Heapsort

This document contains summary of some of the items discussed as part of the COMS 3110 Lectures. This document **does not replace the lecture materials**. This document may contain some topics that are not covered as part of the lecture; you will not be tested on those parts, they are made available to you for gaining further knowledge on topics/concepts that are related to class-lecture.

1 Heapsort

Given an array of integers, invoke makeHeap to arrange the elements in the form of a heap. Then extract elements in a for-loop. If the heap is a min-heap, then the elements will be extracted in ascending order; otherwise, the elements will be extracted in descending order. The runtime is proportional to the sum of cost of makeHeap and iterative extract operations. For n elements, makeHeap is $O(n)$. For n sequential extract operations, the cost is

$$\log(n) + \log(n-1) + \log(n-2) + \ldots + \log(1) \;=\; \log(n!)$$

By Stirling's approximation: $\log n! \approx n \log n - n$. Therefore, the runtime for sorting using heap (heapsort) is $O(n \log n)$.

2 A typical application

Finding top-k elements in order or the k-th top element are problems where arranging the elements in a heap can reduce the overall runtime from $O(n \log n)$ (runtime for sorting all n elements) to ($O(n \log k)$ (runtime for managing a heap of size k). The strategy is as follows. Pick the first k elements from the array and form a minheap ($O(k)$) (if the objective is to find top-k largest elements) and then insert a new element from the array ($O(\log k)$). Extract the minimum element ($O(\log k)$) and discard it. Continue with the process till there are no elements left. The heap should now contain the k largest elements. This is because everytime we are discarding an element (using extraction process) that is smaller than at least k other elements.