Lindsey Bieda and Joe Frambach
Greedy Algorithm Problems
9.1.2011

1. Consider the following problem:
INPUT: A set $S = \{(x_i, y_i) | 1 \leq i \leq n\}$ of intervals over the real line.
OUTPUT: A maximum cardinality subset S' of S such that no pair of intervals in S' overlap.
Consider the following algorithm:

Repeat until S is empty

1. Select the interval I that overlaps the least number of other intervals.
2. Add I to final solution set S'.
3. Remove all intervals from S that overlap with I.

Prove or disprove that this algorithm solves the problem.

Consider the following input:
$S = \{(1,3), (3,5), (5,7), (7,9), (2,4), (4,6), (6,8), (2,4), (6,8)\}$.
It is constructed by concatenating an optimal solution of cardinality 4:
$OPT = \{(1,3), (3,5), (5,7), (7,9)\}$
with a non-optimal solution of cardinality 3:
$NONOPT = \{(2,4), (4,6), (6,8)\}$
then adding additional intervals to force the non-optimal $(4,6)$ to be chosen first:
$ADD = \{(2,4), (6,8)\}$.

The algorithm does not solve the problem. S yields a non-optimal solution.

2. Consider the following Interval Coloring Problem.

INPUT: A set $S = \{(x_i, y_i) | 1 \leq i \leq n\}$ of intervals over the real line. Think of interval $(x_i, y_i$ as being a request for a room for a class that meets from time x_i to time y_i.
OUTPUT: Find an assignment of classes to rooms that uses the fewest number of rooms.

Note that every room request must be honored and that no two classes can use a room at the same time.

(a) Consider the following iterative algorithm. Assign as many classes as possible to the first room (we can do this using the greedy algorithm discussed in class, and in the class notes), then assign as many classes as possible to the second room, then assign as many classes as possible to the third room, etc. Does this algorithm solve the Interval Coloring Problem? Justify your answer.

Let s be the maximum number of intervals that overlap at one particular point in time. For each iteration of the algorithm the maximum number of non-overlapping times will be selected. From this we can assume that the maximum number of iterations in order for this algorithm to complete will be equivalent to s. As we assign a single room in each iteration the maximum number of rooms will be s. This algorithm does solve the Interval Coloring Problem as it is able to find the optimal solution.

(b) Consider the following algorithm. Process the classes in increasing order of start time. Assume that you are processing class C. If there is a room R such that R has been assigned to an earlier class, and C can be assigned to R without overlapping previously assigned classes, then assign C to R. Otherwise, put C in a new room. Does this algorithm solve the Interval Coloring Problem? Justify your answer.

Let s be the maximum number of intervals that overlap at one particular point in time. Given that classes are processed in increasing order of start times we can interpret that to be the earliest classes will be processed first. In each step the earliest class of the unprocessed set U is placed into room R given the criteria that room R has been assigned to an earlier class and C does not overlap with any of the classes assigned to room R. Each class must be placed in a room with a class earlier than it in it and since the classes are processed from the earliest class onward we can state that classes must attempt to be placed in a room with a class already in it. The only other criteria examined is if an overlap exists or not. This means that only in the case that all R with classes in them all overlap with class C will an additional room be allocated. In which case the number of rooms is constrained in this algorithm by s. Therefore, this algorithm does solve the Interval Coloring Problem.