T3044

TECHNOLOGY STUDENT ASSOCIATION ®

SOFTWARE DEVELOPMENT

ORLANDO - FLORIDA

21th-25th JUNE, 2017

TABLE OF CONTENTS

RESEARCH

Understanding the working process of algorithms is a necessity to build up our project. Therefore we as developers must be able to dominate the most commonly used algorithms that are complicated to learn for the beginners. In the following the key parts of how certain algorithms work will be pointed out.

Breadth First Search

It is designed for all trees and graphs. It traverses from a root and explores all adjacent nodes.

Depth First Search

It is designed for all trees and graphs. It traverses from a root and explores as far as possible along each branch*.

Djikstra

It is designed for weighted directed graphs. It finds the single-source shortest path. The only condition to use this algorithm is the weight of every edge must be equal or bigger than null.

Kosaraju Sharir Algorithm

It is designed for directed graphs and finds the strongly connected components. It makes use of the fact that the transpose graph (the same graph with the direction of every edge reversed) has exactly the same strongly connected components as the original graph.*

Kruskal's Algorithm

It is designed for a connected weighted graph. It finds the minimum spanning tree but some auxiliary functions will be needed to realize its behaviourally basic algorithmic expression.

Prim's Algorithm

It is designed for weighted undirected graphs. It finds the minimum spanning tree. It starts from an arbitrary starting vertex, at each step adding the cheapest possible connection from the tree to another vertex.*

Topological Sorting

It is designed for directed acyclic graphs. If the algorithm is based on DFS, then it loops through each node of the graph, terminates when it hits any visited node.

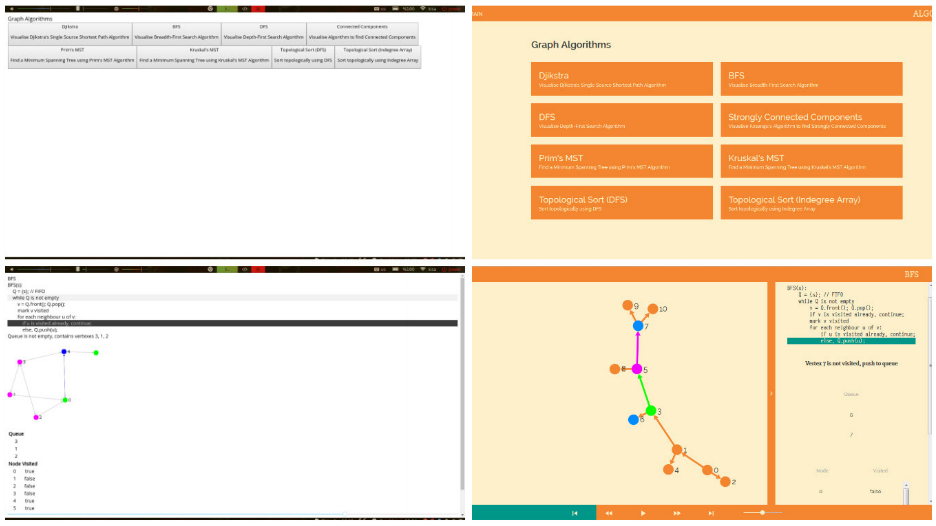
*)https://www.wikiwand.com/en/

DESCRIPTION

Heading away from the fact that science and technology shape our future, we want to make a contribution to education. According to the survey* results, the most popular subject of late years is the computer science. Considering the mass of people interested in computer science enlarges directly proportional to the popularity, more problems come up and wait to be solved during the learning process. There are some aspects are worth to take a look at, in order to specify the problem that we are going to deal with.

First of all, the requirements for a code to be well written should be listed. Readability, modularity, expressivity, and efficiency are the main concerns about a good program. Different techniques for readability exist, whereas expressivity and efficiency are more complicated and depend mainly on the programmer. Therefore modularity builds a bridge in between. A system's components may be separated via self-contained sequences of actions to be performed, as known as algorithms. In an attempt to achieve the most efficient written code, a programmer must acknowledge the working principle of algorithms. So that one can calculate the estimated run-time and necessary memory. Only then one gain the ability to optimize a code.

A nonignorable method for understanding how an algorithm works is called abstraction*, which is a technique for arranging complexity of computer systems. When it comes to the education process of programming, a big majority of students face problems, especially about abstraction. For instance, a few members of our chapter including us are preparing for the USA Computing Olympiad. In a given interview about how hard is it to learn and study programming, it is mentioned that programming class has a high rate of daunting mainly because of the inability of the comprehension the concept of abstraction.


