CMPSCI 311: Introduction to Algorithms **Spring 2018**

Discussion 10

4/26/2018 Names:

Note: *LaTeX template courtesy of UC Berkeley EECS dept.*

Instructions. You will be randomly assigned groups to work on these problems in discussion section. List your group members on your worksheet and turn it in at the end of class. Write first and last names. Each group member should turn in their own paper.

1. **Subset Sum.** Recall the subset sum or knapsack problem, where you have a collection of items with different positive weights and you want to find the subset of those weights that get as close as possible to a target weight W without exceeding it.

   You attempt to approximately solve this problem by sorting the weights and then adding weights in order from heaviest to lightest, never exceeding W.

   (a) What is the maximum amount of error that this approximation algorithm can have?

   (b) Loosely speaking, when would this greedy approximation be faster to perform than the accurate dynamic programming solution demonstrated in class?

2. **Vertex Cover.** You want to find the minimum number of vertexes needed to make a vertex cover in a graph G. This can be solved using brute force as demonstrated in the algorithm below.

   (a) What is the run time complexity of this algorithm?

   (b) If NP != P, does there exist a polynomial time algorithm to solve this problem accurately?

   (c) Design an approximation algorithm that runs in polynomial time. Your algorithm should never underestimate the size of the minimum vertex cover. Hint: One way of approaching this problem is to generate a vertex cover.

Algorithm 1 `Brute-Force-Minimum-Vertex-Cover`(G)

```
cover-size = 0
while TRUE do
  for vertex-set ∈ All-Subsets (G.nodes) do
    if |vertex-set| == cover-size then
      uncovered-edge-count = 0
      for edge ∈ G.edges do
        if ¬(edge.node1 ∈ vertex-set) ∧ ¬(edge.node2 ∈ vertex-set) then
          uncovered-edge-count = uncovered-edge-count + 1
        end if
      end for
      if uncovered-edge-count == 0 then
        return cover-size
      end if
    end if
  end for
  cover-size = cover-size + 1
end while
```

(d) Find a case where your approximation algorithm returns an inaccurate result.

3. **Reduction Review.**

(a) Prove that 3-COLORING $\leq_p$ SAT.