Greedy Algorithms I

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_0 of S such that no pair of intervals in S_0 overlap.

   Consider the following algorithm:
   1. Repeat until S is empty
   2. Select the interval I that overlaps the least number of other intervals.
   3. Add I to the final solution set S_0z
   4. Remove all intervals from S that overlap with I.

   Prove or disprove that this algorithm solves the problem.
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 a request 000000000000000000000000000000000000000000
   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. 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), 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.
3. We consider two change-making problems:
   a. The input to this problem is an integer L. The output should be the minimum cardinality collection of coins required to make L shillings of change (that is, you want to use as few coins as possible). The coins are worth 1, 5, 10, 20, 25, 50 Shillings. Assume that you have an unlimited number of coins of each type. Formally prove or disprove that the greedy algorithm (that takes as many coins as possible from the highest denominations) correctly solves the Change Problem. So for example, to make a change for 234 Shillings the greedy algorithms would take four 50 shilling coins, one 25 shilling coin, one 5 shilling coin, and four 1 shilling coins.
   b. The input to this problem is an integer L. The output should be the minimum cardinality collection of coins required to make L nibbles of change (that is, you want to use as few coins as possible). Now the coins are worth $1, 2, 2^2, 2^3, \ldots 2^{1000}$ nibbles. Assume that you have an unlimited number of coins of each type. Prove or disprove that the greedy algorithm (that takes as many coins of the highest value as possible) solves the change problem.
      HINT: The greedy algorithm is correct for one of the above two subproblems and is incorrect for the other. For the problem where greedy is correct, use the following proof strategy: Assume to reach a contradiction that there is an input I on which greedy is not correct. Let OPT(I) be a solution for input I that is better than the greedy output G(I). Show that the existence of such an optimal solution OPT(I) that is different from greedy is a contradiction. So what you can conclude from this is that for every input, the output of the greedy algorithm is the unique optimal/correct solution.
4. You wish to drive from point A to point B along a highway minimizing the time that you are stopped for gas. You are told beforehand the capacity C of your gas tank in liters, your rate F of fuel consumption in liters/kilometer, the rate r in liters/minute at which you can fill your tank at a gas station, and the

locations $A = x_1, \ldots x_n = B$ of the gas stations along the highway. So if you stop to fill your tank from 2 liters to 8 liters, you would have to stop for $6/r$ minutes. Consider the following two algorithms:

a. Stop at every gas station and fill the tank with just enough gas to make it to the next gas station.
b. Stop if and only if you don't have enough gas to make it to the next gas station, and if you stop, fill the tank up all the way.

For each algorithm either prove or disprove that this algorithm correctly solves the problem. Your proof of correctness must use an exchange argument.

5. Consider the following problem. The input is a collection $A = \{a_1, \ldots a_n\}$of n points on the real line. The problem is to find a minimum cardinality collection S of unit intervals that covers every point in A. Another way to think about this same problem is the following. You know a collection of times (A) that trains will arrive at a station. When a train arrives there must be someone manning the station. Due to union rules, each employee can work at most one hour at the station. The problem is to find a schedule of employees that covers all the times in A and uses the fewest number of employees.
(a) Prove or disprove that the following algorithm correctly solves this problem. Let I be the interval that covers the most number of points in A. Add I to the solution set S. Then recursively continue on the points in A not covered by I.
(b) Prove or disprove that the following algorithm correctly solves this problem. Let a_j be the smallest (leftmost) point in A. Add the interval $I = (a_j, a_{j+1})$to the solution set S. Then recursively continue on the points in A not covered by I.
HINT: One of the above greedy algorithms is correct and one is incorrect for the other. The proof of correctness must use an exchange argument.
6. We consider a greedy algorithm for two related problems
    a. The input to this problem consists of an ordered list of n words. The length of the ith word is w_i that is the ith word takes up w_i spaces. (For simplicity assume that there are no spaces between words.) The goal is to break this ordered list of words into lines, this is called a layout. Note that you can not reorder the words. The length of a line is the sum of the lengths of the words on that line. The ideal line length is L. No line may be longer than L, although it may be shorter. The penalty for having a line of length K is $L - K$. The total penalty is the sum of the line penalties. The problem is to find a layout that minimizes the total penalty. Prove or disprove that the following greedy algorithm correctly solves this problem.
    For i= 1 to n
        Place the ith word on the current line if it fits
        else place the ith word on a new line
    b. The input to this problem consists of an ordered list of n words. The length of the ith word is w_i, that is the ith word takes up $w_i$spaces. (For simplicity assume that there are no spaces between words.) The goal is to break this ordered list of words into lines, this is called a layout. Note that you can not reorder the words. The length of a line is the sum of the lengths of the words on that line. The ideal line length is L. No line may be longer than L, although it may be shorter. The penalty for having a line of length K is $L - K$. The total penalty is the maximum of the line penalties. The problem is to find a layout that minimizes the total penalty. Prove or disprove that the following greedy algorithm correctly solves this problem.
    For i= 1 to n
        Place the ith word on the current line if it fits
        else place the ith word on a new line
    HINT: The greedy algorithm is correct for one of the above two problems and is incorrect for the other. The proof of correctness must be done using an exchange argument.

7. Consider the following problem. The input consists of the lengths $l_1, \ldots, l_n$, and access probabilities $p_1, \ldots, p_n$, for n files $F_1, \ldots, F_n$. The problem is to order these files on a tape so as to minimize the expected access time. If the files are placed in the order $F_{s(1)}, \ldots F_{s(n)}$ then the expected access time is $\sum_{i=1}^{n} p_{s(i)} \sum_{j=1}^{i} l_{s(j)}$. Note that the term $p_{s(i)} \sum_{j=1}^{i} l_{s(j)}$ is the probability that you access the ith file times the length of the first i files. For each of the below algorithms, either give a proof that the algorithm is correct using an exchange argument, or a proof that the algorithm is incorrect.
    a. Order the files from shortest to longest on the tape. That is, $l_i < l_j$ implies that $s(i) < s(j)$.
    b. Order the files from most likely to be accessed to least likely to be accessed. That is, $p_i < p_j$ implies that $s(i) > s(j)$.
    c. Order the files from the smallest ratio of the length over access probability to the largest ratio of the length over access probability. That is, $l_i/p_i < l_j/p_j$ implies that $s(i) < s(j)$.