Tutorial

Question 1

Review the definitions of the basic notations such as Θ, O, Ω, o, and ω.

O-notation

For a given function $g(n)$, denoted by $O(g(n))$ the set of functions.

$O(g(n)) = \{f(n)$: there exist positive constants c and n_0 such that $0 \leq f(n) \leq cg(n)$ for all $n \geq n_0\}$.

Ω-notation

For a given function $g(n)$, denoted by $\Omega(g(n))$ the set of functions.

$\Omega(g(n)) = \{f(n)$: there exist positive constants c and n_0 such that $0 \leq cg(n) \leq f(n)$ for all $n \geq n_0\}$.

Θ-notation

For a given function $g(n)$, denoted by $\Theta(g(n))$ the set of functions.

$\Theta(g(n)) = \{f(n)$: there exist positive constants c_1, c_2, and n_0 such that $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$ for all $n \geq n_0\}$.

o-notation

For a given function, $g(n)$, we denote by $o(g(n))$ the set of functions.

$o(g(n)) = \{f(n)$: for *any* positive constant $c > 0$, there exists a constant $n_0 > 0$ such that $0 \leq f(n) < cg(n)$ for all $n \geq n_0\}$.

The implication of this, is that $f(n)$ becomes insignificant relative to $g(n)$ as n approaches infinity:

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$$

ω-notation

For a given function, $g(n)$, we denote by $\omega(g(n))$ the set of functions.

$\omega(g(n)) = \{f(n)$: for *any* positive constant $c > 0$, there exists a constant $n_0 > 0$ such that $0 \leq cg(n) < f(n)$ for all $n \geq n_0\}$.

The implication of this is that $g(n)$ becomes insignificant relative to $f(n)$ as n approaches infinity:

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$$

Question 2

What is the difference between O and o as well as Ω and ω?

The difference between O and o is that O represents an upper bound of the solution for a problem, which may also be the tight bound of solving that problem, while o clearly indicates that it is an upper bound but definitely not the tight bound.

For example, $2n = o(n^2)$ but $2n^2 \neq o(n^2)$.

For the Omegas...

For example $n^2/2 = \omega(n)$ but $n^2/2 \neq \omega(n^2)$.

Question 3

Rank the following functions in the order of growth: $n^2 \log n$, $n^{3/2} \log^4 n$, $3^{0.7n}$, $\log^* n \log^2 n$, $n^{1/\log n}$, $\sqrt{n^{1/3} \log n}$, $n^{\log \log n}$, $\sqrt{n}$

First, work out that the following is actually a constant...

$$\begin{aligned} n^{\frac{1}{\log_2 n}} &= n^{\frac{\lg 2}{\lg n}} \\ &= n^{\log_n 2} \\ &= 2^{\log_n n} \\ &= 2^1 \\ &= 2 \end{aligned}$$

(5) $n^{1/\log n}$ $\rightarrow$ 2
(4) $\log^* n \log^2 n$ $\rightarrow$ $\log n$
(6) $\sqrt{n^{1/3} \log n}$ $\rightarrow$ $n^{1/6}$
(8) $\sqrt{n}$ $\rightarrow$ $n^{1/2}$
(2) $n^{3/2} \log^4 n$ $\rightarrow$ $n^{3/2}$
(1) $n^2 \log n$ $\rightarrow$ n^2
(7) $n^{\log \log n}$ $\rightarrow$ $n^{\log \log n}$
(3) $3^{0.7n}$ $\rightarrow$ 3^n

Note that "Log star of n" is iterated logarithm.

$\log^* n = \min\{i \geq 0 : \log^i n \leq 1\}$

Question 4

Use the divide-and-conquer paradigm to explain the basic steps of the quicksort algorithm.

The divide-and-conquer strategy divides a problem A into a number of subproblems of roughly equal sizes. Then each subproblem is solved recursively. The base case being a subproblem that is small enough to solve without further dividing. Next, the solutions of the subproblems are combined to obtain the solution for the original problem.

