T3044

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's Algorithm
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.

DESCRIPTION

The Problem and Its Solution

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.

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.

Screenshots from the first iteration (on the left) and from the last prototype (on the right)
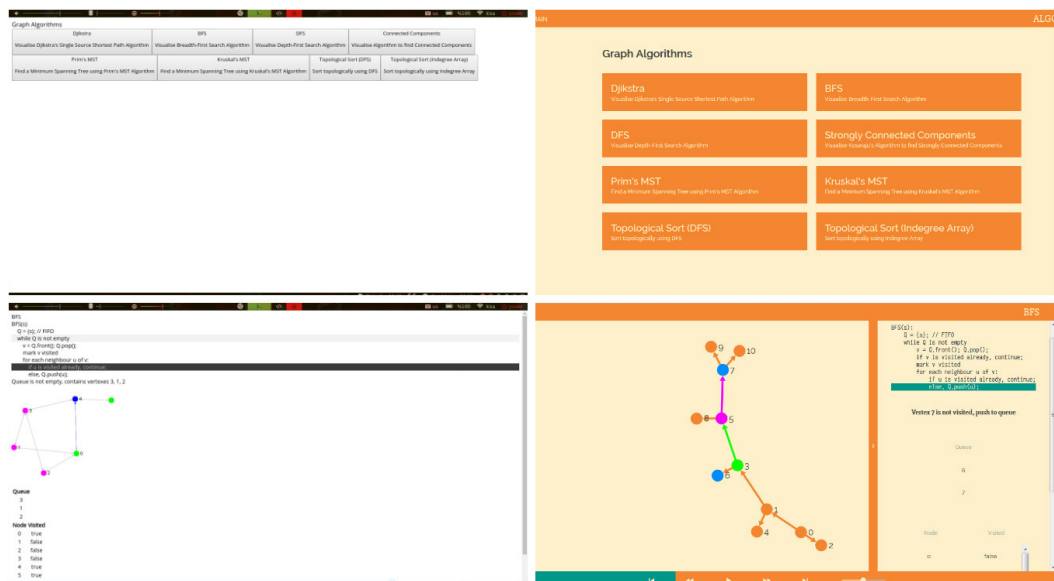
1) en.wikipedia.org/wiki/Abstraction_(software_engineering)

Purpose

Algoriva aims 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

There are two main functions of the software. First one is visualizing the pre-defined algorithms, such as BFS, DFS, Djikstra's / Kosaraju's / Kruskal's / Prim's Algorithms and Topological Sorting. The user chooses one of the example graphs titled by their virtues or creates a new graph, which will be memorized by Algoriva later if saved. Not only the code but also additional general information about the visualized algorithm are given at the same time. Bright colors are used in the animation so that important parts in it can be highlighted.

The other main function is to compile, run and visualize the custom code written by the user. Algoriva understands that newly described algorithm and develops a genuine animation. This Custom Code mode is what makes the biggest difference between Algoriva and other visualization programs on the market. While Algoriva is demonstrating how the user's algorithm works, the processed line of the code is highlighted and the table of the nodes shows the values of the nodes.

Moreover, 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.

Intended Population

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

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. Algoriva is designed for those who faces problems when it comes to the abstraction. During the development of Algoriva teaching methods such as demonstration and collaboration are considered. Thus Algoriva can be used in the classrooms as an educational tool. The cross-platform aspect of the software allows portability even on smartboards. 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 of the future.

www.telegraph.co.uk/education/educationpicturegalleries/10643255/Student-life-top-ten-most-popular-subjects.html

	TECHNOLOGY STUDENT ASSOCIATION PLAN OF WORK			
Date	Task	Time involved	Team member responsible	Comments
02/21/2017 1	Detect a problem Find a solution	16 days	S4959 S4955	Brainstorming across the whole chapter Interviewing with relevant students
03/09/2017 2	Plan development cycle	110 min	S4959	Determining of the language, libraries, framework to use
04/09/2017 3	Initial git commit	120 min	S4959	First step for coding
04/25/2017 4	Plan abstract layouts	70 min	S4959	Adding Modules API
05/16/2017 5	Finish first iteration	95 min	S4959 S4955	BFS Algorithm can be visualized
05/28/2017 6	Packaging	40 min	S4959	Arranging compatibility

Advisor signature _____

Technology Student Association (TSA) High School Competitive Events Guide for the 2017 and 2018 National TSA Conferences

5

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 and be able to add or remove nodes.

