CS 4820, Spring 2014 **Homework 10, Problem 1**
Name: Piyush Maheshwari
NetID: pm489
Collaborators: None

1. *(15 points in total)* 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 size $s_i \geq 0$. Given a total size budget B, 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 size constraint $\sum_{i \in S} s_i \leq B$.

1.a. *(5 points)* 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.)

1.b. *(5 points)* 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.

1.c. *(5 points)* Use part (b) to show that the the dynamic-programming based approximation algorithm for Knapsack presented in class can be modified so that it achieves an approximation ratio of at most $1 + \varepsilon$ with running time $O(n^2/\varepsilon)$. In your solution, it is not necessary to repeat the proof of correctness of the algorithm presented in class.

Solution

1.a. Counterexample for GA

Consider 2 elements with total budget size B. Let $B > 10$. The first element value $\sqrt{B}$ and weight $\sqrt{B}$. The 2^{nd} element has value B-1 and weight B. The greedy algorithm will pick element of weight $\sqrt{B}$ since the ratio v/s is 1 which is greater than the other element. So the total value would be $\sqrt{B}$. The optimal algorithm will pick element with weight B and the total value will be B-1.
The approximation ratio will be B-1/$\sqrt{B}$ which d grows as B grows. Hence it is not bounded by any constant.

Counterexample for EMGA

Take n = $\sqrt{B} + 1$. First $\sqrt{B}$ elements has value .99B and $\sqrt{B}$. The last element has value B and weight B. The EMGA will pick element with weight B since its value is the highest among all the elements. Then it won't be able to pick any more element since it has exhausted it's budget. So the total value would be B. The optimal algorithm will pick $\sqrt{B}$ of value .99B each. The total value would be .99B$\sqrt{B}$. The approximation ratio is .99B$\sqrt{B}$/B which grows as B grows. Hence it is unbounded by any constant.

1.b.

We know that GA $\leq$ OPT and EMGA $\leq$ OPT. Now we need to show that OPT $\leq$ 2 * max{ GA, EMGA}. This means that if we show that GA + EMGA $\geq$ OPT, this would imply that either GA or EMGA is greater than OPT/2. And since we pick the maximum of the two values this would imply that this algorithm is a 2 approximation of the knapsack problem.

Let G = max{ GA, EMGA} Now we know that GA picks up items in decreasing order of v/s. Suppose the item on which the weight sum exceeds the bound B be i and let its weight be w_i and its value be v_i. Consider another algorithm which picks is same as GA and picks up the element i as well. Call this algorithm G*. Obviously this algorithm does not satisfy the knapsack constraints.

Lemma 1. $v(G^*) > v(OPT)$

Proof. G* picks the items in the most optimal way possible. So for the bound B it picks up the highest value elements and since it is also picking up the element i, the sum of its value clearly is greater than OPT. □

Lemma 2. $v(G) > v(G^*)$

Proof. All values except the last picked up by G* is same as GA. We also know that the first value that EGMA picks up is v_{max}. So the sum of values picked up by EMGA is atleast v_{max}.

v(G*) = $v_1 + v_2 + ...v_i$
v(G*) = v(GA) + v_i
v(G*) < v(GA) + v_{max}
v(G*) < v(GA) + v(EMGA) □

Combining lemma 1 and lemma 2 we have v(G) $>$ v(OPT). This implies that G is a 2 approximation of the knapsack problem.

1.c.

Let the algorithm max{ GA, EMGA} be called G. We know that v(OPT) / v(G) $\leq$ 2 which means that v(OPT) $\leq$ 2 * v(G). This gives up a bound on the maximum value that v(OPT) can take. Now consider the dynamic programming algorithm presented in the class. The loop for value j goes from 1 to $\sum_{1}^{n} v_i$. This is because the maximum possible value that the optimal solution can take is the sum of values of all items. Since we know a tighter bound for v(OPT) we can replace that line in the loop by making j takes values from 1 to 2 * $\sim v(G)$ where $\sim v(G)$ represents running G on the scaled down values. The running time of the modified algorithm would become $O(n * 2* \sim v(G))$.

$O(n * 2* \sim v(G)) = O(n * 2 * v(G)/k)$
Now assume k = ε v(G) / n
$\implies O(n * 2* \sim v(G)) = O(n * 2 * v(G)/(\varepsilon * v(G)/n))$
$\implies O(n * 2* \sim v(G)) = O(n^2 * 2/\varepsilon)$

So running time = $O(2 * n^2/\varepsilon)$

Now we prove that this forms a (1 + ε) approximation for the knapsack problem. From lecture we know that v(G) $\leq$ v(S) + kn. Putting value of k as ε v(G) / n, we have
v(G) $\leq$ v(S) + ε v(G)
v(G) $\leq$ v(S) / (1 + ε)

Also from lecture we know that v(OPT) $\leq$ v(S) + kn
v(OPT) $\leq$ v(S) + ε * v(G)

Now using the inequality proved above

v(OPT) $\leq$ v(S) + ε * v(S) / (1 + ε)
(1 - ε) v(OPT) $\leq$ v(S)
v(OPT) / v(S) $\leq$ 1 / (1 - ε)

Now using the fact that 1/ (1 - ε) = 1 + $\varepsilon + \varepsilon^2 + \varepsilon^3$..., we have

v(OPT) / v(S) $\leq$ 1 + $\varepsilon + \varepsilon^2 + \varepsilon^3$ ...

Now if we pick an ε^* such that $\varepsilon^* = \varepsilon + \varepsilon^2 + \varepsilon^3$ ...
This is an valid approximation factor since when $\varepsilon < 1$ then $\varepsilon^* < 1$ and vica versa. Plugging in this value of ε^*, we have

v(OPT) / v(S) $\leq$ 1 + ε^*, which completes the proof.