CSE 150 Homework Assignment 1

Part 6 Report

**Problem 1:**

Description of problem 1: Find a prime path from one number(argument #1) to another(argument #2). The path must only contain prime numbers. Adjacent numbers can only differ by one digit, anywhere in the number. Find the shortest path possible using breadth-first search. Don't visit the same node twice.

Description of algorithm used: Take each digit of the first argument, and find all the primes that can be generated by altering just that one digit. Do the same for all the digits. This is the first layer. Then, take each of the newly generated primes (pop them off the queue), and expand them again by generating all the primes that can be generated by changing one digit at a time. Keep traversing all the layers until the goal node is reached.

**Problem 2:**

* Algorithm: Depth Limited Search
* Implementation: Using stack (FILO) data structure to perform depth search. Using a dictionary to keep track of discovery flag as well as depth. In this implementation, our depth limit is 5 (with root having depth of 0). Thus our code will terminate one search path if the depth of current path is more than 5
* Completeness: In general no, yes if the length of shortest path is no greater than depth limit
* Optimality: In general no, it will only return an optimal path if the length of shortest path is exactly the same as the depth limit
* Space Complexity: This is essentially depth first search, so space complexity should be fairly small—O(bl)
* Time Complexity: since the max depth is limited, the max number of nodes visited are also capped—O(b^l)

**Problem 3:**

* Algorithm: Iterative Deepening Depth First Search
* Implementation: Basic data structure same as Depth Limited Search, except we use a for loop to run it over different max depth
* Completeness: If length of shortest path is smaller than max depth, then yes
* Optimality: whenever it finds a path, it’s optimal
* Space Complexity: For each iteration, its space complexity is the same as depth limited search, thus space complexity is small—O(bl)
* Time Complexity: It’s a constant proportion to depth limited search, thus max number of nodes visited are also capped—O(b^l)