Greedy Algorithms II

1. Consider the following problem. The input consists of n skiers with heights $p_1, \ldots, p_n$, and n skies with heights $s_1, \ldots s_n$. The problem is to assign each skier a ski to minimize the average difference between the height of a skier and his/her assigned ski. That is, if the ith skier is given the $\alpha(i)$th ski, then you want to minimize: $\frac{1}{n}\sum_{i=1}^{n} |p_i - s_{\alpha(i)}|$
    1. Consider the following greedy algorithm. Find the skier and ski whose height difference is minimized. Assign this skier this ski. Repeat the process until every skier has a ski. Prove or disprove that this algorithm is correct.
    2. Consider the following greedy algorithm. Give the shortest skier the shortest ski, give the second shortest skier the second shortest ski, give the third shortest skier the third shortest ski, etc.
       Prove or disprove that this algorithm is correct.

    HINT: One of the above greedy algorithms is correct and one is incorrect for the other. The proof of correctness must be done using an exchange argument.
2. We consider the following scheduling problem:
   INPUT: A collection of jobs $J_1, \ldots, J_n$, where the ith job is a tuple (r_i, x_i) of non-negative integers specifying the release time and size of the job.
   OUTPUT: A preemptive feasible schedule for these jobs on one processor that minimizes the total completion time $\sum_{i=1}^{n} C_i$.

   A *schedule* specifies for each unit time interval, the unique job that is run during that time interval. In a *feasible* schedule, every job J_i has to be run for exactly x_i time units after time r_i. The completion time C_i for job J_i is the earliest time when J_i has been run for x_i time units. Examples of these basic definitions can be found below.
   We consider two greedy algorithms for solving this problem that schedule times in an online fashion, that is the algorithms are of the following form:

   $t = 0$
   while there are jobs left not completely scheduled
   Among those jobs J_i such that $r_i \leq t$, and that have previously been scheduled for less than x_i time units, pick a job J_m to schedule at time t according to some rule;
   increment t

   One can get different greedy algorithms depending on the rule for selecting Jm. For each of the following
   greedy algorithms, prove or disprove that the algorithm is correct. Proofs of correctness must use an exchange argument.
    1. SJF: Pick J_m to be the job with minimal size x_i. Ties may be broken arbitrarily.
    2. SRPT: Let $y_{i,t}$ be the total time that job J_i has been run before time t. Pick J_m to be a job that has minimal remaining processing time, that it, that has minimal $x_i - y_{i,t}$. Ties may be broken arbitrarily.

   As an example of SJF and SRPT consider the following instance: J_1 = (0,100), J_2 = (10,10) and

J_3= (1,4). Both SJF and SRPT schedule job J_1between time 0 and time 1, and job J_3 between time 1 and time 5, when job J_3completes, and job J_1again between time 5 and time 10. At time 10, SJF schedules job J_2because its original size 10 is less than job J_1's original size 100. At time 10, SRPT schedules job J_2because its remaining processing time 10 is less than job J_1's remaining processing time 94. Both SJF and SRPT schedule job J_2between time 10 and 20, when J_2 completes, and then job $J_1$from time 20 until time 114, which job J_1completes. Thus for both SJF and SRPT on this instance C_1= 114, C_2= 20 and C_3= 5 and thus both SJF and SRPT have total completion time 139.

3. We consider the following scheduling problem:
   INPUT: A collection of jobs $J_1, \ldots, J_n$. The size of job J_i is x_i, which is a non-negative integer. An integer m.
   OUTPUT: A non-preemptive feasible schedule for these jobs on m processors that minimizes the total completion time $\sum_{i=1}^{n} C_i$.
   A schedule specifies for each unit time interval and for each processor, the unique job that is run during that time interval on that processor. In a feasible schedule, every job J_i has to be run for exactly x_i time units after time 0. In a non-preemptive schedule, once a job starts running on a particular processor, it has to be run to completion on that particular processor. The completion time Ci for job $J_i$is the earliest time when J_i has been run for x_i time units. So for example if m = 2 jobs of size 1,4,3 are run in that order on the first processor, and jobs of size 7,10 are run on the second processor in that order, then the total completion time would be 1 + 5 + 8 + 7 + 17 = 38. Give a greedy algorithm for this problem and prove that it is correct.
4. Consider the following problem.
   INPUT: Positive integers $r_1, \ldots, r_n$and $c_1, \ldots, c_n$.
   OUTPUT: An $n \times n$ matrix A with 0/1 entries such that for all i the sum of the ith row in A is r_i and the sum of the ith column in A is c_i, if such a matrix exists.
   Consider the following greedy algorithm that constructs A row by row. Assume that the first $i \geq 1$ rows have been constructed. Let a_j be the number of 1's in the jth column in the first i rows. Now the r_i columns with maximum $c_j - a_j$ are assigned 1's in row i, and the rest of the columns are assigned 0's. That is, the columns that still needs the most 1's are given 1's. Formally prove that this algorithm is correct using an exchange argument.
5. We consider two problems where the input contains n jobs, where each job j has an integer release time r_j, and an integer deadline d_j. In a feasible schedule, each job j must be run for one unit of time, not starting before r_j, and not ending after d_j. Note that no two jobs may be run at the same time. Assume all release times are nonnegative, and let $T = max_j d_j$.
   1. In the warmup problem, the input also specifies for each integer $t \in [0, T)$ whether a particular machine is turned on during the time interval $(t, t+1)$. The problem is to determine if each job j can feasibly be run for one unit of time when the machine is turned on. Give a greedy algorithm and prove that it is correct using an exchange argument.
   2. In the real problem, the input also contains a positive integer L. The problem is to determine the minimum number k of time intervals of length L such all jobs can feasibly be run for one unit of time during one of these k intervals.

