Ashutosh Rai
CS 335

# Minilab 6: Heapsort Writeup

**Writeup #A**

Heapsort is a sorting algorithm that makes use of the heap data structure. On a higher level, heapsort has two main steps: first it heapifies the list of numbers to be sorted, then the numbers are pulled out (removed) from the heap one at a time, inserting them in the new sorted list. Heapsorts can have different designs, depending on the type of heaps used as well. Even though heapsort might seem inefficient at a quick glance due to the multiple operations/steps, it is a great sorting algorithm with a running time of O(n log n), which is as good as any sorting algorithm.

<u>Pseudocode for the Heapsort:</u>

```
# Assuming we have a working heap with the methods to find_min(),
# remove_min() and heapify(list) along with the other core methods

main_heap = heapify(original_list)
while (main_heap is not empty):
      current_min = main_heap.find_min()
      sorted_list.append(current_min)
      main_heap.remove_min()
```

**Writeup #B**

The running time of heapsort is O(n log n) as mentioned above, heapsort takes place in two main steps: heapifying the unsorted list, and continuous removal of the minimum from the heap and inserting it into the new sorted list. The heapifying step has a running time of O(n). The removal of the minimum from a heap takes O(log n) time, and as there are n numbers, the minimum needs to be removed n times, pushing the running time for the heapsort to be O(n log n).

**Writeup #C**

Position: Which heap is best depends on what you expect your data to look like.

Different types of heaps have different pros and cons. They have different advantages and disadvantages as a result of the differing running times for different operations, which makes them suitable for different purposes. For example, Binary heaps have a running time of O(1) for finding minimum, O(log n) for insertion and O (n) for merging while Binomial heaps have a running time of O(log n) for finding minimum and merging and O(1) amortised running time for insertion. This means that Binary heaps is the better choice if there aren't a whole lot of insertion and merging operations going on, but Binomial heaps is the better choice if there are. This directly translates to the heapsort as well, as heapsorts primarily uses heap data structure for implementation. Hence, depending on the data, and the nature of operations that might take place more often in the data, different heaps might be the best choice.