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)

JavaScript Conventions

Complies with the famous style guide of Douglas Crockford)?? bunu gerçekten yazmaya gerek var mı?
Uses linting, testing
Checks for duplicates (jsinspect)
Documentation as JSDOC comments

---

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

HIGH-LEVEL SOFTWARE DESIGN

Solution

The app should work on Windows, Linux and Mac. It should also be available as a web app on a website. To meet these requirements we wrote the app as a web app using React*, a component based UI design approach by Facebook, and used Electron*, a framework for creating native applications with web technologies like JavaScript, HTML, and CSS, to make it work on Windows, Mac an Linux.

System

-Bleeding Edge JavaScript features

The app had to be written using newest ES6* features such as Promises and arrow-functions, hence we used Babel, a compiler that transforms ES6 code (and other things like React's JSX syntax) into ES5 code.

-Ability to execute Custom Code

For Custom Code section the app needs to be able to evaluate custom codes written by the user. Since exposing the running environment of the app would be dangerous, we use JS-Interpreter, a sandboxed JavaScript interpreter in JavaScript, isolating it from the main context.

-Bundling

The app composes multiple modules, both third-party and in-house modules. We used Webpack module bundler, to bundle these modules into the app. Using webpack we could also minify the code, which we did, in order to reduce the loading time for the version on the browser.

-Testing

Testing is done with the popular Mocha* framework and Chai*. To make mocha work with webpack we are using a plugin, mocha-webpack, and jsdom, a framework for emulating browser in node.js.

-Linting and Duplicate Code

To find syntax and coding errors, enforce complying to javascript standards, and enforce consistent styling; eslint, a pluggable javascript linter. CSS and SCSS is linted using sass-lint. Duplicate code in the app is found using jsinspect.

reactjs.net/
electron.atom.io/
github.com/lukehoban/es6features

7

mochajs.org/
chaijs.com
babeljs.io/
eslint.org/

TESTING

```
yarn run v0.24.4
$ mocha-webpack --include app/__tests__/test.js --webpack-config
webpack/webpack.test.config.js "app/**/*.test.js"
 utils
   themeVars
     ✓ throws an error if style couldnt be found
     ✓ returns the style for themes
   themedStyle
     ✓ throws an error if no style is given
     ✓ throws an error if the class couldnt be found
     ✓ returns the name of the style with both -theme prefixed one and the nonprefixed one
AlgorithmInner
   ✓ renders correctly
AnimationControls
   ✓ renders correctly speed: 0, progress: 0, paused: true
   ✓ renders correctly speed: 0, progress: 0, paused: false
   ✓ renders correctly speed: 0, progress: 50, paused: true
   ✓ renders correctly speed: 0, progress: 50, paused: false
   ✓ renders correctly speed: 0, progress: 100, paused: true
   ✓ renders correctly speed: 0, progress: 100, paused: false
   ✓ renders correctly speed: 50, progress: 0, paused: true
   ✓ renders correctly speed: 50, progress: 0, paused: false
   ✓ renders correctly speed: 50, progress: 50, paused: true
   ✓ renders correctly speed: 50, progress: 50, paused: false
   ✓ renders correctly speed: 50, progress: 100, paused: true
   ✓ renders correctly speed: 50, progress: 100, paused: false
   ✓ renders correctly speed: 100, progress: 0, paused: true
   ✓ renders correctly speed: 100, progress: 0, paused: false
   ✓ renders correctly speed: 100, progress: 50, paused: true
   ✓ renders correctly speed: 100, progress: 50, paused: false
   ✓ renders correctly speed: 100, progress: 100, paused: true
   ✓ renders correctly speed: 100, progress: 100, paused: false
```

