COMP 7270 Assignment 4 No late submissions!

Upload your submission well before this deadline. Even if you are a few minutes late, as a result of which Canvas marks your submission late, <u>your assignment may not be accepted</u>.

Instructions:

1. This is an individual assignment. You should do your own work. **Any evidence of copying will result in a zero grade and additional penalties/actions.**
2. Late submissions **will not** be accepted unless prior permission has been granted or there is a valid and verifiable excuse.
3. **Think carefully; formulate your answers, and then write them out concisely** using English, logic, mathematics and pseudocode (<u>no programming language syntax</u>).
4. Algorithms should be provided in numbered pseudocode steps.
5. **Type your answers in this Word document and submit it. If that is not possible, use a word processor to type your answers as much as possible (you may hand-write/draw equations and figures), turn it into a PDF document and upload**.

All questions carry equal weight

Greedy Algorithm

1. (16.1-2) To prove that the stated approach yields an optimal solution, we have to prove two things: (1) that the choice being made is the greedy choice (this proof will be along the lines of the proof of Theorem 16.1 in the text; but do not copy that proof; your proof will be different, yet similar in structure), and (2) that the resulting solution has optimal substructure.
   This would be an example of a greedy algorithm, we are just starting from the end. When this is proposing corresponds to the solution that the text finds for the reverse. So, in this case it would be considered optimal. This is because the reverse optimal solution maps to the regular optimal solution.

2. (16.1-4) Suppose that we have a set of activities to schedule among a large number of lecture halls, where any activity can take place in any lecture hall. We wish to schedule all the activities using as few lecture halls as possible. Give an efficient greedy algorithm to determine which activity should use which lecture hall.
   There are two different algorithms that can be used for this problem. I have provided the most efficient. This algorithm will exit once every activity has been scheduled to a room

   ```
   LHS(S):
   int j = 1
   for int i = 2 to n
           if activity(i) = hallActivityType(j)
                   hallActivityType (j)+=activity(i)
           else
                   hallActivityType (j+1)+=activity(i)
   j=j+1
   ```
   the run time for this would be in O(n log n)

3. (16.2-1) Prove that the fractional knapsack problem has the greedy-choice property.
We can prove this by using either induction or contradiction. I will use contradiction for this problem.
N=item count
V_i = value of the ith item
C = capacity
S=solution[array]
We can also assume that the items are sorted by value and capacity.
The greedy algorithm will assign the ith value in the solution the minimum capacity, then continuing until either the capacity OR the item count = 0.
If we decrease the ith solution value to max(0,C-C_n) AND increase the solution item count by the same, this will provide a better solution. Because this is a contradiction, the assumption would be false, hence the greedy choice property.

4. (16.2-7) Suppose you are given two sets *A* and *B*, each containing n positive integers. You can choose to reorder each set however you like. After reordering, let a_i be the *i*-th element of set *A*, and let b_i be the *i*-th element of set *B*. You then receive a payoff of $\prod_{i=1}^{n} a_i^{b_i}$. Give an algorithm that will maximize your payoff. Prove that your algorithm maximizes the payoff, and state its running time.
The solution for this would be to use a greedy algorithm. We can assume the following:
$a_i = a_1 >= a_2 >= a_3 \ldots >= a_n$
$b_i = b_1 >= b_2 >= b_3 \ldots >= b_n$
we will sort these in decreasing order
we are also assuming a<b
this would return the following equation:

HW4 Robin Ward

$\prod_{i=1}^{n} a_i^{b_i}$ in which a_i is paired with b_i

Also, if these pairs are sorted, the run time = O(n)

If they need to be sorted, this will change the run time = O(n log n)

5. Let's consider a long, straight country road with *n* houses scattered very sparsely along it. We can picture this road as a long line segment with an eastern endpoint and a western endpoint. Let d_i be the distance of the *i*-th house from the eastern endpoint. We are given as input d_i, for all $1 £ i £ n$. Further, let's suppose that despite the bucolic setting, the residents of all these houses are avid cell phone users. You want to place cell phone base stations at certain points along the road, so that every house is within four miles of one of the base stations. Give an efficient algorithm that achieves this goal, using as few base stations as possible. Identify the complexity of your algorithm.

For a greedy solution to this, we can picture a road running from east to west/ left to right. Along the road are houses=h. to satisfy these requirements, we can move west/right until we reach h(house). That will then be our first base station, followed by removed the houses associated to that base station. We will then repeat this process until every house is covered. The complexity would be O(n) if h is already sorted, otherwise it will be O(n log n)