CPSC 335 – Project 01 Report

Algorithm 01: The Alternating Disk Problem

Members:

Name	Email
Andrew Lee	aslee@csu.fullerton.edu
Michael Bui	buimichael@csu.fullerton.edu
Natalia Garcia	natgarcia@csu.fullerton.edu

Algorithm Design & Implementation:

For the alternating disk problem, we decided to design an algorithm that implements a doubly linked list, which checks two concurrent disks at a time with the recursiveSort() function. In this case, disks will also be referred to as nodes that will use the function to check for four conditions. Other helper functions like swap() and deleteNode() were also implemented to the program.

To start, the first condition checks if there are no more nodes to the right, which then terminates the program. The second condition checks if a dark disk has a light disk to the left, in which case, will be swapped. The third condition checks for a light disk having a dark disk to the right and if there is, the disks are swapped. Lastly, the fourth condition is there for when the disks don't need to be swapped, so nothing is done. Except for the first condition, the function recursiveSort() is called recursively for the other conditions when the disks are swapped and when there isn't a need to have the disks swapped.

Algorithm 02: Matching Group Schedules

Members:

Name	Email
Andrew Lee	aslee@csu.fullerton.edu
Michael Bui	buimichael@csu.fullerton.edu
Natalia Garcia	natgarcia@csu.fullerton.edu

Algorithm Design & Implementation:

For matching group schedules, we decided to store each schedule into an array of schedules as a data structure which is used to compare the combined schedules of each individual. Data conversion is used to compare times accurately, and the program accepts a dynamic number of peoples' schedules.

First, the program parses the input given and stores the data in a combined schedule array. The schedule and daily availability data is compared to generate the availability of each person. This data is then compared to other peoples' availability and returns the overlapping time intervals where a meeting is possible based on the meeting duration given in the input.