5DV149 LP4 — Assignment 4

Graphs
Submission v1.0

Niclas Borlin (niclas.borlin@cs.umu.se)

2022-05-19

Contents

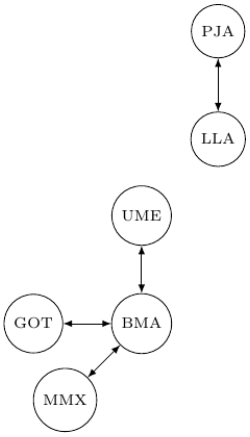

Figure 1: Graph illustration corresponding to the file `airmap1.map`, png version.

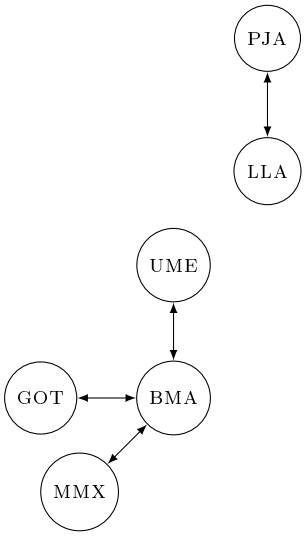

Figure 2: Graph illustration corresponding to the file `airmap1.map`, tikz version.

0 Version history

v1.0 2022-05-19 First submission.[1]

1 Introduction

The target audience is someone with basic understanding in Computing Science, but not specifically graphs nor the application, e.g., a future colleague at your work who did not take part in this particular project. Describe the application and how it connects to graphs. You may include a figure, e.g. as in figures 1 and 2.

[1] If this is a resubmission, include a list of changes with respect to the previous submission.

2 User guide

2.1 Compilation

Describe exactly how the reader might compile the code, assuming he/she has access to the source code, e.g.

```
gcc -o is_connected -std=c99 -Wall -I .../path/to/include is_connected.c graph.c
  .../path/to/other/files.c
```

2.2 File format

Describe the file format, including an example, e.g. (NOTE: you MUST change this example!)

```
# This is a text file
2
UME BMA # Umea-Bromma
BMA UME # Bromma-Umea
```

2.3 Test runs

Describe a complete test run, i.e., how the program is started, input and output. Screen dumps can be useful here, but be sure to trim them to the terminal window where the program is run.

3 System description

3.1 Data structures

3.1.1 Graph

The graph data structure is central to this assignment. Describe how you have designed and implemented the data structure. What information is stored and where? If you use a `struct`, you should probably describe it and its fields. How is the allocate/deallocate responsibility handled. What help data structures have you used? How have you defined equality between nodes? How do you handle (avoid?) node duplicate in the graph?

Describe each function in the interface to the data type `Graph`. Each implemented function should be described in some detail, e.g., input, output, actions and any side effects. The interface should be organised to be easy to read, e.g., in a table or a bullet list. It is ok to include the function declaration, e.g.,[2]

```
graph *graph_insert_edge(graph *g, node *n1, node *n2);
```

but in the text you should refer to *the graph* ***g***, *the node* ***n1***, etc. Never use `*` in the text. Instead, if it is necessary, write "pointer to...".

You may group and the unimplemented functions separately. The unimplemented function do not need to be described nor commented, only listed.

3.1.2 Graph #2

If you have implemented two different graphs, describe the second implementation here.

3.1.3 Other data structures

For each data type from the code base that you have used, e.g., `queue`, describe it in a separate subsection. In each section, describe the datatype briefly in one or a few sentences and then list all functions in the user interface. Only make references to *published* information about the data type. That includes what is known from the header files but not the source.

[2] If you use latex and minted for this kind of color-coding of source code you must probably need to compile the .tex file with `pdflatex -shell-escape file.tex`.

3.2 Algorithms

All algorithms must be described in *psedocode*. You may use text to summarize the algorithm, but each step must be in some form of bulleted or enumerated list. Remember to use variable names, etc., to make the algorithm more specific, e.g., "the current node n", etc.

3.2.1 Parsing the text file and constructing the graph

Describe your algorithm for parsing the text file and constructing the graph. Depending on your implementation, this may be one algorithm or two separate ones. You do not have to describe the parsing of each line in detail.

3.2.2 `find_path`

Describe your version of the breadth-first-algorithm that you have implemented in `find_path`.

