

## COMP 3711

### 2025 Fall Semester - Tutorial 5

**Question 1:** In class we learned the *Randomized Selection* algorithm and used a geometric series analysis approach to show that it runs in  $O(n)$  expected time. In this tutorial you will re-derive the  $O(n)$  time in a different way, using the *Indicator Random Variable* method used to analyze Quicksort. Recall the *Randomized Selection* algorithm to find the  **$k$ -th smallest element**. Pick a random pivot, divide the array into 3 parts:

left subarray, pivot item, right subarray.

We then either stop immediately or recursively solve the problem in the left **OR** the right part of the array.

- As in quicksort, denote the elements in **sorted order** by  $z_1, \dots, z_n$  (so we are searching for  $z_k$ )
- We use the same random model for choosing the pivot as for quicksort.

I) Define:

$$X_{ij} = \begin{cases} 1 & \text{if } z_i \text{ and } z_j \text{ are compared by the algorithm} \\ 0 & \text{otherwise} \end{cases}$$

Prove the following three facts

- (a)  $i \leq k \leq j : \Pr[X_{ij} = 1] = \frac{2}{j-i+1}$
- (b)  $i < j < k : \Pr[X_{ij} = 1] = \frac{2}{k-i+1}$
- (c)  $k < i < j : \Pr[X_{ij} = 1] = \frac{2}{j-k+1}$

II) By the indicator random variable technique, the expected total number of comparisons is

$$\sum_{j < j} \mathbb{E}[X_{ij}] = \sum_{i < j} \Pr[X_{ij} = 1]$$

Use this to show that  $\sum_{i < j} \mathbb{E}[X_{ij}] = O(n)$ .

**Question 2:** The analysis of the expected running time of randomized quicksort in the lecture notes assumed that all element values are distinct. In this problem, we examine what happens when they are not.

- (a) Suppose that all element values are equal. What would be randomized quicksort's running time?
- (b) The PARTITION procedure taught returns an index  $q$  such that each element of  $A[p, \dots, q-1]$  is less than or equal to  $A[q]$  and each element of  $A[q+1, \dots, r]$  is greater than  $A[q]$ . Modify PARTITION to produce a new procedure PARTITION'(A, p, r), which permutes the elements of  $A[p, \dots, r]$  and returns two indices  $q, t$ , where  $p \leq q \leq t \leq r$ , such that:

- \* all elements of  $A[q, \dots, t]$  are equal,
- \* each element of  $A[p, \dots, q-1]$  is less than  $A[q]$ , and
- \* each element of  $A[t+1, \dots, r]$  is greater than  $A[q]$

Like PARTITION, your PARTITION' procedure should take  $\Theta(r - p)$  time.

- (c) Modify the QUICKSORT procedure to produce QUICKSORT'(A, p, r) that calls PARTITION' but then only recurses on  $A[p, q - 1]$  and  $A[t + 1, r]$
- (d) [Advanced Topic: Not needed for class] Our analysis of QUICKSORT in class assumed that all elements were distinct. Using QUICKSORT', how would you adjust the analysis (binary tree plus indicator random variables) in the lecture notes to avoid the assumption that all elements are distinct?

**Question 3:** In class, we learned how to maintain a min-heap implicitly in an array.

Given that  $A[i, \dots, (j - 1)]$  represents an implicit min-heap, we saw how to add  $A[j]$  to the min-heap, in  $O(\log j)$  time.

This led to an  $O(n \log n)$  algorithm for constructing a min-heap from array  $A[1, \dots, n]$ .

**For this problem show how to construct a min-heap from an array  $A[1, \dots, n]$  in  $O(n)$  time.**

It might help to visualize the min-heap as a binary tree and not an array.

For simplification, you may assume that  $n = 2^{k+1} - 1$  for some  $k$ , i.e., the tree is complete.

*Hint: Consider "heapifying" the nodes processing them from bottom to top.*