CSC 2420 Fall 2017, Assignment 1
Due date: October 18

It is certainly preferable for you to solve the questions without consulting a published source. However, if you are using a published source then you must specify the source and you should try to improve upon the presentation of the result.

If you would like to discuss any questions with someone else that is fine BUT at the end of any collaboration you must spend at least one hour playing video games or watching two periods of Maple Leaf hockey or maybe even start reading a good novel before writing anything down.

Unless stated otherwise, all subquestions (1(i), 1(ii), 1(iii), 1(iv), etc) are worth 10 points. If you do not know how to answer a question, state "I do not know how to answer this (sub) question" and you will receive 20% (i.e. 2 of 10 points) for doing so. You can receive partial credit for any reasonable attempt to answer a question BUT no credit for arguments that make no sense.

In class I can clarify any questions you may have about this assignment.

1. Consider the makespan problem for the identical machines model with m machines. We sketched the proof that the worst case *competitive ratio* $\frac{C_{Greedy}}{C_{OPT}}$ of the natural online greedy algorithm is $\leq 2 - \frac{1}{m}$. We also gave an example showing that this ratio is "tight" (i.e. cannot be improved) for this algorithm.

   (i) Finish the proof that was sketched in Lecture 1; that is, use the fact that $C_{OPT} \geq \sum_{1 \leq i \leq n} p_i/m$ for any sequence of n input jobs and $C_{OPT} \geq p_i$ where job J_i has "load" p_i.

   (ii) Argue for $m = 2$ (resp. $m = 3$) machines that *any* (not necessarily greedy) online algorithm would have competitive ratio no better than $\frac{3}{2}$ (resp. $\frac{5}{3}$) so that the above greedy online algorithm approximation is tight for $m = 2$ and $m = 3$ for any online algorithm.

   (iii) Consider greedy online algorithm for the makespan problem but now in the random order ROM model. Show that for any $\epsilon > 0$, there exists a sufficiently large m such that the (expected) approximation ratio $\frac{E[C_{Greedy}]}{C_{OPT}} \geq 2 - \epsilon$. Here the expectation is with respect to the uniform distribution on input arrival order.
   If you cannot prove the stated claim then prove any ratio greater than 1. Hint: generalize the nemesis sequence for the adversarial competitive ratio.

   (iv) Consider the LPT algorithm for the makespan problem. The major steps in the proof are as follows:

      (a) Without loss of generality, the job causing the makespan is p_r, the job having minimum processing cost.

      (b) Use the two facts above about bounds for OPT to show that if the stated approximation bound does not hold, then $p_r > OPT/3$. It follows that OPT can only schedule at most 2 jobs per machine.

      (c) Show how to transform OPT schedule into the LPT schedule without increasing the makespan and thereby deriving a contradiction.

   Provide arguments for the last two major steps.

2. Consider the knapsack problem with input items $\{(v_1, s_1), \ldots, (v_n, s_n)\}$ and capacity C. Without loss of generality the sizes s_j of all items are at most C. Consider the following "natural" greedy algorithms which initially sort the input set and then schedule greedily (i.e. takes the item if it fits). For each algorithm provide input instances which show that the algorithm will not achieve a c-approximation for any constant c.
   Note: For definiteness, assume all input values are integral which, in principle, could make an inapproximation result harder. But here it should be easy to derive appropriate integral examples.

   (i) *Greedy by value*: Sort the items $I_j = (v_j, s_j)$ so that $v_1 \geq v_2 \ldots \geq v_n$.

   (ii) *Greedy by size*: Sort the items so that $s_1 \leq s_2 \ldots \leq s_n$.

   (iii) *Greedy by value-density*: Sort the items so that $\frac{v_1}{s_1} \geq \frac{v_2}{s_2} \ldots \geq \frac{v_n}{s_n}$

   Note: In fact, no deterministic priority algorithm can provide a constant approximation.

3. For the knapsack problem, consider the algorithm that returns the maximum of "Greedy by value" and "Greedy by value-density" as defined in the previous question. Return the better of the two solutions. Show that this algorithm is a 2-approximation for the knapsack problem by showing the following:

   (i) Let item t be the first item that is rejected by Greedy by value density. That is, when $v_1/s_1 \geq v_2/s_2 \ldots \geq v_n/s_n$ then $\sum_{i=1}^{t-1} s_i \leq C$ and $\sum_{i=1}^{t} s_i > C$ where C is the capacity bound. (We can assume there is such a t since otherwise if all items fit in the knapsack then any greedy algorithm will be optimal.) Show that $\sum_{i=1}^{t} v_i \geq OPT$

   (ii) Show how the above fact implies that the algorithm that returns the maximum of "Greedy by value" and "Greedy by value-density" is a 2-approximation.

   Note: This algorithm is a special case of Sahni's PTAS argument but do not use that result.

4. Consider the following knapsack type problem. We need to place a subset of n items in a railroad car of integral length C; each item has the same width and height (that of the car) and different integral lengths. The items are of two types, containing exactly one of two chemical substances, call them R and B type items. Each item I_j has a value v_j. To avoid undesired chemical reactions, between any two R items there must at least one B item. Determine what items to load in the car so as to maximize the value of the items placed in the car. Provide a dynamic programming algorithm with time complexity polynomial in n and C for this problem. Specify in words whatever array you are using and the associated recursive definition for computing the entries in this array. Indicate how the desired output is obtained.

5. Consider the following scheduling problem. Each job J_i is described by a tuple (d_i, p_i, v_i) where d_i is the deadline, p_i is the processing time and v_i is the value of the job if scheduled so that it finishes processing by its deadline. The goal is to schedule jobs without any overlap so as to maximize the value of the items scheduled. The desired time complexity is to be polynomial in n and $\max_i v_i$. Specify in words the array you are using and the associated recursive definition for computing the entries in this array. Indicate how the desired output is obtained.