CSC 2420 Spring 2015, Assignment 1
Due date: February 11,2016 at start of class

NOTE: If you are taking the course for credit, then you may either work by yourself or with at most one other student taking the course for credit. You must specify with whom you are collaborating and the extent of collaboration. 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.

1. 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 sorts the input set and then schedules greedily (i.e. takes the item if it fits). For each algorithm provide input instances which show that these algorithms 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.

   - *Greedy by value*: Sort the items $I_j = (v_j, s_j)$ so that $v_1 \geq v_2 \ldots \geq v_n$.
   - *Greedy by size*: Sort the items so that $s_1 \leq s_2 \ldots \leq s_n$.
   - *Greedy by value-density*: Sort the items so that $\frac{v_1}{s_1} \geq \frac{v_2}{s_2} \ldots \geq \frac{v_n}{s_n}$

2. Consider the special case of the knapsack problem with $v_i = s_i$ for all i which we will call *the simple or proportional profit knapsack problem*. Specify and analyze a greedy algorithm that provides a constant approximation for the simple knapsack problem. That is, specify and prove an approximation ratio.

3. For the knapsack problem, consider the algorithm that returns the maximum of "Greedy by value" and "Greedy by value-density" as defined in question 1. Return the better of the two solutions. Show that this algorithm is a 2-approximation for the knapsack problem by showing the following:

   - 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 $\sum_{i}^{t} v_i \geq OPT$
   - 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.

4. Consider set packing problem and the "greedy-by-weight-per-size" algorithm. Use a charging argument to show that this algorithm provides an s approximation for the weighted s-set packing problem.
   Show also that this approximation ratio is tight; that is, for every $\epsilon > 0$, there is an s-packing problem instance for which the "greedy-by-weight-per-size" algorithm will result in an approximation ratio no better than $s - \epsilon$.

5. Consider Graham's online greedy makespan algorithm for identical machines. In the online model, this algorithm has a tight approximation ratio of $2 - \frac{1}{m}$ for m machines. Now consider the same algorithm in the random order model (ROM). Show that for any $\epsilon > 0$, and for some m sufficiently large, the expected approximation ratio of the algorithm is at least $2 - \epsilon$. That is, for some m sufficiently large, find a problem instance and an input set $\{J_1, \ldots, J_n\}$ such that the expected makespan (randomizing over the $n!$ possible input orderings) will be at least $2 - \epsilon$ times larger than the OPT makespan.
   Open problem (Or at least I don't know the answer.) Is there an algorithm for this makespan problem which in the ROM model achieves an expected approximation ratio no worse than $2 - \epsilon$ for some fixed ϵ and all m.

6. Consider the makespan problem in the related machines model where there are two types of machines, those that run at speed $s_1 > 1$ and those that run at speed $s_2 = 1$. That is, if a job j with processing time p_j is run on machine i, then it completes in time p_j/s_i. Suppose that there are only d different processing times. Here we view the number of machine m as part of the input and not a fixed constant. Provide a dynamic programming (DP) algorithm with time complexity $n^{O(d)}$ for computing the value of an optimal makespan solution for this problem where n is the number of jobs. Specify whatever "semantic array(s)" you are using and the associated recursive definition for computing the entries in these arrays. Specify how the desired output is obtained.

7. 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. For delivery purposes, the items chosen for delivery have to be delivered in the order given. The items are primarily composed of one of two chemical substances, call them R and B items. To avoid undesired chemical reactions, between any two R items there must be a B item. Each R item is valued at \$5000 and each B item is valued at \$2000. Place the items in the car (larger indices will be delivered first) 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 whatever "semantic array(s)" you are using and the associated recursive definition for computing the entries in these arrays. Specify how the desired output is obtained.