CSC 2420 Spring 2015, Assignment 1
Due: January 29 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.
NOTE: Do not worry if you cannot solve every question. Do the best you can, If you have some useful partial ideas then mention them. But if you do not have any ideas on a given question, I prefer that you just leave the question blank. I award 20% of the value of the question for the admission "I do not know how to answer this question". That is, I credit people for knowing what they do not know.

1. Consider the knapsack problem with input items $\{(v_1, s_1), \ldots, (v_n, s_n)\}$ and capacity C. WLG the sizes s_j of all items are at most C. Let $Greedy_{val}(\mathcal{I})$ (resp. $Greedy_{val-d}(\mathcal{I})$) be the greedy algorithm that first sorts the items $I_j = (v_j, s_j)$ so that $v_1 \geq v_2 \ldots \geq v_n$ (resp. $\frac{v_1}{s_1} \geq \frac{v_2}{s_2} \ldots \geq \frac{v_n}{s_n}$) and then accepts items greedily (i.e. as long as they fit).

    - Show that neither $Greedy_{val}$ nor $Greedy_{val-d}$ have constant approximation ratios.
    - Show that no priority algorithm can have a constant approximation.
    - Consider the special case of $v_i = s_i$ for all i which we will call *the simple knapsack problem*. Specify and analyze a greedy algorithm that provides a constant approximation for the simple knapsack problem.
    - Consider the following maximum of two algorithms: Let $i^* = argmax_i\{v_i | i = 1 \ldots n\}$ and let $A = \{I_{i^*}\}$ and $B = Greedy_{val-d}(\mathcal{I})$ Return the better of the two solutions. Show that this algorithm is a 2-approximation for the knapsack problem. (Do not simply quote or apply Sahni's result but rather prove the 2-approximation.)

Clarification: Whenever a tie has to be broken (e.g. in choosing the index i^* of the most valued item), you can assume any tie breaking rule so that the algorithm is well defined. For example, in choosing the index i^* of the most valued item, $argmax_i$ returns a single index and not a set of indices.

2. The following questions refer to the s-set packing problem and the (arbitrary cardinality) set packing problem where the underlying universe has size $m = |U|$.

   - Consider 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.
   - Again, consider the "greedy-by-weight-per-size" algorithm. Use a charging argument to show that the algorithm provides a $\sqrt{m}$ approximation for the unweighted set packing problem. Recall that in the unweighted case, it is equivalent to say that the algorithm sorts the input sets so that $|S_1| \leq |S_2| \ldots \leq |S_N|$. Hint: Consider the i^{th} set (call it X_i) accepted by the greedy algorithm and suppose this set has size x. Show that the number of OPT sets that can be charged to X_i is at most $\min\{x, m/x\}$.
   - Consider the partial enumeration greedy algorithm $PGreedy_k$ for set packing as defined in the class and lecture notes. Show that this algorithm provides a $2\sqrt{m/k}$ approximation for the weighted set packing problem.
     Note: If you would like something a little simpler, you can show that $PGreedy_k$ with $k = 1$ is a $2\sqrt{m}$ approximation.
     Clarification: In the algorithm $PGreedy_k$, Max_k is the best solution possible when restricting solutions to those containing at most k sets. Again, as in question 1, you can assume any tie breaking rule so that the algorithm is well defined.

3. Consider the classical secretary problem (i.e. ROM model) with N inputs.

   - Show how to choose an item so that the expected value of the item is at least a $\frac{1}{4}$ of the best item. Give the analysis showing this $\frac{1}{4}$

approximation. (This is, of course, not as good as the known optimal $\frac{1}{e}$ result; but the analysis is more immediate.)

- Suppose now we are in the online model (i.e. adversarial order). Show that no deterministic online algorithm can acheive a constant approximation with respect to the maximum value obtainable. (The same holds true for randomized online algorithms but for now just consider deterministic algorithms.)

4. This problem concerns the WISP problem on one and two machines. Let the intervals $\{I_1, \ldots, I_n\}$ be sorted so that $f_1 \leq f_2 \ldots \leq f_n$.

   - Consider the following semantic array as an alternative for solving the one machine problem:
     $V[k]$ = the maximum value of a feasible solution restricted to $\{I_1, \ldots, I_k\}$ such I_k is scheduled and 0 if I_k is not scheduled. Provide a corresponding computational array and indicate how the desired optimum value is derived from this array.
   - Show that for two machines it is not optimal to first optimally schedule on the first machine and then optimally schedule the remaining jobs on the second machine.
   - Give an optimal DP for WISP on two machines.
     Hint: Use the idea in the alternative one machine DP given above.