Math 560 Homework 9 Fall 2021

Due: Monday, December 6

1. **Two Stacks can make a Queue**

   A well implemented stack has two operations that it can support in $O(1)$ time: `Push` and `Pop`. It turns out that it is possible to implement an efficient queue using only two stacks. A queue has two supported operations:

   - `Enqueue`: where a value is pushed into the rear of the queue.
   - `Dequeue`: where a value is popped from the front of the queue.

   Your task will be to describe how a queue can be implemented using two stacks, and prove that your stack-based queue has the property that any sequence of n operations (selected from `Enqueue` and `Dequeue`) takes a total of $O(n)$ time resulting in amortized $O(1)$ time for each of these operations!

   (a) [2pts] Describe how a queue can be implemented using two stacks. (Hint: a more standard queue implementation typically tracks **two** values: front and rear. You have **two** stacks to use.)

   (b) [2pts] Using the aggregate method from class to prove that `Enqueue` and `Dequeue` each have $O(1)$ amortized runtimes.

   (c) [2pts] Using the accounting method from class to prove that `Enqueue` and `Dequeue` each have $O(1)$ amortized runtimes.

   (d) [2pts] Using the potential method from class to prove that `Enqueue` and `Dequeue` each have $O(1)$ amortized runtimes.

2. [6pts] **Fast Food Coloring**

   Say there is a group of towns that are all connected by roads, given as a graph where the weights of the edges are the distances between the towns.

   Suppose we are owners of k different fast food chains f_1, f_2, ..., f_k (for example, f_1 might be Pizza Shack, f_2 might be Burger Prince, f_3 might be Pusheye's Chicken, etc). We would like to place exactly one restaurant per town such that no two towns within a distance d of each other have the same restaurant, or determine if this is not possible.

   Meanwhile, the 3 Color Problem is defined by an undirected, unweighted graph and a set of 3 colors. The goal of the problem is to determine if each vertex in the graph can be colored with one of the 3 colors in such a way that no adjacent vertices have the same color.

   Prove that the Fast Food Problem is NP Complete. Do this by reducing the 3 Color Problem **to** the Fast Food Problem.

   In other words, suppose you are given an instance of the 3 Color Problem. Describe how you could, in polynomial time, create an instance of the Fast Food Problem (*specifying all necessary inputs to the problem*). Then describe how you could, in polynomial time, construct a solution for the 3 Color Problem *given an optimal solution* from your created Fast Food Problem. Finally, prove that this returned solution is optimal.