BigButton
- ✓ renders correctly cols: 0
- ✓ renders correctly with desc cols: 0
- ✓ renders correctly cols: 1
- ✓ renders correctly with desc cols: 1
- ✓ renders correctly cols: 2
- ✓ renders correctly with desc cols: 2
- ✓ renders correctly cols: 3
- ✓ renders correctly with desc cols: 3
- ✓ renders correctly cols: 10
- ✓ renders correctly with desc cols: 10
- ✓ renders correctly cols: 50
- ✓ renders correctly with desc cols: 50
- ✓ renders correctly cols: 100
- ✓ renders correctly with desc cols: 100

DialogComponent
- ✓ renders correctly

Header
- ✓ renders correctly with minimum props
- ✓ renders correctly with back prop
- ✓ renders correctly with current prop
- ✓ renders correctly with both back and current props

InformationDemandingButton
- ✓ renders correctly with minimum props
- ✓ renders correctly when demandCondition
- ✓ renders correctly when not demandCondition
- ✓ renders correctly when validate returns false

Prompt
- ✓ renders correctly

SideDrawer
- ✓ renders correctly when closed side: left
- ✓ renders correctly when opened side: left
- ✓ renders correctly when closed side: right
- ✓ renders correctly when opened side: right

algorithm-helpers
 Algorithm
- ✓ throws when not given a name
- ✓ asView has 2 fields, name and view
- ✓ runs dry, when dry run is invoked
- ✓ add normal input
- ✓ add init input
- ✓ instance.addCode
- ✓ instance.addTable
- ✓ instance.addNodedTable
- ✓ instance.addText

modules
 Graph
- ✓ .snap
- ✓ .module.type
- ✓ .module.layout
- ✓ .module.data

 Table
- ✓ .snap
- ✓ .module.type
- ✓ .module.layout
- ✓ .module.data

 Text
- ✓ .snap
- ✓ .module.type
- ✓ .module.layout
- ✓ .module.data

 Examples
- ✓ .snap
- ✓ .module.type
- ✓ .module.layout
- ✓ .module.data

 .data - Methods
- ✓ .addCustom
- ✓ .deleteCustom
- ✓ .renameCustom

 Code
- ✓ .snap
- ✓ .module.type
- ✓ .module.layout
- ✓ .module.data

Description
  ✓ .snap
  ✓ .module.type
  ✓ .module.layout
  ✓ .module.data
settings
 Settings
   1) returns a monad
   ✓ doesn't create unnecessary paths
   ✓ can create paths when needed
   ✓ can overwrite prevs
   ✓ can set defaults
   ✓ defaults do not overwrite
   ✓ can get values
   2) can be referenced from a middle of a chain

AlgorithmPageView
  ✓ renders correctly

AlgorithmView
  ✓ renders correctly

AppView
  ✓ renders correctly

CustomCodeView
  ✓ renders correctly

MainView
  ✓ renders correctly

OptionsView
  ✓ renders correctly

PageViewFactory
  ✓ returns a component that renders correctly when given pure text
  ✓ returns a component that renders correctly when given nothing
  ✓ returns a component that renders correctly when given HTML
  ✓ throws when tried to be used as a React Element

104 passing (171ms)
2 failing

1) settings Settings returns a monad:

   AssertionError: expected [Function: fn] to equal [Function: fn]
    + expected - actual

   at Context.<anonymous> (.tmp/mocha-webpack/369dfcae2ffa9e6ef3a80c64c83a74c7
/369dfcae2ffa9e6ef3a80c64c83a74c7-output.js:7104:22)

