UNIVERSITY OF COPENHAGEN
Department Of Computer Science DIKU

BACHELOR PROJECT

A VISUALIZATION TOOL FOR LEARNING ALGORITHMS AND DATASTRUCTURES

SYNOPSIS

2016 BLOCK 3-4

VALDAS ZABULIONIS

RUNE FRANCH PEDERSEN

AMR EL SAYED

March 14, 2016

A Visualization Tool for Learning Algorithms and Datastructures

DIKU - Bachelor Project

Studens Names	:	Valdas Zabulionis - LHN100
	:	Rune Franch Pedersen - VQR730
	:	Amr El Sayed - VWJ159
University	:	University of Copenhagen
Institution	:	Department of Computer Science DIKU
Supervisor	:	Oleksandr Shturmov
Period	:	Block 3 - block 4
Year	:	2016
Pages	:	7

Certificate

This is to certify that the work contained in the thesis entitled "A Visualization Tool for Learning Algorithms and Datastructures" by Valdas Zabulionis, Rune Franch Pedersen and Amr El Sayed, has been carried out under our supervision and that this work has not been submitted elsewhere.

Supervisor
Oleksandr Shturmov
oleks@di.ku.dk
Department of Computer Science DIKU
University of Copenhagen

Contents

Project Title 4

Problem definition 4

Background 4

Project limitations 5

Time schedule and work tasks 6

References 7

Project Title

A Visualization Tool for Learning Algorithms and Datastructures

Problem definition

For students, who are relatively unfamiliar with algorithms and programming, getting a comprehensive understanding of algorithms can be very difficult.
In this project we will design a visual tool to help student create their own basic algorithms and do a step-by-step run-through of them. This is intended to help students learn the basics of algorithms.
The primary focus is an easy to use tool, which will use visual building blocks, that are predefined statements providing function body, statements, condition expressions, cases, loops, etc.. They can be easily used by the students to create an algorithm. The tool would also include a built-in function which will provide a detailed step-by-step description of each step in the implemented algorithm.

Background

The reason for us choosing this project is that we would like to help new computer science students to learn how algorithms work. Many students have a hard time learning about algorithms and datastructures, and this can be seen by the number of students that fail the introductory algorithm course or get a low grade. We believe that algorithms are a very important part of Computer Science, thus it is important that students learn the basics well from the very beginning. We think that creating a teaching tool to help students visualize how algorithms work would help improve their learning experience, thus improving their studies, and quality of future computer scientists. At the same time we think that creating such a program for our final project is a good way to use what we have learned in our studies. By combining algorithms and datastructures, a bit of compiler design, human-computer interaction, and system development methodologies this becomes a project that will be of great interest as a teaching tool.

We have decided that the program would let the user create algorithms using pseudocode, which would be built up from our pre-defined building blocks. Our program is not meant to be a parser, but will instead provide tools to use our own abstract syntax tree. The users would have the ability to run the pseudocode, to see what happens when an algorithm is run step-by-step in a run-time state. We could avoid infinite loops by limiting the run time to 200 or less iterations. The project would be scalable, so that it could be easily upgraded in the future - this would be made possible by object oriented programming, where most elements are objects that are easily manipulated, program components are relatively easy to manage, and that they are gathered into a single hierarchy. That way it would be simple to implement new buttons or animations, add new algorithms, or upgrade parts of the backend.

We think creating a system that would use several parts to visualize and run an algorithm step-by-step would be a fun challenge. We have decided to implement the back-end using Python, and front-end would be implemented using JavaScript/HTML/CSS. A challenge we would have to overcome is how to get the back-end work together with the visualization methods on the front-end.
We would have to use some of what we have learned in Compiler Design to implement building blocks for our program to parse the information input from the user.
It would be a good idea to make the front-end intuitive and easy to use, using some of the

methodologies that we have learned by studying human-computer interaction. We will have to implement simple and easy to understand visuals, and helpful error messages if something were to go wrong.

Project scope

We will only implement a few of the algorithms shown in the algorithm book((Cormen, 2009)), with an emphasis on making it easy to support further extensions.
We will not estimate the running of the algorithms that the students implement, although it could be a part of the project. We have decided that we would leave the estimation of running time to the students to find on their own, since it is an important part of learning datastructures and algorithms.
It is not intended that the program executes arbitrary student code.
The primary documentation for our project will be our report.
We will only perform internal testing of the program, and will not include any outside users or resources for testing of the functionality.

Time schedule and work tasks

Work assignments

Definition: Interface
Product: graphics and backend for the application interface.
Resource Requirement: open source language (python, java script, HTML, CSS) to implement the product, and browser to visualize the product.
Dependencies: This work task can be completed without any of the other work tasks having been implemented.
Time Required: 2-3 work days.
Deadline: 1 / 04/ 2016

Definition: Grammar
product : definition of the building blocks syntax
Resource Requirement: Introduction to Compiler Design - Torben Ægidius Mogensen (Mogensen, 2011)
Dependencies: This work task can be completed independently of the other tasks.
Time Required: 1-2 work days
Deadline: 1/ 04/ 2016

Definition: Building Blocks
Product: graphics and backend for the buttons used for creating the algorithms as well as implementing them in the interface.
Resource Requirement: open source language (Python, JavaScript, HTML, CSS) to implement the product, and browser to visualize the product.
Dependencies: This work task needs the finished implementation of the interface and the parser to be completed.
Time Required: 5-6 work days
Deadline: 20/ 04/ 2016

Definition: Import and Export
Product: The ability to save code written in the program and export it as a file, so that it may be distributed and imported on a different machine.
Resource Requirement: open source language (Python, JavaScript, HTML, CSS) to implement the product, and browser to visualize the product.
Dependencies: This work task needs the finished implementation of the interface and the building blocks.
Time Required: 1-3 work days
Deadline: 23/ 04/ 2016

Definition: Step-by-Step Simulation
product: graphics and backend for the step by step algorithms simulation, as well as implementing them in the interface.
Resource Requirement: open source language (Python, JavaScript, HTML, CSS) to implement the product, and browser to visualize the product.
Dependencies: This work task depends on the complete implementation of the interface.
Time Required: 5-6 work days
Deadline: 15/ 05/ 2016

Definition: Pretty Print
product: graphics and backend for printing the variables results and the return results of the functions, as well as implementing them in the interface.
resource requirement: open source language (Python, JavaScript, HTML, CSS) to implement the product, and browser to visualize the product.
Dependencies: This work task needs the finished implementation of the interface to be completed.
Time Required: 8-10 work days
Deadline: 01/ 06/ 2016

References

Thomas H Cormen. *Introduction to algorithms.* MIT press, 2009.

Torben Ægidius Mogensen. *Introduction to Compiler Design.* Springer Science & Business Media, 2011.

Philip J. Guo. Online python tutor: Embeddable web-based program visualization for cs education. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, SIGCSE '13, pages 579–584, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1868-6. doi: 10.1145/2445196.2445368. URL `http://doi.acm.org/10.1145/2445196.2445368`.

Christos G Foutsitzis and Stavros N Demetriadis. Teaching algorithms with the use of a web-based scripted collaboration environment and algorithm visualization tool: Results from a case study. In *Intelligent Networking and Collaborative Systems (INCOS), 2010 2nd International Conference on*, pages 116–123. IEEE, 2010.

`https://scratch.mit.edu/`

`http://pythontutor.com/`

Student Signature

Valdas Zabulionis

[illegible]

Rune Franch Pedersen

[illegible]

Amr El Sayed

Amr Elsayed

14/03/2016

Supervisor Signature

Oleksandr Shturmov

[illegible]

2016-03-14