CS 4820, Spring 2019	Homework 8, Problem 2
Name: Kevin Klaben NetID: kek228	

(2) *(15 points)*
Recall that in the Knapsack Problem, one is given a set of items numbered $1, 2, \ldots, n$, such that the i-th item has value $v_i \geq 0$ and weight $w_i \geq 0$. Given a total weight constraint W, the problem is to choose a subset $S \subseteq \{1, 2, \ldots, n\}$ so as to maximize the combined value, $\sum_{i \in S} v_i$, subject to the weight constraint $\sum_{i \in S} w_i \leq W$.

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

1. Eliminate items whose weight w_i is greater than W.
2. For each remaining i, compute the *value density* $\rho_i = v_i / w_i$.
3. Sort the remaining items in order of decreasing ρ_i.
4. 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. Eliminate items whose weight w_i is greater than W.
2. Sort the remaining items in order of decreasing v_i.
3. 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.)

(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.

(c) By combining part (b) with the dynamic programming algorithm for Knapsack presented in class, show that for every $\varepsilon > 0$, there is a Knapsack algorithm with running time $O(n^2 s / \varepsilon)$ whose approximation ratio is at most $1 + \varepsilon$. Here, s denotes the maximum number of bits in the binary representation of any of the numbers v_i, w_i ($i = 1, \ldots, n$). [We will see an algorithm in class with running time $O(n^3 s / \varepsilon)$.] In your solution, it is not necessary to repeat the proof of correctness of the algorithm that will be presented in class.

(a) Consider an example where where we think that the approximation ratio C is bounded above by some constant c and thus C*Alg(knapsack)>Opt(knapsack)> Alg(knapsack) and thus, 1/C*Opt(knapsack) must be less than Alg(knapsack). We can devise an example that will break this equality no matter the value of C. The problem will have a total W that can be carried of C+1. Next there will be two items. The first is of weight 1 with value 2. The second item has weight equal to total weight W and has value of 2*W-1. Thus, the list ordered by density will be item 1 then item 2. Thus, item 1 is selected and that's all that will fit. However, the optimal solution is to select item 2. The value of the optimal solution is 2W-1 while the value of the GA solution is 2. Substituting into the inequality we get C*2>2*(C+1)-1 this gives C>C+1-1/2 which simplifies to C>C+1/2 which is not true. Thus, for all values of C the inequality will be broken

For EMGA we have some constant C which bound as follows C*Alg(knapsack)>Opt(knapsack)> Alg(knapsack). We can devise the following example. The overall weight W will be C+1=W. We will then have W+1 items. The first item will have weight W and value W+1 and the remaining W items will all have weight 1 and value W. Thus, the EMGA algorithm will sort the item such that the first item is first followed by the other W items. EMGA will first select item 1 and that will be all of the weight it can select. This will result in value W+1. The optimal solution is to select all of the other W items each of weight 1 with value W. This means the optimal value is W^2. Thus, substituting into the inequality we get C*(W+1)>=W^2 which simplifies by plugging C+1 in for W we get C*(C+1)>= $(C+1)^2$ which is never true for any values of C and thus we show that this bound C cannot exist.

(b)Claim: the higher value of the output of GA and EMGA results in this being a 2-approximation algorithm for the Knapsack problem.

Suppose we run GA selecting items 1 through i-1 until we reach some item i that is too big with the others to be added. It can be shown that the sum of the values of items 1 through k is greater than or equal to the optimal solution. As we know, the items were ordered by density and then added by decreasing density. In order for the optimal solution to have a greater value than that of the first i most dense items, it would have to contain denser items (which is not possible as these are the densest) or more less dense items. However, if they are less dense, they will take up more space more quickly at lesser values and thus would not generate a greater total value before hitting the limit. Thus we conclude that the sum of items 1 through i are greater than or equal to the optimal subset of items. Thus, either the first 1 through i-1 items or the ith item must be greater than or equal to 1/2OPT. In the case where it is the first 1 through i-1 items, this is the output of the GA algorithm. In the case where it is the ith item, we know that the EMGA algorithm outputs the largest value item with size less than W and thus the output EMGA must be greater than or equal to the weight of the ith item as it must either first select the ith item or an item with greater value than the ith item. Thus, as either GA or EMGA will output a solution which is at least 1/2 of OPT in all cases, it is clear that their maximum is a 2-approximation.

(c)Claim: for every $\varepsilon > 0$, there is a Knapsack algorithm with running time $O(n^2 s/\varepsilon)$ whose approximation ratio is at most $1 + \varepsilon$
Proof: IDK