COMP3506/7505 Week 3 Tutorial

Algorithm Analysis II

Tutorial Questions

Questions marked with a star ($\star$) are strongly recommended. Try to complete these before your tutorial. These questions will be covered during the tutorial class. While your tutor will attempt to cover all these questions, not all may be covered depending on time constraints. The questions without a star will *not* be explicitly covered during the tutorial class but are provided as additional resources for extension and further practice. You are welcome to ask your tutor about these problems if you wish. Otherwise, you are also welcome to post any questions you have on Ed Discussion.

1. ($\star$) Order the following functions by asymptotic growth rate:

$4n \log_4 n + 2n$	2^{10}	$2^{\log_2 n}$	$n^2 + 10n$	$n^2 \log_2(n)$
$100 \log_2 n$	$4n$	2^n	n^3	$3 \log_3(\log_3 n)$

2. ($\star$) Analyse the running time of the following function for arbitrary values of the input parameter n.

```
Algorithm printIntegers(n)
    Input: an integer, n, representing the number of integers to print
    Output: printed integers from 0 to n − 1 (no return value)

    i ← 0
    while i < n do
        println i
        i ← i + 1
```

3. ($\star$) Analyse the running time of the following function for arbitrary values of the input length n.

```
Algorithm binarySum(A, i, n)
    Input: an array of integers A and integers i and n.
    Output: the sum of the n values in A starting at index i.

    if n = 1 then
        return A[i]
    return binarySum(A, i, ⌈n/2⌉) + binarySum(A, i + ⌈n/2⌉, ⌊n/2⌋)
```

4. ($\star$) Suppose you have a sorted sequence S of infinite length (somehow) and without duplicates. You can query any value $S[i]$ in constant time, but the sequence has no end.

   How can we search for a value x in S? Extend binary search to solve this efficiently.

5. ($\star$) State the big-O complexity of the following recursive functions with a short explanation. For each, give a recursive algorithm which has that running time.

   Hint: It may help to think of the algorithm first. Look at how the input size decreases, as well as how much work is done at each step. For this question, you do not need to rigorously derive your answers; it is enough to draw a tree and state the big-O bound.

   (a) $T(n) = \begin{cases} T(n-1) + O(1), & n > 1, \\ O(1), & n = 1. \end{cases}$

(b) $T(n) = \begin{cases} 2\,T(n/2) + O(n), & n > 1, \\ O(1), & n = 1. \end{cases}$

(c) $T(n) = \begin{cases} T(n-1) + O(n), & n > 1, \\ O(1), & n = 1. \end{cases}$

(d) $T(n) = \begin{cases} 2\,T(n-1) + O(1), & n > 1, \\ O(1), & n = 1. \end{cases}$ $(*)$

Extra Fun (Do these questions after you complete the ones above)

The following questions will *not* be explicitly covered during the tutorial class but are provided as additional resources for extension and further practice. You are welcome to ask your tutor about these problems if you wish. Otherwise, you are also welcome to post any questions you have on Ed Discussion.

6. $(*)$ Suppose you have a sorted array A of n integers, possibly with duplicates. It is easy to find a particular value, but sometimes we want the number of times a value occurs in A.

   How would you adapt simple binary search to solve this task in $O(\log n)$ time?

   (Hint: Binary search tests for equality, i.e. $A[i] = x$. What if we changed this to something else?)

7. Show that the following recurrence is $O\big(n^{\log_2 3}\big)$ in the worst case:

$$T(n) = \begin{cases} 3\,T(n/2) + n, & \text{if } n > 1, \\ 1, & \text{if } n = 1. \end{cases}$$

   Try drawing a tree of the recursive calls. You can use the following mathematical facts:

$$\sum_{i=0}^{h} ar^i = \frac{a(r^{h+1} - 1)}{r - 1}, \qquad b^{\log_c a} = a^{\log_c b}.$$

8. For each of the following problems, briefly explain how you would find the desired numbers within the given amount of time. You can reference common sorting algorithms if needed.

   (a) Let A be a *sorted* array of n integers.

      i. Find the pair $x, y \in A$ which maximises $|x - y|$ in $O(1)$ worst-case time.

      ii. Find the pair $x, y \in A$ which minimises $|x - y|$, for $x \neq y$, in $O(n)$ worst-case time.

   (b) Let B be an *unsorted* array of n integers.

      i. Find the pair $x, y \in B$ which maximises $|x - y|$ in $O(n)$ worst-case time.

      ii. Find the pair $x, y \in B$ which minimises $|x - y|$, for $x \neq y$, in $O(n \log n)$ worst-case time.

   (Question from Skiena's *Algorithm Design Manual*.)

9. Suppose you are given two arrays of integers A and B (both length n) and an integer x. Describe a $O(n \log n)$ algorithm for finding whether there exist indices i, j such that $A[i] + B[j] = x$. (Hint: First, try for $O(n^2)$ then consider which algorithms are usually $O(n \log n)$.)

   (Question from Skiena's *Algorithm Design Manual*.)

10. (Possibly difficult) You have an array A of n integers and an integer x. Design a $O(n^{k-1} \log n)$ algorithm which determines if k of the integers in A sum to x.

    (Question from Skiena's *Algorithm Design Manual*.)

11. (Possibly difficult) In lectures and tutorials, we have only formally defined Big-O notation for functions of a single variable. However, it is common in algorithm analysis to provide Big-O bounds that contain multiple variables. For the following questions, you will need to research (e.g. via the internet, textbooks, or research papers).

    (a) For a function f of k variables, $f(n_1, ..., n_k)$, provide a formal definition of a function f being $f(n_1, ..., n_k) \in O(g(n_1, ..., n_k))$, where g is also a function of k variables.

    (b) Similarly provide a definition for $f(n_1, ..., n_k) \in \Omega(g(n_1, ..., n_k))$ and $f(n_1, ..., n_k) \in \Theta(g(n_1, ..., n_k))$

    (c) Find a Big-O and Big-Ω bound for the following function, and then prove the correctness of it using the definitions you have provided in the previous questions.

$$f(n, m, k) = \begin{cases} mnk^2 & m \geq 10 \\ m \log n + k \log m & \text{otherwise} \end{cases}$$