Problem Set 6
ISE 407 – Computational Methods in Optimization
Due: December 5, 2019
Dr. Ralphs

1. A *word ladder* is a sequence of English words connecting two given English words such that consecutive words in the sequence differ by only one letter. An example word ladder connecting FOOL and SAGE is

```
FOOL
POOL
POLL
POLE
PALE
SALE
SAGE
```

   There are many variations on the theme. One may be required to use a given number of words, to use only words of a given length (or not), to use the smallest numbers of words, etc.

   (a) Using GiMPy, implement an algorithm for finding the shortest possible word ladder connecting two given words of the same length. You might find

   `https://github.com/dwyl/english-words`

   useful.

   (b) By generating random pairs of words, determine (i) a lower bound on the maximum size and (2) an estimate of the average size of a word ladder for words of different lengths. Can you find a pair of words of the same length for which there doesn't exist a word ladder? If running time is too long, you may limit your dictionary in some reasonable way, but only if absolutely necessary!

   (c) Make a graph of the length of time it takes to find a shortest word ladder as a function of the length of the ladder.

   (d) Modify your algorithm to produce a word letter of a specified fixed length if one exists.

2. 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.

3. Argue that the average depth of the tree in binary search is $\Theta(\lg n)$. Hint: write a recursion for the average depth a tree of size n and solve it.

4. 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.

5. Suppose we have a graph with n vertices that is represented by a set of n bit vectors v^i, where $v^i_j = 1$ if there is an edge from vertex i to vertex j. Find an $O(n)$ algorithm to determine the vector p^1 for which $p^1_j = 1$ if there is a path from vertex 1 to vertex j. The operations you may use are bitwise logical operations on bit vectors, arithmetic operations on integers, instructions that set the value of a particular bit of a particular vector, and an instruction that assigns the value j to an integer variable a if the left-most "1" in a vector v is in position j or sets $a = 0$ if v consists of all zeros.