COMS21103: Data Structures & Algorithms

Quiz Answers

- **Question 1: [20 Points]** For each of the following functions, give the **tightest** asymptotic upper bound. Your answer should be one of those in the list (in no particular order):

  $$O(n^2),\ O(2^n),\ O(n),\ O(1),\ O(n \log n),\ O(n^3),\ O(\log n),\ O(n^2 \log n)$$

  Note: Not all of the above are used and some might appear more than once. Give the tightest upper bound, e.g. if your answer is "$O(n^3)$" but $f(n)$ is also $O(n^2)$, you will not get any points. You do **NOT** have to prove your answer.

  a) $f(n) = n^2 + 100n + 1000$ — Correct answer is $O(n^2)$

  b) $f(n) = n(10n + n^2) + 500$ — Correct answer is $O(n^3)$

  Hint: $n(10n + n^2) = 10n^2 + n^3$.

  c) $f(n) = 2^{100} + 3^{10}$ — Correct answer is $O(1)$

  Hint: $f(n)$ is independent of n.

  d) $f(n) = n^2 \log n + n \log (n^2)$ — Correct answer is $O(n^2 \log n)$

- **Question 2: [10 Points]** Answer each of the following with either **True** or **False**. You do **NOT** need to justify your answer:

  a) MergeSort has the same best-case and worst-case time complexities. (True)

  b) If $f(n) = n^2 + 100n$, $f(n) = \Theta(n)$. (False)

  Hint: $f(n) = \Omega(n)$ but $f(n) \neq O(n)$. Actually, $f(n) = \Theta(n^2)$.

  c) An algorithm with time $\Theta(2^n)$ is asymptotically better than an algorithm with time $\Theta(n^4)$. (False)

  Hint: 2^n grows much faster than n^4.

  d) If $f(n) = n + (n-1) + (n-2) + \ldots + 1$, $f(n) = O(n)$ (False)

  Hint: This is an Arithmetic Series. (See Slide 4 of Lecture 2.)
  $n + (n-1) + (n-2) + \ldots + 1 = \sum_{i=1}^{n} i = \frac{n(n+1)}{2} = \frac{n^2+n}{2} = O(n^2)$.

- **Question 3: [30 Points]** For each of the following segments of code, give a tight asymptotic upper bound (i.e. Big-Oh in terms of n) on the running time of the worst case.

a)

```
For i := 1 To n
    For k := 1 To Min(10,i)
        sum := sum + k
```

$O(n)$

Note: Min returns the smallest of its two arguments

b)

```
SomeAlg(n)
  If n = 0 Then
        Return 0
  Else
        Return (n+ SomeAlg(n – 1))
```

$O(n)$

c)

```
For i := 1 To n
        For k := 1 To n*n
            sum := sum + k
```

$O(n^3)$

- **Question 4: [5 Points]** What is the recurrence relation for the running time of the following algorithm. Be sure to include the **base** case. You do **NOT** need to solve the relation.

```
SomeAlg(n)
  If n > 0 Then
        Return (SomeAlg(n/2) + SomeAlg(n − 2) +7)
  Else
        Return 1
```

Answer: The question is asking for the recurrence relation for the **running time** of the algorithm and not the relation for the function computed by the algorithm! We have seen many examples of this in Lecture 5.

$$T(n) = \begin{cases} a & n \leq 0 \\ T(\frac{n}{2}) + T(n-2) + b & n > 0 \end{cases}$$

OR

$$T(n) = \begin{cases} \Theta(1) & n \leq 0 \\ T(\frac{n}{2}) + T(n-2) + \Theta(1) & n > 0 \end{cases}$$

OR

$$T(n) = \begin{cases} O(1) & n \leq 0 \\ T(\frac{n}{2}) + T(n-2) + O(1) & n > 0 \end{cases}$$

OR

$$T(n) = \begin{cases} 1 & n \leq 0 \\ T(\frac{n}{2}) + T(n-2) + 1 & n > 0 \end{cases}$$

All of the above are correct answers.

- **Question 5: [20 Points]** Prove using the substitution method that

$$T(n) = 8T(\frac{n}{8}) + n \text{ is } O(n \log n)\cdot$$

Assume $T(1) = 1$. Show all your working.

Answer:

Always read the question first so you know what is required. The question is asking you to **Prove** using the *Substitution Method*. A few of you have used the iteration method (the backward substitution) method which is the wrong method to use here! I have made it clear in the lecture that the iteration method and recursion trees give an intuition rather than a proof and you can easily get the wrong answer.

PROOF.

- Our guess here is $T(n) = O(n \log n)$.
- We want to prove that $\forall n \geq n_0$, $T(n) \leq cn \log n$ for some constant $c > 0$.
- **IH**: Assume this is true for all points $n_0 \leq k < n$. (We choose $k = \frac{n}{8}$), i.e. $T(\frac{n}{8}) \leq c\frac{n}{8} \log \frac{n}{8}$.

$$\begin{aligned}
T(n) &= 8T(\frac{n}{8}) + n \\
&\leq 8(c\frac{n}{8} \log (\frac{n}{8})) + n && \text{(By IH)} \\
&= cn \log (\frac{n}{8}) + n \\
&= cn(\log n - \log 8) + n && (\log (\tfrac{x}{y}) = \log x - \log y) \\
&= cn \log n - 3cn + n \\
&\leq cn \log n && \text{As long as } c \geq \tfrac{1}{3}
\end{aligned}$$

All is left for you now is to find the values of the constants c and n_0. For instance, $n_0 = 8$ and $c = \frac{2}{3}$ is a valid witness.

- **Question 6: [15 Points]** Give a tight asymptotic bound (i.e. $\Theta(\cdot)$) for the recurrence $T(n) = 27T(\frac{n}{3}) + n^2$. Assume $T(1) = \Theta(1)$. You are free to use any of the methods we covered. Mention what method you used. Show all your working.

  Answer:
  The question here did not specify a particular method that must be used so you are free to use any of the methods we covered. IMHO, the easiest one to use is the Master Method if it is applicable to the given recurrence relation, which is the case here. Always state what method you are using. I am using the master method here. Also, always show all your workings so you might still get some marks if things go wrong.

  $a = 27$, $b = 3$, $f(n) = n^2$ and $\log_b a = \log_3 27 = 3$. We have $n^2 = O(n^{3-\epsilon})$ for any $\epsilon \in (0, 1]$. i.e. $f(n)$ grows slower than $n^{\log_b a}$ so this is clearly case 1 of the Master Theorem.

  The answer here is T(n)=$\Theta(n^{\log_b a}) = \Theta(n^3)$.