For the case of quicksort:

1. (DIVIDE) Pick an element $x = a_i$ to be the pivot element. Usually the first element a_1 is picked.
2. (DIVIDE) Partition the sequence into two disjoint subsequences:
   $S_1 = \{a_i : a_i \leq x\}$, and
   $S_2 = \{a_j : a_j > x\}$.
3. (CONQUER) Sort S_1 and S_2 recursively using quicksort.
4. (MERGE) Merge the sorted S_1, $\{x\}$, and S_2. The original sequence is sorted.

Question 5

Review the four techniques for bounding the summations. Use one of the introduced techniques to bound $\sum_{k=1}^{\infty} \frac{k}{5^k}$.

The four techniques for bounding summations are as follows.

1. **Mathematical induction**
   What is mathematical induction? For example, binding the series: $1 + 2 + 3 + \ldots + n$

$$P(n) \text{ says: } 1 + 2 + 3 + \ldots + n = \frac{n(n+1)}{2}$$

   If $P(n)$ is true, is $P(n+1)$ true?

$$\begin{aligned} P(n+1) \text{ says: } 1 + 2 + 3 + \ldots + n + (n+1) &= \frac{n(n+1)}{2} + (n+1) \\ &= \frac{(n+1)(n+2)}{2} \\ &= \frac{(n+1)((n+1)+1)}{2} \end{aligned}$$

Therefore $P(n) \Rightarrow P(n+1)$.

Now, show our base case. $P(1)$ is true because $1 = \frac{1(1+1)}{2} = 1$. And since $P(1)$ is true, $P(2)$ must be true, so $P(3)$ is true, and so on...

2. **Splitting summations**

   For example, finding the lower bound.

   We can change k to n and find the upper bound:

$$\begin{aligned}\sum_{k=1}^{n} k &\leq \sum_{k=1}^{n} n \\ &= n^2 \\ &= O(n^2)\end{aligned}$$

   To find the lower bound we split the summation:

$$\begin{aligned}\sum_{k=1}^{n} k &= \sum_{k=1}^{n/2} k + \sum_{k=n/2+1}^{n} k \\ &\geq \sum_{k=1}^{n/2} 0 + \sum_{k=n/2+1}^{n} n/2 \\ &= \frac{n^2}{4} \\ &= \Omega(n^2)\end{aligned}$$

3. **Approximation by integrals**

   For example, if we have $\sum_{x=p}^{q} f(x)$, then the sum really is the area under the curve $f(x)$ for $p \leq x \leq q$. We can use integration to find the area under a curve. Thus the summation is approximately equal to $\int_p^q f(x)$.

4. **Bounding terms**

   We now apply the bounding terms to show the upper bound of

$$\sum_{k=1}^{\infty} \frac{k}{5^k}$$

   Let $a_k = \frac{k}{5^k}$

   Then...

$$\frac{a_{k+1}}{a_k} = \frac{k+1}{5^{k+1}} \times \frac{5^k}{k} = \frac{k+1}{5k}$$

This is maximum when $k = 1$, so...

$$\begin{aligned} r &\leq \frac{1+1}{5} \\ r &\leq \frac{2}{5} \end{aligned}$$

Given that the first term $a_1 = 1/5$, $\sum_{k=1}^{\infty} \frac{k}{5^k}$ can be bounded by the sum to infinity of the geometric series.

$$\begin{aligned} \sum_{k=1}^{\infty} \frac{k}{5^k} &\leq \frac{1}{5} \times \frac{1}{1-\frac{2}{5}} \\ &= \frac{1}{5} \times \frac{1}{\frac{3}{5}} \\ &= \frac{1}{3} \end{aligned}$$

Question 6

Review the two approaches for solving recurrences, and apply them to solve the following recurrences.

- **Substitution method:** guess a bound and then use mathematical induction to prove that the guess is correct.
- **Iteration method:** convert the recurrence into a summation and then use the techniques for bounding summations to solve the recurrence.