2) settings Settings can be referenced from a middle of a chain:
  TypeError: Cannot create property 'asdf' on string 'ap56ple'
  at Function.fn.set (.tmp/mocha-webpack/369dfcae2ffa9e6ef3a80c64c83a74c7
/369dfcae2ffa9e6ef3a80c64c83a74c7-output.js:7208:17)
   at Function.set (.tmp/mocha-webpack/369dfcae2ffa9e6ef3a80c64c83a74c7
/369dfcae2ffa9e6ef3a80c64c83a74c7-output.js:7269:32)
   at Context.<anonymous> (.tmp/mocha-webpack/369dfcae2ffa9e6ef3a80c64c83a74c7
/369dfcae2ffa9e6ef3a80c64c83a74c7-output.js:7145:16)

info Visit https://yarnpkg.com/en/docs/cli/run for documentation about this command.

END USER PRODUCT DOCUMENTATION

We already implemented a usage manual in Algoriva. There are two use cases shown by diagrams below.

Visualize Your Own Algorithm
Preconditions: Algoriva is installed.

Basic Flow of Events
1.The user runs Algoriva and clicks on the 'Custom Code' section.
2. Right bar is automatically opened.
3. The user chooses an algorithm type and the system enables a starting vertex.
4. The user writes code, the system checks for syntax errors.
5. The user describes the algorithm in pseudo-code section, creates a table if needed.
6. The user saves the code and runs the application.
7. The user clicks on the 'VISUALIZE' button and the system directs to the animation page.

Alternative Flows
1a. The user clicks on one of the other three sections.
   1a1.The user clicks on the MAIN arrow.
2a. The bar is hidden.
   2a1.The user clicks on the darkest section on the right-hand side to open that hidden bar.
4a. The user does not know how to write code.
   4a1. The user clicks on the darkest section on the left-hand side to open left hidden bar.
      4a1a. The user finds an example code.
6a. There are run-time errors found by the system.
   6a1. The user corrects the mistakes.

Preconditions: Algoriva is installed.

Basic Flow of Events

1.The user runs Algoriva and clicks on the 'Algorithms' section.

2.The user chooses DFS and clicks on the play button.

3.The system asks for a starting vertex and the user types any number.

4.Algoriva starts the animation.

Alternative Flows

1a. The user clicks on one of the other three sections.

   1a1.The user clicks on the MAIN arrow.

2a. The user chooses an algorithm else than DFS.

   2a1.The user clicks on ALGORITHMS arrow.

2b. The user clicks on the left bar.

  2b1. The user reads the explanation about DFS and chooses an example graph.

  2b2. The user chooses 'Custom Graphs'.

     2b2a. The user enters a name for the new graph which is on the screen.

2c. The user clicks on the right bar and sees the code block and a table showing whether the nodes visited.

2d. The user clicks on the forward/ previous/rewind/end button.

  2d1. The system assumes starting vertex as 0 and jumps to the 4th step.

3a. The user types a letter and faces with a message: 'node doesn't exist (letter)'

  3a1. The user deletes the letter.

3b. The user types too big number and faces with a message: 'node doesn't exist (too big number)'

  3b1. The user deletes that number and types an existing smaller number.

  3b2. The user adds nodes by CTRL + LEFT CLICK.

     3b2a. The user connects the new node with any other existing node by clicking on the nodes to be connected.

       3b2a1. The system asks for a weight, the user enters a value.

END USER PRODUCT DOCUMENTATION

USE CASE for CUSTOM CODE

Saved codes can be seen on the left hidden bar by click on the darkest left-hand side of the frame. To save a code the user shall click on the plus symbol, enter a project name in the box enabled by the system, then the window is closed and the focus is returned to the page in the state it was in before this window was displayed. In order to rename or delete a saved code, the user shall click on three points next to the name of the code. The user shall be able to indicate the algorithm type, a starting vertex, and their algorithm's pseudocode from the right hidden bar, which can be opened by a click on the darkest right-hand side of the frame. Buttons below the bar shall run and visualize the program as they are labeled

---

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

END USER PRODUCT DOCUMENTATION

Getting help

Whenever you need help, you should check 'Usage' section on the first page.

