York University EECS 4115/5115 November 6, 2020

Homework Assignment #6
Due: Friday, November 13, 2020 at 5:00 p.m.

1. Consider the following optimization problem. Given a set of positive integer values $v_1, v_2, \ldots, v_n$ and a positive integer threshold t, find a subset of the values whose total is as close to t as possible without exceeding t.

(a) Give an example of an application where you might want to solve this problem.

(b) Show that if you had an algorithm to solve this problem that runs in polynomial time (in the size of the input), then P=NP.

(c) Consider the following greedy algorithm.

```
preprocess the input to discard any elements that are bigger than t (since they cannot be used)
sum ← 0
i ← 1
while i ≤ n and sum + v_i ≤ t
      // invariant: sum = v_1 + v_2 + ··· + v_{i−1} ≤ t
      sum ← sum + v_i
      i ← i + 1
end while
if i = n + 1 then return {v_1, v_2, ..., v_n}
else if v_i > sum then return {v_i}
else return {v_1, v_2, ..., v_{i−1}}
```

Prove that, for all inputs, the sum of the elements returned by this algorithm is at least $\frac{1}{2}$ of the optimal sum.

(d) Give an example input where the sum of the elements returned by the algorithm in part (c) is less than 0.51 times the optimal sum.

(e) Consider the following polynomial time algorithm. Classify the values $v_1, v_2, \ldots, v_n$ into two buckets:

- small values that are at most $t/3$, and
- large values that are greater than $t/3$.

Try all combinations of 2 or fewer large elements to see which gives the largest sum s. (If there are no large elements, then s will be 0.) Then, use the algorithm from part (c) to find a subset of the small elements whose sum comes as close as possible to $t - s$ without exceeding $t - s$.

- Give a constant c such that, for all inputs, the sum of the elements returned by this algorithm is at least c times the optimal sum.
- Give a constant c' and an example input where the output of the algorithm is less than c' times the optimal sum.

Your goal is to make c and c' as close to each other as possible.

(f) **For EECS5115 students only**: Modify the algorithm in part (e) to achieve an approximation factor of $1 - \varepsilon$ for any ε. Give an upper bound on the running time of your algorithm in terms of n and $1/\varepsilon$, assuming that arithmetic operations on values can be done in $O(1)$ time.