


COMP3121/9101

Algorithm Design and Analysis


Recitation 2

Foundations 2, Divide & Conquer 1

Announcements

- Carefully, go through the recitation problems and solutions.

1 Definitions

- The *time complexity* of an algorithm is a description of how its run-time grows with input size. When analysing the performance of algorithms, we are often interested in the asymptotic time complexity in response to different kinds of inputs.
  - The *worst-case* time complexity refers to the asymptotic behaviour of the algorithm when given the slowest possible inputs for any given size.
  - The *average* or *expected* time complexity refers to the asymptotic behaviour of the algorithm when given a random input of a given size.

  Time complexity is expressed using *big O notation*.
  - *Big-O* is an upper bound: if $f(n) = O(g(n))$, then $f(n)$ does not grow faster than $g(n)$ asymptotically.
  - *Big-*Ω is a lower bound: if $f(n) = \Omega(g(n))$, then $f(n)$ does not grow slower than $g(n)$ asymptotically.
  - *Big-*Θ is both an upper and lower bound: if $f(n) = \Theta(g(n))$, then $f(n)$ grows the same as $g(n)$ asymptotically (up to a constant factor).

  This notation is simply a tool to describe how a function grows with large inputs. These bounds should not be confused with worst-case, best-case and average case performance!

  To classify time complexity of functions, we categorise by the dominant term, using the following heuristic.

$$\text{constants} \ll \text{logarithms} \ll \text{polynomials} \ll \text{exponentials} \ll \text{factorials} \ll \dots$$

- There are a few data structures that you should already know about from prerequisite courses.
  - A *Binary Heap* is a complete binary tree where every parent is $\geq$ its children (max heap), or $\leq$ its children (min heap). Heaps can be used to implement priority queues, and support $O(n)$ construction, $O(1)$ find max, $O(\log n)$ find min, and $O(\log n)$ insertion.
  - A (self-balancing) *Binary Search Tree* is a binary tree where each node's key compares greater than all keys in its left subtree, and less than all keys in its right subtree. Self-balancing binary search trees support $O(\log n)$ search, $O(\log n)$ insert, and $O(\log n)$ delete.
  - A *Hash table* is used to store *values* indexed by *keys* through the use of a hash function. A hash function maps keys to indices in a fixed size table. The **average** time complexity for searching, updating, inserting, and deleting in a hash table is $O(1)$, but the **worst case** time complexity for all operations is $O(n)$.
- The *divide and conquer* algorithm paradigm solves problems through three key steps:
  - *Divide* the problem up into two or more equal, non-overlapping subproblems.
  - *Conquer* by recursively solving the subproblems, with a trivial base case.
  - *Combine* the subproblem solutions to solve the original instance of the problem.

  You have already seen some famous divide and conquer algorithms in prerequisite material - binary search, quicksort, and merge sort. For merge sort, we
  - *divide* into two equal subproblems (sort each half of the array separately)
  - *conquer* the subproblems recursively, with a base case of arrays of size 1.
  - *combine* the two sorted halves of the array with the $O(n)$ merge operation.

2 Data Structures

You are given two arrays $A[1..n]$ and $B[1..n]$, each containing n positive integers.

(a) Design an algorithm that runs in *worst-case* $O(n \log n)$ time to determine whether there exists some integer x that is in both A and B.

(b) Design an algorithm that runs in *average-case* $O(n)$ time to determine whether there exists some integer x that is in both A and B.

3 Binary Search

Info: Binary search can be used in many more ways than simply finding a value in sorted array. In this problem, we look at a more advanced application.

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 a positive 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) 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 13 or 16.

4 Divide and Conquer

You are given an array $A[1..n]$ of n positive integers, where $A[i]$ represents the value of a stock on day i. Your goal is to buy the stock on a particular day and then sell the stock at a later day to maximise your profit.

Note. *You may assume that n is a power of two.*

(a) Firstly, suppose that you want to buy the stock on the first $n/2$ days and then sell it on the last $n/2$ days. Describe an $O(n)$-time algorithm to find the best pair of days to buy and sell the stock.

(b) Describe an $O(n \log n)$-time algorithm to find the best pair of days to buy and sell the stock.