Problem Set 4
IE 496 – Computational Methods in Optimization
Dr. Ralphs

1. Consider the following sorting algorithm to be implemented as part of the list class discussed in class.

```
void sortSubList(const int i, const int j)
{
   if (compare_(i, j)) exchange_(i, j);
   if (i + 1 >= j) return;
   k = floor((j - i + 1)/3);
   sortSubList(i, j - k);
   sortSubList(i + k, j);
   sortSubList(i, j - k);
}
```

   (a) Prove that this algorithm correctly sorts the array.

   (b) Analyze the running time of the algorithm.

2. Consider an $m \times n$ matrix such that the entries of each row are in sorted order from left to right and the entries in each column are in sorted order from top to bottom. We will call such a matrix a *sorted matrix*. Note that some of the entries in the matrix may be marked as empty, in which case we consider the value to be infinite.

   (a) Argue that all entries of a sorted matrix must be empty if the entry $(1, 1)$ is empty and that the matrix must be completely full if the entry (m, n) is filled.

   (b) Give an algorithm to extract the minimum element of the matrix in $O(m + n)$ time (the main challenge is in restoring the state of the matrix after deleting the minimum element). Your algorithm should use a recursive subroutine that solves an $m \times n$ instance by solving either an $m \times n - 1$ instance or a $m - 1 \times n$ instance. Give and solve a recurrence for the running time in terms of $p = m + n$ that yields the desired running time.

   (c) Give an $O(m + n)$ algorithm for inserting a new element into a non-full sorted matrix.

   (d) Explain how to sort n^2 numbers in $O(n^3)$ time using the above methods without calling any other sorting algorithm as a subroutine.

3. A d-heap is a heap in which non-leaf nodes in the tree can have up to d children instead of just 2.

   (a) Explain how to store a d-heap in an array.

   (b) What is the minimum height of a d heap of n elements in terms of n and d?

(c) Analyze the running times of the basic heap operations in terms of n and d.

4. Implement a priority queue using a binomial heap implementation in Python.