CS 371 HW 3

Josh Blaz

2/19/19

1. Chapter 4 Exercise 3
   O is the optimal solution and G is a greedy solution. Suppose that the first i trucks are loaded the same way for both solutions. Starting at position $i + 1$, the trucks are loaded differently. For the optimal solution O, the truck must have some weight W_O that is less than weight W_G of the greedy solution, as the greedy solution always packs boxes into the truck until no more can fit. The following truck at position $i + 2$ of the optimal solution has to fit more boxes than the truck at position $i+2$ of the greedy algorithm in order to keep up with the packing of the other trucks. This cannot be the case, though, because there are additional boxes that did not fit into the truck at O_{i+1}, those would then fit into O_{i+2}. Given that the greedy solution always packs the truck with as many boxes as possible, it would have already filled up the truck at position G_{i+2}, furthermore, there would be no way for the optimal solution O to fit all of the remaining boxes in the truck at position O_{i+2}, so the greedy solution must be the optimal solution.

2. Chapter 4 Exercise 4

```
Data: Sequence of events $S$ and $S'$, a subsequence of $S$.
Result: Determines whether $S'$ is a subsequence of $S$.
$S$ is of length $n$ and $S'$ is of length $m$.
 i = 1 and j = 1
 while $i < n$ and $j < m$ do
    if $S[i] == S'[j]$ then
       $j = j + 1$
    end
    $i = i + 1$
    if $j >= m$ then
       Return True, $S'$ is a subset of $S$.
    else
       Return False, $S'$ is not a subset of $S$.
    end
end
```

This algorithm works because it iterates through each event in S', checking if the event occurs in S, when it matches an event between S' and S, j is incremented, and it returns True when $j > m$. The worst case run-time of this algorithm is $O(n)$, in the case that S' is not a subset of S, and we have to iterate through all n events in the sequence. The while loop will always terminate, as it breaks when the solution is found ($j \geq m$) or when it doesn't find a solution (when we iterate past the length of sequence S and $i > m$).

3. Chapter 4 Exercise 5

```
Data: Some houses on a road between eastern and westerns endpoints $e$ and $w$
Result: Place towers such that all houses are within 4 miles from a cell tower.
Sort houses by distance from starting point $e$.
 while some house is not within 4 miles from a cell tower do
    Choose the first sorted house $h$.
    Place a tower 4 miles west of that house.
    Remove house $h$ and all houses within 4 miles east or west of the cell tower
    from the list of houses that need cell coverage.
end
```

If there are n houses on the country road that need cell coverage, then the worst case run-time of this algorithm is $O(nlogn)$. This algorithm is capable of satisfying the need for every house to be within 4 miles of a cell tower because the while loop iterates through every house that is not within 4 miles, and ensures that a tower is placed near it. The while loop completes because it iterates until every house is within 4 miles from a cell tower, if a house is not within this range, then we will place a tower within the range of that house in the while loop and that house will not be visited again.

4. Chapter 4 Exercise 7

```
Data: $n$ jobs to schedule through the supercomputer and $n$ PCs.
Result: A schedule for the jobs with minimal completion time.
Sort jobs by longest completion in descending order.
while Some job not pre-processed do
    Feed first job $j$ to the supercomputer to be pre-processed.
    Once job is pre-processed by the supercomputer, send to PCs to be processed.
    Remove job from queue of jobs to be pre-processed.
end
```

The worst case run-time for this algorithm is $O(nlogn)$. This algorithm works because it ensures that every job is pre-processed by the supercomputer and by the PCs afterward. The while loop completes because it iterates until there are no more jobs to process, and when a job is processed, it is removed from the queue. This algorithm minimizes the completion time by queuing the tasks with the longest time needed to process on the PCs first, while these process on the PCs, we queue up and process all other tasks. I designed this algorithm in an attempt to minimize the amount of time spent waiting for the PCs to process their jobs.

5. Chapter 4 Exercise 9

   (a) False. This cannot be the case, as there must be a situation in which the MBT contains a bottleneck that all MSTs in graph G share. It will be an MBT because they will all share the maximum edge that must be traversed. While each other graph shares this bottleneck edge, they could have faster routes to all of the other nodes. This is really tough to do without non-distinct edges.

   (b) True, every MST is a MBT of the same graph G. If we add any edge e with a lesser weight than the bottleneck edge of the MST to the MST, this will create a cycle. To get rid of the cycle, we will remove our bottleneck edge (making e our new bottleneck edge). This is a contradiction as the MST would have already chosen this edge, so it must be true that an MST must be an MBT within the same graph.