Homework 2 sample solution

Due 09/03/15

August 27, 2015

1. Consider the *biggest sum* problem. The input to this problem is a list of integers and a target integer t. The output is a subset of the list whose sum is as close to t as possible without going over. (Ideally, the sum should equal t, but if this is not possible, we want to get as close as possible.) Prove that the algorithm below does *not* find the correct elements for every possible input.

```
   Input: values: an array of integers
   Input: n: the length of values
   Input: t: the target integer
   Output: a subset of values whose sum is as close to t as possible without
           going over
 1 Algorithm: GreedySum
 2 Sort values in decreasing order (i.e., max-first)
 3 sum = 0
 4 select = {}
 5 for i = 1 to n do
 6 |   if values[i] + sum < t then
 7 |   |   Add values[i] to select
 8 |   |   sum = sum + values[i]
 9 |   end
10 end
11 return select
```

Answer: Note: I had originally intended for values to be an array of positive values and the **if** condition in line 6 to use $\leq$ instead of $<$. As a result, there were more correct answers to this question than intended.

Proof. Consider the instance with values = (4, 3, 2), n = 3, and t = 5. First, GreedySum will sort the array values, though values is already sorted, so this will not have any effect. In the first iteration of the `for` loop, sum = 0, so GreedySum will add values[1] = 4 to select, and sum will become 4. In the second and third iterations of the `for` loop, sum + values[i] will be 7 and 6, respectively, so it will not add either 3 or 2 to

select, and it will return select = {4}. However, this subset of values only has a sum of 4, while the subset {3, 2} has a sum of 5, which is closer to the target t. Thus, GreedySum returns an incorrect solution for this problem instance, so it is not correct. □