Another way to interpret this problem: It costs a dollar to turn the machine on. Once the machine is on, it will stay on for L units of time. The question is to figure out the least number of times to turn the machine on so that one can finish all the jobs (which each have to be run for one unit of time). Give a greedy algorithm and prove that it is correct using an exchange argument.

6. The setting for this problem is a line network modeled by a line graph. There are n nodes in this graph, and each node is connected to a left neighbor and a right neighbor (except the leftmost node has no left neighbor and the rightmost node has no right neighbor). The input consists of k packets where packet p consists of an integer release time r_p when the packet p arrives in the system, a source node s_p at which the packet p arrives, and a destination node t_p(which must be to the right of s_p) to which the packet p must reach. Between consecutive integer times, one packet may be forwarded between each pair of adjacent routers. Once a packet p reaches its destination t_p, it leaves the system. We want all packets to reach their destinations. For example if the input consisted of the following triples of the form (name, r_p, s_p, t_p): (A, 0, 1, 3) (B, 0, 2, 4) (C, 1, 2, 3), then one way to have each packet reach its destination is:
    1. between time 0 and 1 packet A is forwarded from node 1 to node 2, and packet B is forwarded from node 2 to node 3
    2. between time 1 and time 2 packet A is forwarded from node 2 to node 3 (leaving the system at time 3) and packet B is forwarded from node 3 to node 4 (leaving the system at time 3). We could have forward packet C instead of packet A, but it is not possible to forward both at this time
    3. between time 2 and time 3, packet C is forward from node 2 to node 3 (leaving the system at time 4)

    (a) Assume that the objective is to minimize the maximum time that a packet leaves the network. That is, we want to clear the network of packets as quickly as possible. Give a greedy algorithm for this problem and prove that it is correct.
    (b) Assume that n = 2, so that the network consists of two nodes and one edge. Further, assume that the objective is to minimize the maximum waiting time for any packet. The waiting time for a packet is the difference between the time that the packet reaches its destination and the release time for that packet. That is, we want the packet that waits for the longest to wait as little time as possible. Give a greedy algorithm for this problem and prove that it is correct.
    (c) Assume all the release times are zero. Further, assume that the objective is to minimize the sum of the waiting times of the packets. This is equivalent to minimizing the average waiting times of the packets. Give a greedy algorithm for this problem and prove that it is correct. This problem is quite nontrivial.
7. Consider the following bridge crossing problem where n people with speeds $s_1, \ldots s_n$ wish to cross the bridge as quickly as possible. The rules remain:
    It is nighttime and you only have one flashlight.
    A maximum of two people can cross at any one time
    Any party who crosses, either 1 or 2 people must have the flashlight with them.
    The flashlight must be walked back and forth, it cannot be thrown, etc.
    A pair must walk together at the rate of the slower person's pace.

    Give an efficient algorithm to find the fastest way to get a group of people across the bridge. You must have a proof of correctness for your method.
8. A couple is divorcing and need to partition n goods $G_1, \ldots, G_n$. Each of the husband and wife has a valuation for each item. But you do not know this valuation. However, based on their valuations, each of the husband and wife does provide you with an ordered list specifying the order that they value the items. Your goal is to determine whether the lists contain enough information to determine with certainty

that it is possible to partition the goods so that each of the husband and wife thinks that the total value of the goods that they receive is more than half of the total valuation of the goods (according to their evaluation). Let us call this a fair partition. If so, you should give a fair partition. Note that the husband and wife may value goods very differently.
So for example if there were 4 goods, the husband's list (from most to least desirable was): G_2, G_1, G_3, G_4 , and the wife's list was: G_4, G_1, G_3, G_2, then a fair partition is possible by giving the husband $G_2$and G_3, and the wife $G_4$and G_1. In contrast, if there were 4 goods, the husband's list (from most to least desirable was): G_2, G_1, G_3, G_4,, and the wife's list was: G_2, G_1, G_3, G_4, then there is not enough information to determine whether a fair partition is possible. Give a greedy algorithm for this problem and prove that it is correct.