Usage

Changing settings

If you want to change the theme, higher or lower the animation speed, or configure the modules, then find the 'Options' section on the first page.

Options

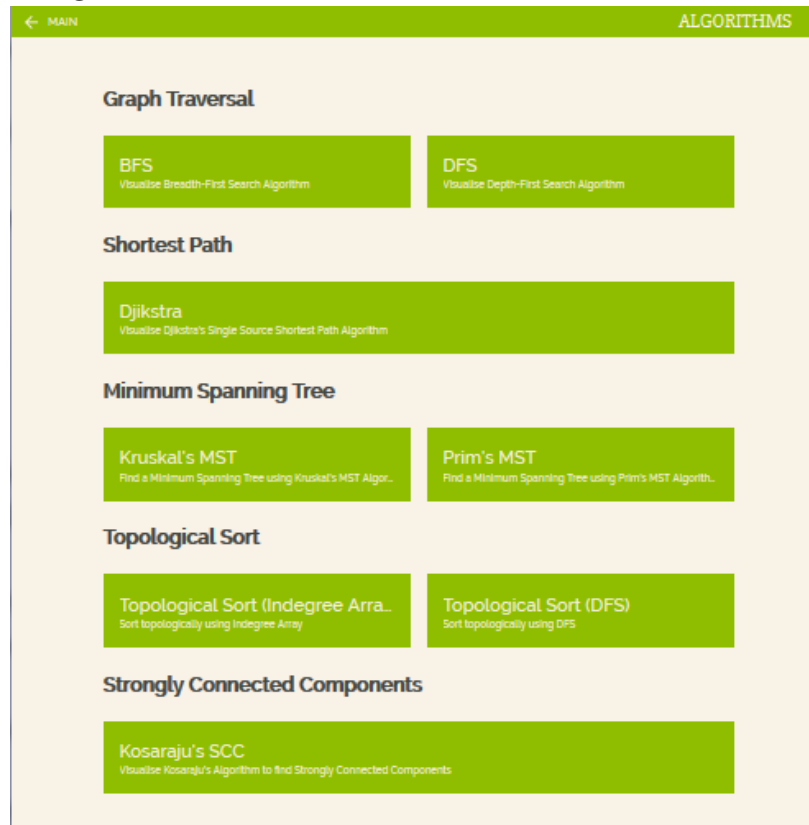## Visualizing Pre-Defined Algorithms

Your first step should be clicking on the 'Algorithms' section on the first page.

Algorithms
Choose from numerous algorithms to visualize

Variety of Algorithms

Choose your desired algorithm to be later visualized.

Using the Algorithm Page

An example graph appears automatically after the algorithm is chosen. (in this case BFS) Other example graphs and general information can be found in the left bar. Actual code and additional information can be found in right bar.

