Project Description - Software Engineering SS18

Design a software for visualizing and tracking algorithms applied on graphs

Marc Hellmuth and Nikolai Nøjgaard

JProf. Biomathematics and Computer Science
Inst. of Mathematics and Computer Science
University of Greifswald

1 Introduction

Graphs are mathematical structures used to model pairwise relations between objects and play a central role in many scientific areas.

A graph is a tuple $G = (V, E)$ that consists of a vertex set V and an edge set $E \subseteq V \times V$. By slight abuse of notation, if $(x, y) \in E$ implies that $(y, x) \in E$, then the graph is often called *undirected* and the notation $\{x, y\} \in E$ is often used. A graph may have labels, weights, colors or other objects associated to its vertices or edges. See Fig. 1 for an example.

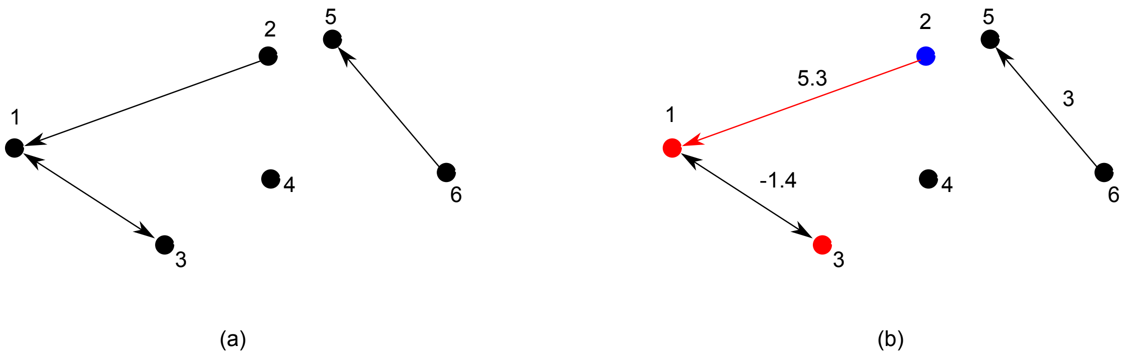

Figure 1: (a) A vertex-labeled graph $G = (V, E)$ with vertex set $V = \{1, 2, \ldots, 6\}$ and edge set $E = \{(1, 3), (3, 1), (2, 1), (6, 5)\}$. (b) The graph $G = (V, E)$ where some vertices and edges obtained colors or weights.

Many graph algorithms that process a graph exists. These algorithms are usually applied to understand the structure of the graph in more detail or to answer specific questions related to the problem domain that the graph models. These algorithm usually compute a "final" solution for the considered problem. However, such algorithms often consists of several "intermediate steps" that need to be performed to obtain the final solution. To easily understand the behavior of such algorithms a program is desired that visualizes the intermediated steps applied on some automatically drawn input graph G. For a very advanced example see e.g. `https://visualgo.net/`.

2 Basic Requirements on the Software

You will work in a group of 2-3 people. The details about the software requirements are discussed in the lecture and tutorial. For a first overview look at the provided wiki-links.

On a top-level the JAVA-software and should . . .

(1) . . . be able to read a graph[1] (with or without labeling, colors, weights etc.pp.) that is given via text-file or drawn via some GUI.

(2) . . . be able to properly visualize a graph.

To this end, there are two ways:

- (2a) Either you use a JAVA library **(1 point)**
- (2b) You visualize the graph with a force-directed layout algorithm[2] [3] [4] and permit the user to re-adjust the drawing (e.g. move vertices, stretch edges, remove/add vertices etc.pp.) **(4 points)**

(3A) . . . provide *each* of the following algorithms:

- Breadth-First Search[5] and Depth-first search[6]
- Topological Sort[7]
- Find a minimum spanning tree[8]
- Find a maximal matching[9]

(3B) . . . provide *some* of the following algorithms:

- Test Bipartite Graph[10] **(1 point)**
- Find a cycle basis of a graph[11] **(1 point)**
- Find shortest path between two given vertices[12] **(1 point)**
- Find strongly connected components[13] **(1 points)**
- Implement the BUILD algorithm to test consistency of triples for phylogenetic trees[14] [15] [16]. Here both, the "auxiliary" graph and the resulting tree should be displayed in each step. **(4 points)**

(4) . . . visualize the *important* intermediate steps of the user-chosen algorithm with the drawn graph.

(5) . . . should be designed such that new requirements and specifications to the software are easily incorporated.

You will possibly be asked to provide a special add-on to your software after the 1st release!

3 Requirements to pass the course

1. *Active* Participation
2. Self-study (within the group): Graphs and the algorithms. Share the work!
3. Presentations:
   (a) 02.05. Definition of Project (around 10-20 min per group)
   Which of the items will you realize?
   What are the Requirements? (eg. User Stories)
   Some details to the chosen algorithms.
   (b) ∼29.05. State of the Project *"1st Release"* (around 10-20 min per group)
   What has been done and what has to be done?
   (c) July (exact date to be specified)
   Final Presentation of the project (20-30 min per group)

4. Provide the Software (delivered as tar.gz).

   While each Item (1)-(5) in Section 2 is required, there are additional constraints for Item (2) and (3B):

   Each correct implementation of the algorithms in Item (2) and (3B) is rewarded with the points at the right of each item.

   In total the sum of the points should be greater or equal to 8.

5. Provide the tests that are used for establishing the correctness and reliability of the software + a documentation (PDF) about the test sets and the reason why you have chosen these tests. (specified in lecture)

6. Provide "Software Requirements Specification" as PDF (specified in lecture)

7. Provide "Documentation of the Software" (specified in lecture)

Notes

[1] `https://en.wikipedia.org/wiki/Graph_(discrete_mathematics)`

[2] `https://en.wikipedia.org/wiki/Force-directed_graph_drawing`

[3] BOOK: Handbook of Graph Drawing and Visualization (Chp. 12), Roberto Tamassia (Editor), CRC Press, 2013 – *available at homepage*

[4] Simple Algorithms for Network Visualization: A Tutorial (Section 1), Michael J. McGuffin, Tsinghua Science and Technology 17(4), 2012 – *available at homepage*

[5] `https://en.wikipedia.org/wiki/Breadth-first_search`

[6] `https://en.wikipedia.org/wiki/Depth-first_search`

[7] `https://en.wikipedia.org/wiki/Topological_sorting`

[8] `https://en.wikipedia.org/wiki/Minimum_spanning_tree`

[9] `https://en.wikipedia.org/wiki/Matching_(graph_theory)`

[10] `https://en.wikipedia.org/wiki/Bipartite_graph`

[11] `https://en.wikipedia.org/wiki/Cycle_basis`

[12] `https://en.wikipedia.org/wiki/Shortest_path_problem`

[13] `https://en.wikipedia.org/wiki/Strongly_connected_component`

[14] `https://math-inf.uni-greifswald.de/fileadmin/uni-greifswald/fakultaet/mnf/mathinf/hellmuth/Teaching/SoftwareEngineering18/BUILD.pdf`

[15] BOOK: Phylogenetics (Section 6.4), Semple and Steel, Oxford University Press, 2009 – *available at homepage*

[16] Inferring a Tree from Lowest Common Ancestors with an Application to the Optimization of Relational Expressions, Aho et al. SIAM J COMPUT, 10(3), 405-421, 1981 – *available at homepage*