Amanda Shohdy | amandashohdy@csu.fullerton.edu

Project 04 Report

Problem 1: Rentals

Problem 1: Rentals is intended to determine the minimum number of laptops needed for a university to be able to provide each student access when needed. The algorithm takes in intervals as pairs and stores them in a vector.

The vector of intervals is sorted by start time. A vector of interval pairs is created where each element is the current interval a laptop is being rented for. The length of this vector is the minimum number of laptops needed. As an interval is accounted for, it is removed from the interval vector. If an interval begins after the rental period of a laptop ends, the laptop is updated to the current interval period and the current interval is removed from the interval vector. If no laptop is available for the current interval period, the current interval is added to the end of the laptops vector, symbolizing adding another laptop, and is removed from the interval vector. This is repeated until there are no intervals left in the interval vector. The size of the laptops vector is output as the minimum number of laptops needed.

The efficiency class for problem 1 is O(nlog(n) + n*m + 9). Sorting the interval list by start time takes O(nlogn) time. The while loop runs once for each element in the interval list, making it O(n) time complexity. The for loop runs m times (each laptop) for the n items in the interval list, making the time complexity of the while loop O(n*m).

Problem 2: Network Delay Time

Problem 2: Network Delay Time is intended to determine the minimum amount of time a signal takes to travel around a number of nodes given a source node. The algorithm takes in the number of nodes, the source node, and data about the paths and the time the paths take.

First, a for loop checks that at least 1 path option starts with the source node. A vector to keep track of all the nodes visited is created and the source node is added. A vector to keep track of all the tracked paths is created and initialized to have 1 path starting at the source with length 0. If a path option starts with the same node as tracked path, the tracked path is updated to the next node and the length of time is updated. Otherwise, a new tracked path starting at the source node with length 0 is added to the vector and the path option is tried again. After all options are exhausted, if not all nodes have been visited, the algorithm returns -1. Otherwise, the largest length of time is returned.

The efficiency class for problem 2 is O(t*p + t + n + p + 16). The for loop to check that an option starts at the source node takes O(t) time. The while loop that maps each option in times takes O(t) time. The for loop that tests each tracked path against the options in the times vector takes O(p), making the while loop take O(t*p). The for loop to ensure each node is visited takes O(n) time. Lastly, the for loop to determine the minimum amount of time needed takes O(p) time.