COMP3121/9101
Algorithm Design

Tutorial 1
Data Structures and Algorithms Revision

Before the Tutorial

Before coming to the tutorial, try and answer these discussion questions yourself to get a firm understanding of how you're pacing in the course. No solutions to these discussion points will be officially released, but feel free to discuss your thoughts on the forum. Your tutor will give you some comments on your understanding.

- Outline the differences between linear search and binary search.
  - It would be helpful to discuss what arrays each searching method can be applied on. What is the time complexity for each strategy?
- Briefly outline the following sorting methods and explain their asymptotic complexities for an unsorted array on n integers:
  - Bubble sort.
  - Insertion sort.
  - Selection sort.
  - Merge sort.
- Let $f(n)$ and $g(n)$ be two positive functions in terms of the input n. Briefly outline what the following asymptotic relations mean.
  - $f(n) = O(g(n))$.
  - $f(n) = \Omega(g(n))$.
  - $f(n) = \Theta(g(n))$.
- Read through the problem prompts and think about how you would approach the problems.

Tutorial Problems

The problems in this tutorial can be done purely with techniques found in COMP2521 or COMP9024.

Problem 1. Given two arrays of n distinct integers, design an $O(n \log n)$ algorithm to determine if there is a common element among the two arrays.

For example, $A = [1, 8, 9, 2]$ and $B = [2, 7, 4, 5]$ have the element 2 in common.

Then, using a known data structure, design an algorithm that solves the same problem that runs in $O(n)$ *expected time*.

- Why doesn't the second method run in $O(n)$ time in the *worst case*?

Problem 2. In this problem, we solve a more advanced application of binary search.

A conveyor belt contains n packages that need to be delivered to AlgoTown within K days. Assume that $K \geq 1$. Package i has an associated weight $w_i \leq M$, where w_i is an integer. Each day, the packages are loaded onto a truck in the order of their position on the belt. We may not load more than the capacity of the truck. We may also assume that the capacity of the truck is an integer.

(a) To get a better understanding of the description above, consider the following example.

There are $n = 10$ items with the following weights: $w = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]$ and $K = 5$.

- Determine if it is possible to load all packages within 5 days if the truck's capacity is 12.
- Determine if it is possible to load all packages within 5 days if the truck's capacity is 17.

(b) Explain why it is always possible to load all packages within K days if the capacity of the truck is equal to the sum of the weights of the packages, regardless of the value of K.

(c) Given the capacity C of the truck, the weights of the packages and the number of days K, design an $O(n)$ algorithm that checks whether it is possible to deliver all packages within K days.

- In this tutorial, you may informally prove the correctness of the algorithm.
- Ensure that you argue that your algorithm runs within the specified time complexity.

(d) Suppose now that you get to choose the capacity of the truck. To save money, you would like to choose the smallest possible truck that allows you to deliver all packages within K days. Explain why binary search is a feasible strategy to solve the problem.

- Is there perhaps a monotonic behaviour hidden in the problem?

(e) Hence, design an $O(n \log(nM))$ algorithm that solves the problem in part (d).

- If time permits, use the algorithm proposed to find the smallest possible truck that solves the example case in part (a). Determine, without further computation, whether it is possible to load all packages within 5 days if the truck's capacity is 11 or 18.

Problem 3. Decide whether the following asymptotic relations are true or false, and explain why.

(a) $n^2 \log n = O(n^3)$.

(b) $n^2 \log n = \Omega(n^3)$.

(c) $n = \Theta(100n + 100)$.

After the Tutorial

After your allocated tutorial (or after having done the tutorial problems), review the discussion points. Reflect on how your understanding has changed (if at all).

- In your own time, try and attempt some of the practice problems marked [K] for further practice. Attempt the [H] problems once you're comfortable with the [K] problems. All practice problems will contain fully-written solutions.

- If time permits, try and implement one of the algorithms from the tutorial in your preferred language. How would you translate high-level algorithm design to an implementation setting?