Graph Algorithms
Djikstra
Visualise Djikstra's Single Source Shortest Path Algorithm
BFS
Visualise Breadth-First Search Algorithm
DFS
Visualise Depth-First Search Algorithm
Strongly Connected Components
Visualise Kosaraju's Algorithm to find Strongly Connected Components
Prim's MST
Find a Minimum Spanning Tree using Prim's MST Algorithm
Kruskal's MST
Find a Minimum Spanning Tree using Kruskal's MST Algorithm
Topological Sort (DFS)
Sort topologically using DFS
Topological Sort (Indegree Array)
Sort topologically using Indegree Array
BFS
Queue
Node
Visited

```
BFS(s):
    Q = {s}; // FIFO
    while Q is not empty
        v = Q.front(); Q.pop();
        if v is visited already, continue;
        mark v visited
        for each neighbour u of v:
            if u is visited already, continue;
            else, Q.push(u);
```

Vertex 7 is not visited, push to queue

As a conclusion, we decide to find an easier way of teaching in order to smooth over the adaptation period for abstract thinking, which is bringing in more visualization in contrast to the abstraction. We make a program that teaches how to program setting sight on algorithms.

1) www.telegraph.co.uk/education/educationpicturegalleries/10643255/Student-life-top-ten-most-popular-subjects.html
2) en.wikipedia.org/wiki/Abstraction_(software_engineering)

DESCRIPTION

Purpose

Algoriva works up to give a hand for programmers who suffer from the abstract flow of algorithm design. It interacts with the user at every step so that the user can arrange the functions of Algoriva in her own way.

Scope

This project consists of an animation visualizing customizable codes based on the predefined algorithms. Modules of the program include graphs and trees, which can be regulated by the user. Not only the code but also additional general information about the visualized algorithm are given at the same time. Bright colors are used for the animation to pave the way for following up the processed nodes. The user can change themes, for example when Algoriva is used under low light, one can set the dark theme so that eye health is protected. Furthermore one can write her own code and Algoriva checks for syntax errors, then runs the code and visualizes that new algorithm defined by the user without any preconditions. This Custom Code mode is what makes the biggest difference between Algoriva and other visualization programs on the market.

Stakeholder Identification

The program is for intermediate programmers, who basically knows fundamental elements of JavaScript. Majorly visual learners are recommended to experience Algoriva in order to obtain efficiency.

Social & Educational Value

Algoriva is designed for the programming learners considering teaching methods such as demonstration and collaboration.
With Algoriva more and more people keep learning to program without giving up on neither abstract thinking nor algorithm designing. Algoriva aims to lower the daunting ratio of programmers, who will become computer scientists and software developers in the future.

TECHNOLOGY STUDENT ASSOCIATION PLAN OF WORK

	Date	Task	Time involved	Team member responsible	Comments
1	02/21/2017	Detect a problem Find a solution	16 days	EY NS	Brainstorming across the whole chapter Interviewing with relevant students
2	03/09/2017	Plan development cycle	110 min	EY	Determining of the language, libraries, framework to use
3	04/09/2017	Initial git commit	120 min	EY	
4	04/25/2017	Plan abstract layouts	70 min	EY	Adding Modules API
5	05/16/2017	Finish first iteration	95 min	EY NS	
6	05/28/2017	Packaging	40 min	EY	Arranging compatibility

Advisor signature ____

REQUIREMENTS

Functional Requirements

The user shall be able to discover how to use the application from the "Usage" section, customize the theme, animation speed, and change which modules will be used in algorithms under the "Options" section.With a click on "Custom Code" section, the system enables the user to write, save, run and visualize her own code. The user shall choose various types of algorithms later to be visualized after clicking on the "Algorithms" section, add new nodes with the combination alt + left click or delete with the combination ctrl + alt + left click.

Non-functional Requirements

The system shall be compatible with Windows and Linux; portable via a web browser; effective since visual learning is one of the three primary ways to learn*; modular due to the fact that Algorithms section and Custom Code section shares the same infrastructure; mostly predictable because of the fact that solid techniques are used for teaching; reliable by virtue of the successful testings; repeatable with rewind option; safe to use on account of not reaching additional information from the computer; understandable in view of easy navigation and included usage manual.

