Project **Winter 2022/23**
Algorithm Engineering for hard problems **TU Berlin**
Nichterlein/Boehmer **October 17, 2022**

1st Exercise Sheet

Vertex Cover I

Deadlines
Solver: October 31 **Handout:** October 01 **Presentation:** October 02

The task is to implement algorithms to solve the Vertex Cover problem introduced in the lecture.

Vertex Cover
Input: An undirected graph $G = (V, E)$.
Output: Find a minimum size vertex cover, that is, a vertex subset $V' \subseteq V$ such that for each edge $e \in E$ at least one endpoint is in V' (i. e. $V' \cap e \neq \emptyset$).

The goal of the 1st exercise sheet is to implement the simple search tree algorithm from the lecture.

All implementations *have* to compile / run on Linux on one of our machines:

`abaXX.akt.tu-berlin.de` with `XX` $\in \{01, 02, \ldots, 10, 11\}$.

We provide accounts once the participants are fixed. Java, C++, Python, etc. are installed. Please ask in the discussion forum in the ISIS course if you want to have further software or packages installed.

ISIS-website

All necessary material will be provided through the ISIS-website of the course. Address:

https://isis.tu-berlin.de/course/view.php?id=30369

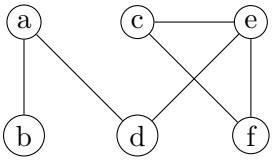
Input and output format

We use the following format for graphs.

- Whitespaces at the beginning and end of every line are ignored.
- Text after # until the end of the line is ignored (treated as comment).
- Empty lines are ignored.
- All other lines define an edge by stating start and end vertex of the edge separated by a whitespace. Names of vertices may contain an arbitrary number of letters, numbers, and underscores.

Example:

Input graph:

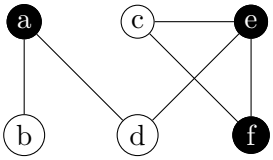


Input file:

```
# 6 vertices
# 6 edges
a b
a d
c e
c f
d e
e f
```

The standard input for your program is a graph in the above described format. (In case the input does not match the format, the program should abort with a corresponding error message.) The output of the program is a minimum cardinality vertex cover. Every vertex should be written in one line (in an arbitrary order). For the above example this would be:

Vertex Cover (black vertices):



Output (only the solution):

```
a
e
f
```

Verification

We provide optimal solutions for most of the test instances in a separate `.solution` file.

Benchmarks and presentation

Test instances: The test instances are available for download:

`http://fpt.akt.tu-berlin.de/alg-eng-data/vc-data-students.zip`

The running time will be evaluated on those instances. There are three types of instances: `random`, `medium-sized`, and `social networks`.

Within the zip-file you will also find a bash script `benchmark-fast.sh`. It allows easy execution of your program on all the test instances under the following requirements: The computation of an instance is aborted if the program did not find a solution after **5 min**. After **10** unsolved instances, the script skips the remaining instances in the current subfolder. (These will be also the settings we test your submissions.)

The bash script also provides these information in an csv-file. We refer to README.md in the zip-file for the usage and options of the bash script.

Handout and presentation. Submit a handout of your evaluation and present your findings. Your handout should give a short description of what you implemented as well as comparisons of the theoretical and practical running times (depending on the cost and the number of vertices). In later exercises it should also give a comparison to the results of the last exercise.

You can find a LaTeX template for the handout in the ISIS course. There should be **one** page of double-column written text (including references) as well as **one** additional page containing some nice figures (find templates for these in ISIS as well).

Your presentation should last at most 10 minutes and answer the following questions:

- How did you implement? What data structures and (possibly) other clever ideas did you use? In the first sheet also: Which programming language and packages did you use?
- How does the running time depend on the number of vertices of the input graph?
- How does the running time depend on the minimum vertex cover size?
- Is the theoretical running time bound reflected in the empirical running time? If not, what could be possible reasons?
- How do the answers of the above questions depend on the type of the instances, that is, are there differences between the synthetic and the real world (social network) instances? If yes, what could be possible reasons?

We recommend to use graphical representations of the data (e.g., diagrams) to answer these questions. (There is an example presentation with source code in the ISIS course and an example for making diagrams in LaTeX.) Please keep in mind that every member of your team should be able to answer the above questions, and every member of the team should have a fair share of presentation time during the five presentations.