Question 6(a)

$T(n) = T(n/3) + n^{4/3}$

Using the iterative method...

$$\begin{aligned} T(n) &= n^{4/3} + T(n/3) \\ &= n^{4/3} + (\frac{n}{3})^{4/3} + T(\frac{n}{3^2}) \\ &= n^{4/3} + (\frac{n}{3})^{4/3} + (\frac{n}{3^2})^{4/3} + T(\frac{n}{3^3}) \\ &= n^{4/3} + (\frac{n}{3})^{4/3} + (\frac{n}{3^2})^{4/3} + \ldots + (\frac{n}{3^k})^{4/3} \end{aligned}$$

$$\begin{aligned} T(n) &= n^{4/3} \sum_{i=0}^{k} \left(\frac{1}{3^i}\right)^{4/3} \\ &= n^{4/3} \sum_{i=0}^{k} \left(\frac{1}{3^{4/3}}\right)^{i} \\ &< n^{4/3} \sum_{i=0}^{\infty} \left(\frac{1}{3^{4/3}}\right)^{i} \\ &< n^{4/3} \frac{1}{1-\frac{1}{3^{4/3}}} \end{aligned}$$

Let constant $c = \frac{1}{1-\frac{1}{3^{4/3}}}$

$$\begin{aligned} T(n) &< cn^{4/3} \\ &= O(n^{4/3}) \end{aligned}$$

If you want to find k...

$$\begin{aligned} \frac{n}{3^k} &= 1 \\ k &= \log_3 n \end{aligned}$$

Question 6(b)

$T(n) \leq T(n/4) + T(n/5 + 57) + \log_2 n$

Using the substitution method, first...

$$\begin{aligned} n/4 &\geq n/5 + 57 \\ n &\geq 1140 \end{aligned}$$

Since $n/4 \geq n/5+57$ when $n \geq 1140$, $T(n/4)$ will dominate $T(n/5+57)$ when $n \geq 1140$. So we simplify the recurrence into:

$$T(n) \leq 2T(n/4) + \log_2 n$$

GUESS 1: Now we guess that the solution is:

$$T(n) \leq c\sqrt{n}$$

Assuming that this holds for $T(n/4)$:

$$\begin{aligned} T(n/4) &\leq c\sqrt{n/4} \\ &= \frac{1}{2}c\sqrt{n} \end{aligned}$$

Substitute this into our original $T(n)$:

$$\begin{aligned} T(n) &\leq 2\left(\frac{1}{2}c\sqrt{n}\right) + \log_2 n \\ &= c\sqrt{n} + \log_2 n \end{aligned}$$

This doesn't agree with our guess.
GUESS 2: Revise our guess to $T(n) \leq c\sqrt{n} + b\log_2 n$
Assume $T(n/4) \leq c\sqrt{n/4} + b\log_2(n/4)$
Substitute:

$$\begin{aligned} T(n) &\leq 2\left(c\sqrt{n/4} + b\log_2(n/4)\right) + \log_2 n \\ &= c\sqrt{n} + 2b\log_2 n - 2b\log_2 4 + \log_2 n \\ &= c\sqrt{n} + (2b+1)\log_2 n - 4b \end{aligned}$$

Still doesn't agree with our guess.
GUESS 3: Revise our guess to $T(n) \leq c\sqrt{n} + b\log_2 n + a$
Assume $T(n/4) \leq c\sqrt{n/4} + b\log_2(n/4) + a$
Substitute:

$$\begin{aligned} T(n) &\leq 2\left(c\sqrt{n/4} + b\log_2(n/4) + a\right) + \log_2 n \\ &= c\sqrt{n} + 2b\log_2 n - 2b\log_2 4 + \log_2 n + 2a \\ &= c\sqrt{n} + (2b+1)\log_2 n + 2a - 4b \end{aligned}$$

Since this is the same form as our guess, we solve for b and a.

