

You can work in groups of up to 3. Please let us know, for each question, if you have seen the question before. You will be graded out of  $\min(100, \text{max possible})$  points for this assignment. Unless otherwise specified,  $n$  is the number of vertices of a graph. Please submit your solution in ALL of the following formats by email to `virgi+algo@cs.cmu.edu` and `chengwen+gradalg@cs.cmu.edu`: ps, pdf and tex. Please use ‘Submission: names’ in the subject line. If you use the result of any paper, please understand and rephrase its content.

**WARNING:** This week’s homework is not trivial. One or more of the problems may be open. We will give partial credit for any good ideas.

1. Here is an approximation algorithm for Vertex Cover:

MAXDEGREERATIO( $G$ ):

For every vertex  $v$  let  $d(v)$  be its degree.

For every edge  $(u, v) \in E$  let  $w(u, v) = \max\{\frac{d(u)}{d(v)}, \frac{d(v)}{d(u)}\}$ .

While  $E$  is nonempty do

    Let  $e = (u, v)$  be the edge with maximum weight.

    Let  $v$  be the end point with  $d(v) \geq d(u)$

    put  $v$  in  $S$ .

    Remove  $v$  and all incident edges from the graph.

    Update the degrees of all neighbors of  $v$ , and the weights of all their incident edges.

In the Vertex Cover approximation algorithm we saw in class we picked both endpoints for each chosen edge to be in the vertex cover, because we did not know how to choose between them. Intuitively, the algorithm above attempts to pick the edge with the two most different end points, and then to place only the higher degree end point in the vertex cover.

For this problem, either show that the algorithm gives a constant factor approximation to the optimal vertex cover, for some constant  $k > 1$  [40pts], or show that no such constant exists, *i.e.* there exists a class of instances for which the algorithm returns solutions whose approximation ratios grow (and do not converge) as the instance size grows. [40pts].

Recall, an algorithm  $A$  is a  $k$ -approximation to vertex cover if for any instance the solution  $S_A$  the algorithm returns is of size at most  $k$  times the optimal solution  $S_{opt}$ :

$$\frac{|S_A|}{|S_{opt}|} \leq k.$$

2. [10pts] Given an integer  $N > 1$  the Factoring problem asks for two integers  $p, q > 1$  so that  $N = pq$ . Consider the following algorithm for Factoring:

For all integers  $i: 2 \leq i \leq \sqrt{N}$   
if  $i$  divides  $N$ , return  $i$  and  $N/i$ .

Return **prime**.

Is this a polynomial time algorithm for Factoring? Why or why not?

3. Given a graph  $G$ , the  $k$ -Bounded Spanning Tree ( $k$ -BST) problem for a fixed integer  $k \geq 2$  is as follows

INSTANCE: Graph  $G$

QUESTION: Does  $G$  have a spanning tree with exactly  $k$  leaves?

Show that  $k$ -BST is NP-complete for every  $k$  or find a polynomial time algorithm.

5 pts for a correct NP-completeness proof for some fixed  $k$

15pts for a correct NP-completeness proof for all  $k$

If there is a polynomial time algorithm:

40 pts for  $O(kn)$

30 pts for  $O(n \log n)$

20 pts for  $O(k^3 n^2)$

10 pts for polynomial time algorithm.

4. Suppose someone gives you a graph  $G$  and promises you that  $G$  contains a clique of size at least  $\frac{n}{2}$ . Given this information, can you find a clique of size  $\frac{n}{100}$  in polynomial time? Can you show that this is NP-hard?

40 pts for proving this is NP-hard

and/or

40 pts for  $O(n)$

30 pts for  $O(n^3)$

20 pts for polynomial time algorithm.