Exercise Sheet 5
COMS10017 Algorithms 2022/2023

Reminder: $\log n$ denotes the binary logarithm, i.e., $\log n = \log_2 n$.

1 Heapsort

Consider the following array A:

4	3	9	10	14	8	7	2	1	7

1. Interpret A as a binary tree as in the lecture (on heaps) and draw the tree.
2. Run Create-Heap() on the initial array. Give the sequence of node exchanges. Draw the resulting heap.
3. What is the worst-case runtime of Create-Heap() and how is the runtime established?
4. Explain how Heapsort uses the heap for sorting. Explain why the algorithm has a worst-case runtime of $O(n \log n)$.

2 Heapsort: An Alternative to Create-Heap()

Let A be an integer array of length n. Heapsort interprets the input array A as a binary tree, and the Create-Heap() function shuffles the elements of A such that a valid heap is obtained, i.e., the heap property is fulfilled at every node. In this exercise, we will analyse an alternative to the Create-Heap() function that uses the auxiliary function Heapify-Up():

Heapify-Up(c) is called on a node c of the tree. It operates as follows. If the value stored at c is smaller or equal to the value stored at c's parent then do nothing. Otherwise, the value stored at c is larger than the value stored at c's parent. In this case, exchange c and c's parent. Heapify-Up() is then called recursively on the new location of c.

Based on Heapify-Up(), we now consider the function Alt-Create-Heap():

Algorithm 1 Alt-Create-Heap()

Require: Array A of n integers

for $i = 1$ **to** $n - 1$ **do**
  Interpret the prefix array $A[0, \dots, i]$ as a binary tree as in the lectures
  Run Heapify-Up(c) on the node c associated with $A[i]$
end for

1. Consider the prefix $A[0, \dots, i]$. What is the runtime of Heapify-Up(c) when called on the node c associated with $A[i]$?

2. What is the runtime of Alt-Create-Heap()?

3. Prove the following loop-invariant:

At the beginning of iteration i, the binary tree associated with the prefix $A[0, \ldots, i-1]$ constitutes a heap.

   Conclude that Alt-Create-Heap() indeed creates a valid heap.

3 Mergesort

Illustrate how the Mergesort algorithm sorts the following array using a recursion tree:

$$11 \quad 7 \quad 2 \quad 5 \quad 9 \quad 6 \quad 1$$

4 Circularly Shifted Arrays

Suppose you are given an array A of length n of **distinct** (all integers are different) sorted integers that has been circularly shifted by k positions to the right. For example, $[35, 42, 5, 15, 27, 29]$ is a sorted array that has been circularly shifted by $k = 2$ positions, while $[27, 29, 35, 42, 5, 15]$ has been shifted by $k = 4$ positions. Describe an $O(\log n)$ time algorithm that allows us to find the maximum element.

5 Optional and Difficult Questions

Exercises in this section are intentionally more difficult and are there to challenge yourself.

5.1 "Is this the simplest (and most surprising) sorting algorithm ever?", Stanley P. Y. Fung

Please read and appreciate chapters 1 and 2 of the following paper, published in 2021:

`https://arxiv.org/pdf/2110.01111.pdf`