down is uniformly distributed, i.e., the pivot stops at position $1, \cdots, n$ with the same probability $\frac{1}{n}$.
As the recursion indicates, we have

$$T_n = \sum_{i=1}^{n} \frac{1}{n}(T_{i-1} + T_{n-i} + n) = n + \frac{2}{n}\sum_{i=0}^{n-1} T_i = n + \frac{2}{n}\sum_{i=1}^{n-1} T_i$$

Denote $\sum_{i=1}^{n} T_i$ as S_n, we have

$$T_n = n + \frac{2}{n} S_{n-1} \tag{1}$$

$$T_{n-1} = n - 1 + \frac{2}{n-1} S_{n-2} \tag{2}$$

$(1) \times n - (2) \times (n-1)$ results in

$$\begin{aligned} nT_n - (n-1)T_{n-1} &= n^2 - (n-1)^2 + 2(S_{n-1} - S_{n-2}) \quad (3) \\ &= 2n - 1 + 2T_{n-1} \quad (4) \end{aligned}$$

With (4) we have

$$\frac{T_n}{n+1} - \frac{T_{n-1}}{n} = \frac{3}{n+1} - \frac{1}{n} \tag{5}$$

Then we can obtain T_n by unfolding

$$\frac{T_n}{n+1} - \frac{T_1}{2} = 2\left(\frac{1}{3} + \frac{1}{4} + \cdots + \frac{1}{n}\right) + \frac{3}{n+1} - \frac{1}{2} \tag{6}$$

$$T_n = 2(n+1)\left(\frac{1}{3} + \frac{1}{4} + \cdots + \frac{1}{n}\right) + 3 \sim n \log n \tag{7}$$

Therefore, 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 $A[1, \ldots, n]$
Output: $A[1, \ldots, n]$ 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.

Best Case: $\Omega(n)$.
The best case happens when the array is already sorted.
Average Case: $O(n^2)$.
For average case analysis,

$$\text{comparison times} \sim n + \text{inversion number}$$

(a pair $(A[i], A[j])$ is inverted if $i < j$ and $A[i] > A[j]$).
There are $C_n^2 = \frac{n(n-1)}{2}$ pairs in $A[1, \ldots, n]$. In average, half of them will be inversions, i.e.,

$$E(\text{inversion number}) = \frac{n(n-1)}{4}$$

. Hence we have

$$\begin{aligned} E(\text{comparison times}) &\sim n + \frac{n(n-1)}{4} \\ &\sim n^2 \end{aligned}$$

$\square$

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 $\log_2 n$.

$$\begin{array}{ccccc} 2^{\log n} & (\log n)^{\log n} & n^2 & n! & (n+1)! \\ 2^n & n^3 & \log^2 n & e^n & 2^{2^n} \\ \log\log n & n \cdot 2^n & n & \log n & 4^{\log n} \end{array}$$

Solution.

$$\begin{aligned} &\log\log n \prec \log n \prec \log^2 n \prec n = 2^{\log n} \prec \\ &\qquad 4^{\log n} = n^2 \prec n^3 \prec (\log n)^{\log n} \prec 2^n \prec \\ &\qquad\qquad n \cdot 2^n \prec e^n \prec n! \prec (n+1)! \prec 2^{2^n} \end{aligned}$$

Explanations:

- $n^3 \prec (\log n)^{\log n} \prec 2^n$

$$\begin{aligned} \log n^3 &= 3\log n \\ \log[(\log n)^{\log n}] &= \log n(\log\log n) \\ \log(2^n) &= n \end{aligned}$$

- $n \cdot 2^n \prec e^n$

$$\lim_{n\to\infty} \frac{n \cdot 2^n}{e^n} = \lim_{n\to\infty} \frac{n}{(\frac{e}{2})^n} = 0$$

- $e^n \prec n!$
  Prove by induction since $k+1 > e$ from some point k_0.
- $(n+1)! \prec 2^{2^n}$
  Prove by induction since $2^{2^k} > k+2$ from some point k_0.

$\square$

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