Technical Requirements

Operating System: Windows 7 and later are supported. Both x86 and amd64 (x64) binaries are provided for Windows. However, the ARM version of Windows is not supported.Linux is supported. Ubuntu 12.04 and later, Fedora 21, Debian 8 are guaranteed to work.
Hardware: for Windows An Intel Pentium 4 processor or later that's SSE2 capable 512 MB of RAM; for Linux An Intel Pentium 4 processor or later that's SSE2 capable
Cross-Browser/Platform Support (all modern browsers including mobile browsers)

1) http://oedb.org/ilibrarian/hacking-knowledge/

STUDENT COPYRIGHT CHECKLIST

(for students to complete and advisors to verify)

1) Does your solution to the competitive event integrate any music? YES _____ NO __X__

If NO, go to question 2.

If YES, is the music copyrighted? **YES** _____ **NO** _____

If YES, move to question 1A. If NO, move to question 1B.

1A) Have you asked for author permission to use the music in your solution and included that permission (letter/form) in your documentation? If YES, move to question 2. If NO, ask for permission (OR use royalty free/your own original music) and if permission is granted, include the permission in your documentation.

1B) Is the music royalty free, or did you create the music yourself? If YES, cite the royalty free music OR your original music properly in your documentation.

CHAPTER ADVISOR: Sign below if your student has integrated any music into his/her competitive event solution.

I, _______________________________ (chapter advisor), have checked my student's solution and confirm that the use of music is done so with proper permission and is cited correctly in the student's documentation.

2) Does your solution to the competitive event integrate any graphics? YES __X__ NO _____

If NO, go to question 3.

If YES, is the graphic copyrighted, registered and/or trademarked? **YES** __X__ **NO** _____

If YES, move to question 2A. If NO, move to question 2B.

2A) Have you asked for author permission to use the graphic in your solution and included that permission (letter/form) in your documentation? If YES, move to question 3. If NO, ask for permission (OR use royalty free/your own original graphic) and if permission is granted, include the permission in your documentation.

2B) Is the graphic royalty free, or did you create your own graphic? If YES, cite the royalty free graphic OR your own original graphic properly in your documentation.

CHAPTER ADVISOR: Sign below if your student has integrated any graphics into his/her competitive event solution.

I, _______________________________ (chapter advisor), have checked my student's solution and confirm that the use of graphics is done so with proper permission and is cited correctly in the student's documentation.

3) Does your solution to the competitive event use another's thoughts or research? YES __X__ NO _____

If NO, this is the end of the checklist.

If YES, have you properly cited other's thoughts or research in your documentation? If YES, this is the end of the checklist.

If NO, properly cite the thoughts/research of others in your documentation.

CHAPTER ADVISOR: Sign below if your student has integrated any thoughts/research of others into his/her competitive event solution.

I, _______________________________ (chapter advisor), have checked my student's solution and confirm that the use of the thoughts/research of others is done so with proper permission and is cited correctly in the student's documentation.

TSA LEAP LEADERSHIP RESUME – TEAM EVENT

The resume must be typed using 11pt Arial or Calibri font. For more information about how to complete the resume, visit this link: (http://www.tsaweb.org/LEAP-competition-engagement)

TEAM IDENTIFICATION

Team ID: T3044

Competitive event: Software Development

Level: High School

LEADERSHIP EXPERIENCES (specific to a competitive event)

Enrolled a winter school for two weeks to master the computing languages. (Know)
Attended an informatics camp in Germany organized by a school network called MINT-Ec in order to learn to work with graphics and GitLab. (Know)
Made interviews and a survey to interact with students facing problems when they learn to program. (Do)

LEADERSHIP EXPERIENCES (connected to one or more of these categories: *Leadership Roles; Community Service/Volunteer Experiences; Leadership Development/Training; College/Career Planning)*

Leadership Development/Training
Attended LEAP workshop at the national TSA conference in Tennessee and at Bogazici University. (Know)
Encouraged other state members to attend trainings this year. (Do)
Leadership Roles
Presidents of computing club and science club. (Be)

TSA LEAP LEADERSHIP RESUME – TEAM EVENT

The resume must be typed using 11pt Arial or Calibri font. For more information about how to complete the resume, visit this link: (http://www.tsaweb.org/LEAP-competition-engagement)

TEAM IDENTIFICATION

Team ID: T3044

Competitive event: Coding

Level: High School

LEADERSHIP EXPERIENCES (specific to a competitive event)

Internship at an insurance company to learn how to work with databases (Know).
Membership of coding websites, where one can solve five problems given by the website in limited time every week and track self-evaluation with a success graph (Be).
Organized a local computing competition including a preparation camp for three days (Do).

LEADERSHIP EXPERIENCES (connected to one or more of these categories: *Leadership Roles; Community Service/Volunteer Experiences; Leadership Development/Training; College/Career Planning)*

Community Service/Volunteer Experiences
Coordination of Avrasya Marathon and Istanbul Half-Marathon. (Do)