CS314: Algorithms 
Homework 8

1. Recall the makespan problem discussed in class. We discussed the fact that our greedy approximation algorithm does not always give an optimal makespan assignment, but only a 2-approximation. Given an example of a set of jobs (along with a number of machines) where the greedy algorithm fails to return a solution with optimal size.

2. Recall the shortest first greedy algorithm for the interval scheduling problem that we discussed in class: Given a set of intervals, repeatedly pick the shortest interval $I$, delete all other intervals that overlap $I$, and repeat as long as there is an interval still in the set.

In an earlier lecture, we saw that this does NOT always produce a maximum size set of non-overlapping intervals. However, it turns out to have the following interesting approximation guarantee. If $s^*$ is the maximum size of a set of non-overlapping intervals, and $s$ is the size of the set produced by our greedy shortest first algorithm, then $s \geq \frac{1}{2}s^*$, so that this greedy algorithm is a 2-approximation. Prove this fact.

3. Suppose you’re acting as a consultant for the Port Authority of an ocean-side city. They’re currently doing good business, and their revenue is constrained almost entirely by the rate at which they can unload the ships arriving in their port.

Here’s a basic sort of problem they face. A ship arrives, with $n$ containers of weight $w_1, w_2, \ldots, w_n$. Standing on the dock is a set of trucks, each of which can hold up to $K$ units of weight. (You can assume the $w_i$’s and $K$ are integers.) You can stack multiple containers in each truck, as long as you don’t exceed total weight $K$ on any one of them; the goal is the minimize the total number of trucks needed. (Note: This problem is NP-Complete, but you don’t need to prove that fact.)

A greedy algorithm (which should look familiar) for this might proceed as follows: Start with an empty truck, and begin piling containers $1, 2, 3, \ldots$ into it until the next container would overflow the capacity $K$. Now declare this truck loaded and send it off, and start loading the next truck. This algorithm, by considering trucks only one at a time, might not get the best total packing.

(a) Give an example of a set of weights and a value of $K$ where this algorithm does not use the minimum number of trucks.

(b) Show, however, that the number of trucks used by this algorithm is within a factor of 2 of the minimum possible number, for any set of weights and any value of $K$. 
