CS 4820, Spring 2018 **Homework 11, Problem 2**
Name: Rongguang Wang
NetID: rw564
Collaborators: Siyuao Liu (sl2928); Yihao Chen (yc2288)

(2) *(15 points)*
Recall that in the Knapsack Problem, one is given a set of items numbered $1, 2, \dots, n$, such that the i^{th} item has value $v_i \geq 0$ and size $s_i \geq 0$. Given a total size constraint B, the problem is to choose a subset $S \subseteq \{1, 2, \dots, n\}$ so as to maximize the combined value, $\sum_{i \in S} v_i$, subject to the size constraint $\sum_{i \in S} s_i \leq B$. The input is assumed to satisfy $s_i \leq B$ for all $i \in \{1, \dots, n\}$.

(a) Consider the following *greedy algorithm*, GA.

1. For each i, compute the *value density* $\rho_i = v_i / s_i$.
2. Sort the remaining items in order of decreasing ρ_i.
3. Choose the longest initial segment of this sorted list that does not violate the size constraint.

Also consider the following *even more greedy algorithm*, EMGA.

1. Sort the items in order of decreasing v_i.
2. Choose the longest initial segment of this sorted list that does not violate the size constraint.

For each of these two algorithms, give a counterexample to demonstrate that its approximation ratio is not bounded above by any constant C. (Use different counterexamples for the two algorithms.)

Solution:

One counterexample for GA is an instance with two elements: $\{v_1 = 2, w_1 = 1\}$ and $\{v_2 = 2C + 1, w_2 = 2C + 1\}$. If the knapsack capacity is $2C + 1$, then GA will pick the first element and the second one wont fit. The value will be 2, whereas value $2C + 1$ will be obtained by choosing the second element instead.

One counterexample for $EMGA$ is an instance with $2C + 2$ elements: one element with $\{v_1 = 1, w_1 = 2C + 1\}$ and $2C + 1$ elements with $\{v_i = 1, w_i = 1\}$ each. If the knapsack capacity is $2C + 1$, then $EMGA$ will pick the first element and then all remaining ones wont fit. The value obtained will be 2, whereas value $2C + 1$ will be obtained by choosing all elements except the first one.

(b) Now consider the following algorithm: run GA and EMGA, look at the two solutions they produce, and pick the one with higher total value. Prove that this is a 2-approximation algorithm for the Knapsack Problem, i.e. it selects a set whose value is at least half of the value of the optimal set.

Solution:

Assume that there are no items whose weight is greater than W. Such items will never appear in any solution, and can be eliminated without changing the value of the optimum. We try to prove that $GA + EMGA \geq OPT$ from which follows that $\max\{GA, EMGA\} \geq \frac{1}{2} OPT$. Let $\{i_1, \dots, i_k\}$ denote the sequence of the items selected by GA and let i_{k+1} be the next element that GA would have selected, if it had not run out of space. $EMGA$ gets a value achieved by GA. On the other hand, the combined value of items$\{i_1, \ldots, i_k, i_{k+1}\}$ is greater than that of the optimal knapsack solution *Spot*, since *Spot* has a smaller combined size, and any elements of *Spot* that are not among $\{i_1, \ldots, i_{k+1}\}$ have a value density that is smaller than the least dense element of the set.

Let $S_0 = S_{OPT} \cup \{i_1, \ldots, i_{k+1}\}, S_1 = \{i_1, \ldots, i_{k+1}\}/S_{OPT}, S_2 = S_{OPT}/\{i_1, \ldots, i_{k+1}\}$, if the value density of each item i by $\rho_i = v_i/w_i$, and let $\rho^* = \rho_{i_{k+1}}$. According to greedy, $\rho_{i_j} \geq \rho$ for $j = 1, \ldots, k+1$ whereas $\rho_i \leq \rho^*$ for every i not belongs to $\{i_1, \ldots, i_{k+1}\}$. So $v_i \geq \rho^* w_i$ for every i belongs to S_1, while $v_i \leq \rho^* w_i$ for every i belongs to S_2. It shows that,

$$\sum_{j=1}^{k+1} v_{i_j} - \sum_{i \in S_{opt}} v_i = \sum_{i \in S_1} v_i - \sum_{i \in S_2} v_i \geq \sum_{i \in S_1} \rho^* w_i - \sum_{i \in S_2} \rho^* w_i = \rho^* (\sum_{i \in S_1} w_i - \sum_{i \in S_2} w_i)$$

