Week 13 Additional Exercises on Heaps

Objectives of this Tutorial

- To understand heaps, heapsort and Queue ADT.

Exercise 1

Insert the following elements into an empty (max) heap, in the order shown. After each insertion, show the structure of the heap using a binary tree.

```
15 12 19 10 18 20 22 40
```

Exercise 2

Respond to the following in `Ex2.txt`:

1. Consider the array `10, 20, 30, 40, 50, 60, 70, 80`, where element `10` is at index 1, `20` at index 2, etc. Is this array a heap? Explain.

2. Consider now the array `80, 70, 60, 50, 40, 30, 20, 10`, where element `80` is at index 1, `70` at index 2, etc. Is this array a heap? Explain.

Exercise 3

Is heapsort stable? In `Ex3.txt`, explain and justify your answer. If you believe it is not stable, give an example as well.

Exercise 4

1. If you have an array-based (min-)heap H containing n items, the minimum element will be at index 1 (the left-most element). What are the possible indices that could contain the *maximum* element of H?
2. Describe how would you find the *third* smallest value (assuming you have access to the underlying array). What is the complexity of your method?

Exercise 5

1. Explain how you could use a heap to implement a **Queue** ADT. What is the complexity of each of the **Queue** operations?
2. Try the same for a **Stack** ADT. What would you need to do differently?

Exercise 6

Recall the implementation of `heapify`:

```
def heapify(list_of_values)
    # Initialise s as a heap with s.array = [None] + list_of_values + [None ...]
    for x in range(len(list_of_values)//2, 0, -1):
        s.sink(x)
    return s
```

1: Why does this work? Why are we guaranteed the heap property?

Hint:

▶ Expand

2: What is the worst / best case complexity of `heapify` ?

3: Suppose, we instead implemented `heapify` as

```
def heapify(list_of_values)
    # Initialise s as a heap with s.array = [None] + list_of_values + [None ...]
    for x in range(1, len(list_of_values)+1):
        s.rise(x)
    return s
```

First, confirm that this other method of heapifying works. Second, what is the worst / best case complexity of `heapify` now?