1. Implement Heap sort. (In addition to the slides, you could also read up pages 151--160 of CLRS for a description and analysis, but note that the arrays are indexed starting with 1).

You write three routines, Maxheapify which heapifies the input array, BuildMaxHeap, which builds the MaxHeap and finally HeapSort, which sorts the input array. Test your program thoroughly.

2. Compare the performance of Merge Sort, Quick Sort, and Heap Sort on random arrays and already sorted arrays. Report the results and findings especially their relative performances on different kinds of data.

3. Consider two *sorted* arrays *A* of size *m* and *B* of size *n.*

- (a) Design an efficient (*O(log m + log n))* Divide-and-Conquer algorithm to find the *k*'th element in the merged array. To be efficient, you should do this without actually merging the two arrays.
- (b) Prove that the time complexity is *O(log n + log m).*

1. Implementation of heap sort: 5 points

2. Evaluation of heap sort: 2 points

3. Algorithm design and analysis: 3 points

How to submit?

Label the source file 'hw3.py' and the report report3.pdf. Submit the report on Canvas and the source file at the teach site: https://teach.engr.oregonstate.edu/