$\sum_{j=1}^{k+1} v_{i_j} - \sum_{i \in S_{opt}} v_i$ is positive. Since $\sum_{i \in S_1} w_i > W - \sum_{i \in S_0} w_i \geq \sum_{i \in S_2} w_i$, $\rho^* (\sum_{i \in S_1} w_i - \sum_{i \in S_2} w_i)$ is positive and $v_{i_1} + \cdots + v_{i_{k+1}} > OPT$. Therefore, $GA + EMGA \geq OPT$ is proved.

(c) By combining part (b) with the dynamic programming algorithm for Knapsack presented in class, show that for every $\delta > 0$, there is a Knapsack algorithm with running time $O(n^2/\delta)$ whose approximation ratio is at most $1 + \delta$. [Recall that the algorithm presented in class had running time $O(n^3/\delta)$.] In your solution, it is not necessary to repeat the proof of correctness of the dynamic programming algorithm presented in class, i.e. you can assume the correctness of the pseudopolynomial algorithm that computes an exact solution to the knapsack problem in time $O(nV) = O(n \sum_{i=1}^{n} v_i)$, when the values v_i are integers.

Solution:

It is presented that dynamic programming algorithm for the knapsack problem that runs in $O(nv^*)$, where v is the upper bound of the value of the optimum solution. It is not a polynomial algorithm because the running time is proportional to v^*. Compute a set S^* obtained by running *GA* and *EMGA* to get two sets, and taking the one with the higher total value. Let $v^* = \sum_{i \in S^*} v_i$ and $k = \frac{\epsilon v^*}{(1+\epsilon)n}$, for every item i let $v_i' = v_i/k$. S is the output of the dynamic program on the instance defined by $\{(v_i', w_i) | i = 1, \ldots, n\}$ and *Spot* by $\{(v_i, w_i) | i = 1, \ldots, n\}$. Because it outputs the optimal solution for the knapsack instance on $\{(v_i', w_i) | i = 1, \ldots, n\}$. We get,

$$\sum_{i \in S} v_i' \geq \sum_{i \in OPT} v_i'$$

$$\sum_{i \in S} v_i \geq \sum_{i \in S} k v_i' \geq \sum_{i \in OPT} k v_i' \geq \sum_{i \in OPT} (v_i - k)$$

$$\sum_{i \in S} v_i \geq (\sum_{i \in OPT} v_i) - nk = (\sum_{i \in OPT} v_i) - \frac{\epsilon v^*}{(1+\epsilon)} \geq (1 - \frac{\epsilon}{(1+\epsilon)n}) \sum_{i \in S} v_i$$

Since $v^* \leq \sum_{i \in OPT} v_i$, $\sum_{i \in S} v_i \geq (\frac{1}{(1+\epsilon)n}) \sum_{i \in OPT} v_i$. Thus, S is a $(1+\epsilon)$ approximation of *Spot*. Based on the value of k, the running time is $O(n^2/\epsilon)$. The computation of the value density of the elements and running *GA* and *EMGA* for deciding the set S^* takes $O(n \log n)$. According to the algorithm of $O(n^3 s/\epsilon)$ presented in class, the overall ruling time of the algorithm is $O(n^2 s/\epsilon)$ where s denote the maximum number of bits in the binary representation of any numbers $v_i, w_i (i = 1, \ldots, n)$.