$$\begin{aligned} 2b+1 &= b \\ b &= -1 \end{aligned}$$

$$\begin{aligned} 2a - 4b &= a \\ a &= 4b \\ a &= -4 \end{aligned}$$

Thus, the inequality holds when $b = -1$ and $a = -4$.
For all $n < 1140$, let the $T(n)$ be bounded by a constant R.
Solve for c when $n = 4096$:

$$\begin{aligned} T(4096) &\leq c\sqrt{4096} - \log_2 4096 - 4 \\ 2T(1024) + log_2 4096 &\leq c\sqrt{4096} - \log_2 4096 - 4 \\ 2R + 12 &\leq 64c - 12 - 4 \\ c &\geq \frac{2R + 28}{64} \end{aligned}$$

Thus, $0 \leq T(n) \leq c\sqrt{n} - \log_2 n - 4$ when $c \geq \frac{2R+28}{64}$ for all $n \geq 1140$
$T(n) = O(\sqrt{n})$

Question 6(c) iterative

$T(n) = T(\lfloor\sqrt{n}\rfloor) + 3n$

Using the iterative method...

$$\begin{aligned} T(n) &= 3n + T(\lfloor\sqrt{n}\rfloor) \\ &= 3n + 3n^{1/2} + T(\lfloor n^{1/4}\rfloor) \\ &= 3n + 3n^{(\frac{1}{2})} + 3n^{(\frac{1}{2})^2} + \ldots + 3n^{(\frac{1}{2})^k} \end{aligned}$$

$$\begin{aligned} k &= \text{no. of times we can square root } n \text{ until it is 2} \\ n^{(\frac{1}{2})^k} &= 2 \\ \left(\frac{1}{2}\right)^k \lg n &= \lg 2 \\ \left(\frac{1}{2}\right)^k &= \frac{\lg 2}{\lg n} \\ 2^k &= \lg n \\ k \lg 2 &= \lg \lg n \\ k &= \lg \lg n \end{aligned}$$

$$\begin{aligned} T(n) &\leq 3n \times (k+1) \\ &\leq 3n \times (\lg \lg n + 1) \\ &\leq 3n \lg \lg n + 3n \\ &= O(n \lg \lg n) \end{aligned}$$

Question 6(c) changing variables

$T(n) = T(\lfloor\sqrt{n}\rfloor) + 3n$

An alternative method is to use "changing variables" together with the iterative method.

Let $n = 2^m$.

Substituting this into the recurrence, we get:

$$\begin{aligned} T(2^m) &= T(2^{m/2}) + 3 \times 2^m \\ &\quad \text{Note that square root is essentially halving the power.} \end{aligned}$$

Rename $S(m) = T(2^m)$:

$$S(m) = S(m/2) + 3 \times 2^m$$

This is possible because now we have changed the "problem size" variable to m. Note that work done at each level of the recursion is still $3 \times 2^m = 3n$. So using the iterative method to solve $S(m)\ldots$

$$\begin{aligned} S(m) &= S(m/2) + 3 \times 2^m \\ &= 3 \times 2^m + 3 \times 2^{m/2} + S(m/2^2) \\ &= 3 \times 2^m + 3 \times 2^{m/2} + 3 \times 2^{m/2^2} + \ldots + 3 \times 2^{m/2^k} \\ &= 3 \sum_{i=0}^{k} 2^{m/2^i} \end{aligned}$$

Find k:

$$\begin{aligned} 2^{m/2^k} &= 2 \\ m/2^k &= 1 \\ k &= \log_2 m \end{aligned}$$

Since the biggest element in the summation is when $i = 0$, we can bind the summation by replacing $2^{m/2^i}$ with 2^m.

$$\begin{aligned} S(m) &\leq 3 \times 2^m \log_2 m \\ &= O(2^m \log_2 m) \end{aligned}$$

$$
\begin{aligned}
T(n) &= T(2^m) \\
&= S(m) \\
&= O(2^m \log_2 m) \\
&= O(n \log_2 \log_2 n)
\end{aligned}
$$