14
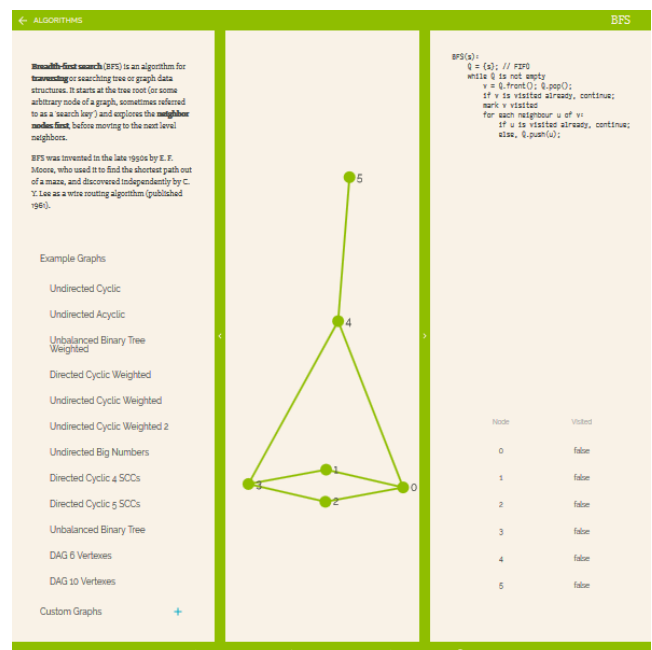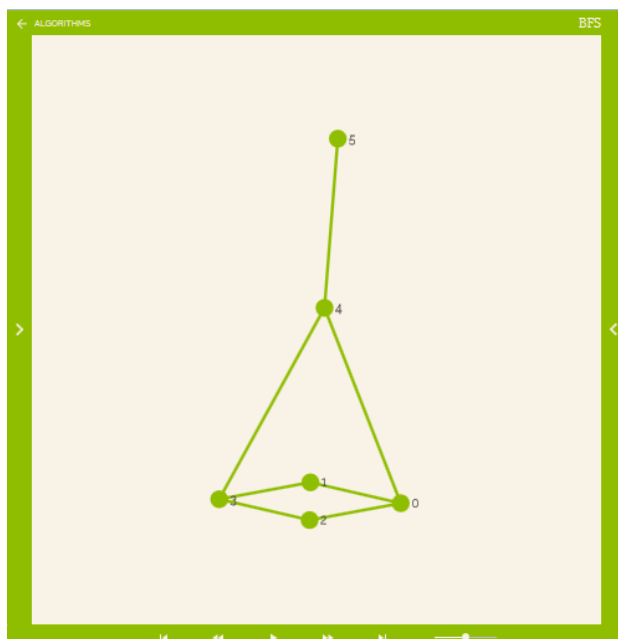
Control Bar

Play: Starts the animation. Asks for a starting vertex if needed.
Rewind: Resets the animation so that you can give a new starting vertex if needed.
Scan Backward: Undoes last step, jumps back to the previous stage.
Fast forward: Makes one step progress, visualizes the next step.
End: Finishes the visualization in one step.
Animation Speed: The user can adjust how fast the animation works.
Progress Bar: Progress can be seen with a different color.

Starting Vertex

0

Creating Own Graphs

If you have made any changes on the graph, then you can save that current graph giving it a name. You can always rename or delete your custom graph > ⋮ >

Rename ✏
Delete 🗑

← ALGORITHMS

Breadth-first search (BFS) is an algorithm for **traversing** or searching tree or graph data structures. It starts at the tree root (or some arbitrary node of a graph, sometimes referred to as a 'search key') and explores the **neighbor nodes first**, before moving to the next level neighbors.

BFS was invented in the late 1950s by E. F. Moore, who used it to find the shortest path out of a maze, and discovered independently by C. Y. Lee as a wire routing algorithm (published 1961).

Example Graphs

Undirected Cyclic

Undirected Acyclic

Unbalanced Binary Tree Weighted

Directed Cyclic Weighted

Undirected Cyclic Weighted

Undirected Cyclic Weighted 2

Undirected Big Numbers

Directed Cyclic 4 SCCs

Directed Cyclic 5 SCCs

Unbalanced Binary Tree

DAG 6 Vertexes

DAG 10 Vertexes

Custom Graphs   +

Custom Code

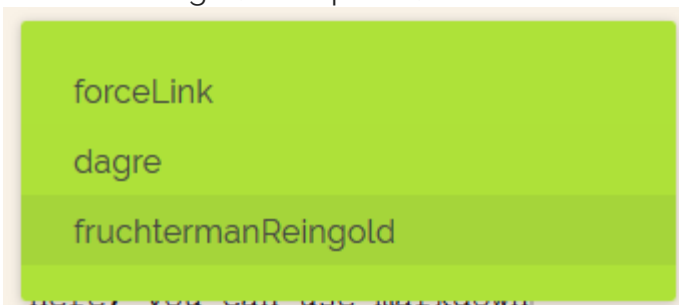You can write your own code under the 'Custom Code' section on the first page.
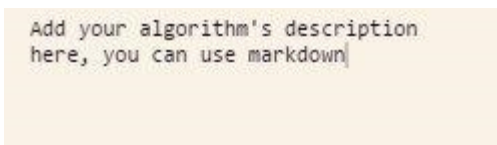

