Spring 2018 **CS 513: # 9 Coping with NP-Completeness** Ghashami, Kleindessner

Due by March 30, 23:59 pm.

Exercise 1 (5 + 15 points)

a) Give an example of a graph for which Approx-Vertex-Cover always returns a vertex cover that comprises twice as many vertices as an optimal vertex cover.

b) Provide an infinite family of graphs for which Approx-Vertex-Cover always returns a vertex cover that comprises twice as many vertices as an optimal vertex cover. Prove your claim.

Exercise 2 (35 points)

Show that the following greedy strategy yields a polynomial-time logarithmic-factor approximation algorithm for the vertex-cover problem: repeatedly select a vertex of highest degree and remove all of its incident edges, until there are no edges left.

Exercise 3 (20 + 25 points)

a) Devise a branch-and-bound algorithm for the set-cover problem. Provide the pseudocode of your algorithm and show that you are actually computing a lower bound on the cost of partial solutions.

b) Implement your algorithm in a language of your choice and apply it to the following instance of the set-cover problem:

$$
\begin{aligned}
X =& \{1, \ldots, 20\}, \\
\mathcal{F} =& \big\{\{20\}, \{16\}, \{19, 13, 6, 8, 18\}, \{18\}, \{11, 3, 16\}, \{8, 10, 20, 2\}, \{3, 4\}, \{9, 1\}, \{10, 18, 8, 16, 15\}, \\
& \{2, 3, 17\}, \{12, 3, 4, 18, 11\}, \{10, 11, 6, 13\}, \{14\}, \{8\}, \{16, 2, 17, 14\}, \{9, 20, 6\}, \{11, 16, 15\}, \\
& \{8, 16, 10, 6\}, \{20, 10, 7, 11, 14\}, \{4, 5\}, \{7, 18, 8\}, \{19\}\big\}
\end{aligned}
$$

The membership-table of $\mathcal{F}$ (an indicator matrix $\in \{0,1\}^{|\mathcal{F}| \times |\mathcal{X}|}$) is provided in the file `SetCoverInstance.txt` on the course website.

Provide answers to the following questions:

(a) Which cover does your algorithm find? What is its size?

(b) How often do you compute a lower bound on the cost of a partial solution? How often do you examine a complete solution? You should compare these numbers to $2^{|\mathcal{F}|} = 2^{22} = 4194304$, which is the number of complete solutions examined by an exhaustive search.

(c) How long does your algorithm run on this instance? Implement an exhaustive search and report its running time too.

(d) Which cover does your exhaustive search find? What is its size?