4 Reflections

4.1 Work distribution

If you worked in groups, how has the work been distributed. How have you made sure that everyone understands each part, including parts that have been the resposibility of others?

4.2 Reflections

Reflect on the assignment! Did you find anything fun, challenging, surprising, frustrating, rewarding, etc. If you submit for a group, you may write one reflection for each group member, or one for the whole group.

4.3 Future work (optional)

Did you think of anything interesting to try that you did not have time to include? If yes, this is the place to present it.

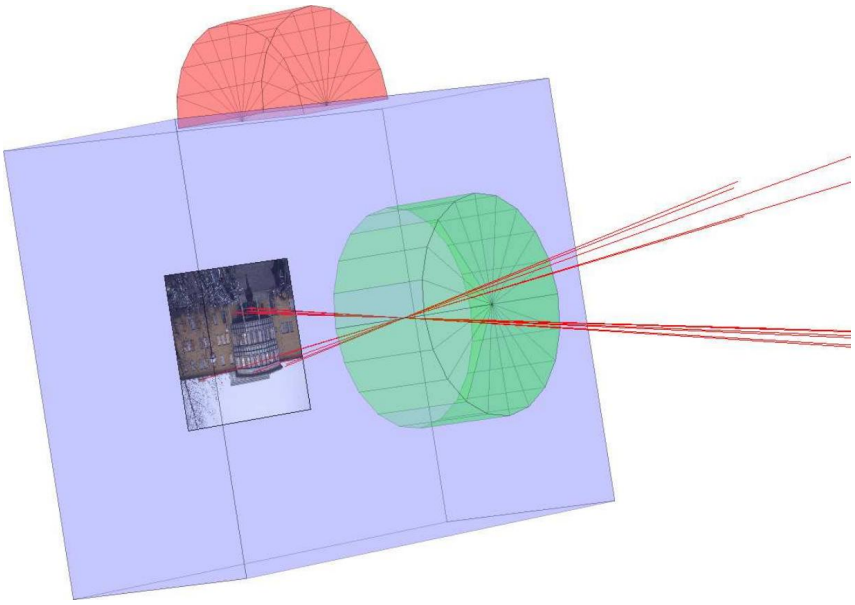

Figure 3: A figure/image caption should provide sufficient information to make the figure/image as self-explanatory as possible. The caption should be placed under the figure. The latex source shows how to include an image file into a latex document.

A Useful LaTeX examples

Stuff that may be important to some readers, but not all, may be deferred to an appendix. The same is true for lengthy material that would disrupt the flow of the document if placed immediately where it is first referenced. Examples include code listings, file formats, standards, complete tables of all experiments, etc.

A.1 Figures

Figure 3 shows an example of a figure. Exactly *where* (at top or bottom of a page, on a separate page, or "here" in the text) to put figures/tables is a matter of style. The author of this document is of the opinion that "here" should be avoided at all cost. It might seem advantageous to have the figure close to the text that describes it. However, the figure/table should be as self-contained as possible. In general, it should be possible to read and understand the body text *without* having to look at the figure. Thus, if you are forced to write the body text and present the figure such that they will work independently, your report and writing style will benefit.

As the placement of figures and other floats in LaTeX may shift due to changes in text, you are encouraged to leave the fine-tuning of image placement **until your document is complete**.

A.2 Tables

Tables are often used to present tabulated (no sh*t, sherlock?) data about the experiment setup, test data, etc., or with results of the experiments. In the former case, the body text would typically describe what is common with the data sets and then refer to a table with detailed information. In the latter case, do not discuss the structure of the table in the body text! That would just confuse the reader. Such information belongs to the caption. In general, do not refer to the table such that the reader cannot continue without inspecting the table. Instead, summarize enough of the content of the table to allow the reader to continue to the next paragraph.

Data that can better be summarized in the body text should so appear, e.g., "The execution time for experiment x was below 2ms. The other execution times are given in Table x."

Table 1: A table caption should provide information that helps the reader to understand what data is in the table. Some additional information, e.g., units can also be part of the caption. A table caption should be placed above the table proper. Use as few borders in the table as possible! For instance, adding left and right borders to the table below would make it harder to read.

Table type	Lookup speed (ms)
MTFTable	x
Arraytable	y
DListTable	z

In all cases, consider the number of significant digits! Do not put a gazillion decimals in your tables just because your code spits it out! Make the table as easy to read as possible. An example of a stub of a results table is given in Table 1.