Block 1

In this exercise, algorithms for the solution of the following graph problem

implemented.

Vertex cover

Input: A graph G = (V, E) and an integer k.

Question: Is there a set S ⊆ V (called a vertex cover) with | S | ≤ k such that

every edge e ∈ E has at least one end point in S?

A graph G = (V, E) consists of a set of nodes V and a set

from edges E. Edges can exist between pairs of different nodes; a

Edge between nodes u and v is mathematically the two-element set {u, v}. It

in the following it is about finding the smallest possible number k for a given graph ¨ G

find such that (G, k) is a yes instance for Vertex Cover. Let's look at the

following graph.

(Graph)

In this case the searched number is k = 2, since every edge has an end point in the set S = {2, 3} and not a one-element set {x} ⊆ {1, 2, 3, 4, 5} with this property exists. In the following it is assumed that G has no edge of the form {v, v} and that each edge occurs at most once.

Task 1

In order to solve vertex covers, it is first necessary to implement a graph data structure. Create a MyGraph class that will make the Graph interface

Graph.java implemented.

Task 2

In the next step, a graph must be read from a file. For simplification

let us assume that the graph is given in row format. In this format

each line of exactly one edge from E;

the edge is given by the two node names,

which are separated from each other by whitespaces.

For example, describes the file with the following content

xy7 REW

xy7 CGD

CGD REW

a graph consisting of exactly three nodes and exactly three edges. In other words, the graph described in this file is a triangle.

The line "xy7 REW “ means

for example that the graph has an edge between node xy7 and REW.

Extend your MyGraph class with a public MyGraph (String

filename), which reads in a graph in line format. You can find the data records in ILIAS.

Task 3

This exercise is about solving vertex covers. To do this, create a class

SearchTree with an inner class Instance. The class Instance represents a

Instance (G, k) of Vertex Cover.

The SearchTree class is to be solved via a method ¨ private boolean solve (Instance i)

feature. This method should test whether the graph ¨ G of the instance i over a ¨

Vertex cover of size k has whether (¨ G, k) is a yes instance. The algorithm

works as follows:

(Algorithm )

Here G - u is the graph that can be obtained from G by deleting u and all an u

the adjacent edges.

Furthermore, this class should have a method public int solve (Graph g),

which outputs the smallest k such that the graph G has a vertex cover of size k. In a

Method should be the number of nodes | V |, the number of edges | E |, the quantity k

and output the runtime.

Task 4

This exercise deals with the implementation of reduction rules for Vertex Cover. The idea is to simplify an instance (G, k) of Vertex Cover without affecting the correctness. This means that nodes are to be identified which are in the

Vertex Cover must be recorded or deleted.

This leads to a Speed ​​up of the algorithm.

In the following, three reduction rules for Vertex Cover are presented. Expand

Your implementation of the SearchTree class so that these rules come before everyone

Branch step in (G − u, k −1) and (G − v, k −1) can be applied. To implement

To do this, follow the three rules below.

1) A node v ∈ V (G) is said to be isolated if deg (v) = 0.

Implement a method removeSingletons for a given

Graph G deletes all isolated nodes.

2) The second reduction rule is about deleting nodes of degree one. As long as G has a vertex v with deg (v) = 1: Let u be the neighbor of v. Delete u from G and

reduce k by one.

Implement a method removeDegOne that corresponds to the above reduction rule.

3) A vertex v ∈ V (G) has high degree if deg (v)> k. Delete v from G and reduce k

at one.

Implement a method removeHighDeg for a given graph ¨ G

determines all high-grade nodes and deletes them.