Archie Ross
2/13/23

CSE3500 Homework 3

Question 1: **Interval Scheduling with the latest starting times.**

In order to obtain a reversed version of the algorithm design given to us, we have to start by sorting the jobs by their starting times. We could use the already existing list which is sorted for the earliest, but it's just easier to understand when you're not iterating backwards. Once we have the sorted list, we need to make a set that allows us to store the jobs we select. We call this set S (for selected). Once we've initialized this, we can start the iterations. We need to find the first latest start time and add it to S, we then use that as our start time and find the following compatible components and for every iteration that is compatible, we add it to the set. Each time we add a new component we change the start time to the end time of the newest item in set S. This allows us to find the next compatible start time.

This method isn't much different than what we had before and yields an optimal solution by storing it in set S. This is a greedy algorithm because it forces multiple iterations of a function for each time we add a new item to the set. It solves the problem by breaking it into a problem per item we add to set S and this can be rather greedy when we get to inputs of high volume.

Question 2:

In order to make a greedy algorithm that converts an input of a fraction into a unit fraction, we have to create a set that can contain the fractions that make up the input. We can call this set F. Next we need to find the largest fraction that the input fraction may contain, starting from ½ we can iterate the denominator until the part of the unit fraction can be subtracted from the input fraction. Once we find it, we add it to the set F and find the next fraction and the next until the sum of the set is equal to the input fraction. Each time we add a new item to the set, we have to subtract the item from the input fraction in order to test for the next item to put in the set. Like I just stated, we have to continue to do that until the sum of the set is equal to the input fraction. That way, once the iterations are finished, our optimal answer is contained in set F.

Now, to prove the correctness of this algorithm, we can do so by taking a fraction such that the numerator is less than the denominator and use the method of induction. There is a base case that the numerator is already 1 in which the algorithm will correctly return the same as the input.

We need to show that P(n+1/k) is also true, where k is a positive integer. We start by finding the largest unit fraction that can be subtracted from n+1/k which will be 1/(k+1) if n+1/k is greater than 1/(k+1), otherwise it is 1/k. We can call this fraction u1. We now add this to set F and subtract it from n+1/k resulting in a new fraction. We repeat this process for the new fraction and add it to set F according to the induction hypothesis. If u1 is equal to 1/k we added it to the set F and subtracted it from n +1/k resulting in u1 = n which produces the correct unit fraction we're asking for. If u1 isn't 1/k we continue this until un is 1/k and that will be our final addition to the set. Once we've stopped iteration, the testing has found that the sum of F is equal to the input and that means we have successfully found the unit fraction. This proves that we find the optimal solution asked for by the requirements.