Using the Custom Code Page

The user should first fill the form on the right bar

Choosing the Rendering Layout for Graphs

If your graph is directed acyclic, dagre option is recommended. Otherwise it depends on your default settings. (see: Options)

Description of Your Algorithm

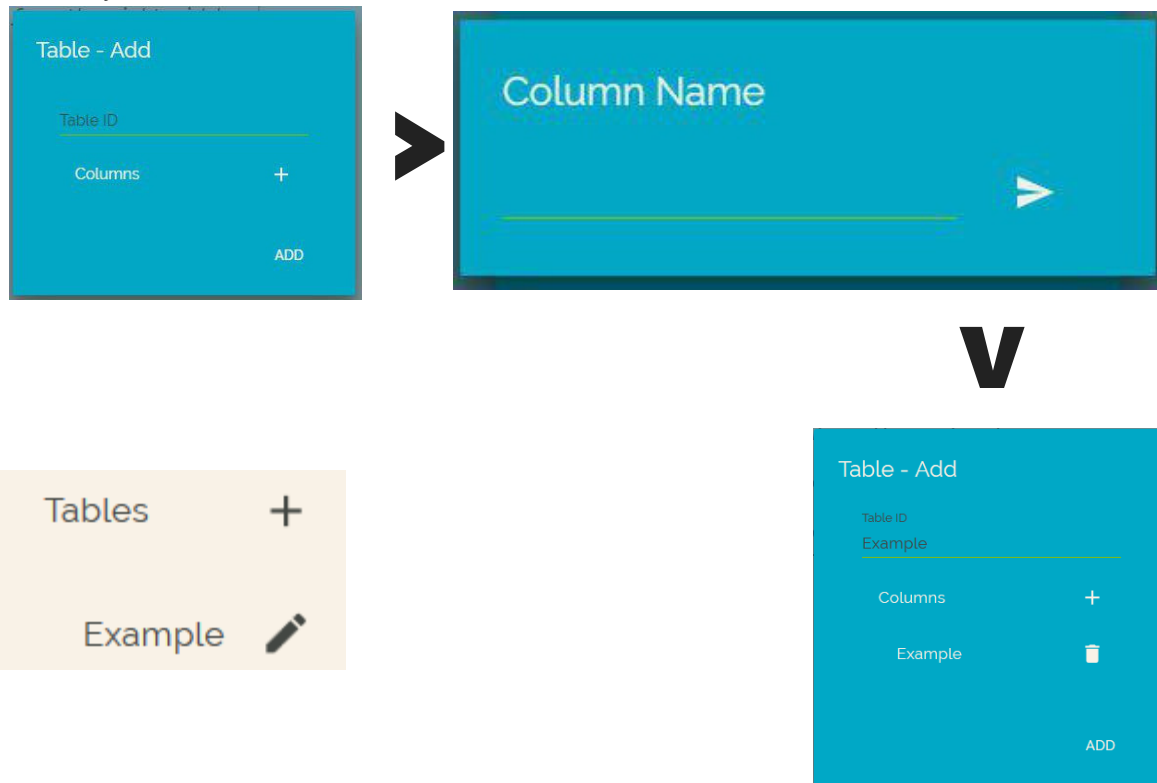This part will appear on the left hidden bar (see: Using the Algorithm Page)

Pseudo-Code of Your Algorithm

This part will appear on the right hidden bar. (see: Using the Algorithm Page) Every line will be later highlighted.

14

Adding Tables on Custom Code Mode

Name your table and add columns as needed.

You can add as many tables as you need by clicking on the plus sign. You can always rename your tables. delete columns or tables.

Running Your Code Before Visualizing

When you are done with your code, be sure that you have no errors. Now you can run your code. Debug bindings are activated, too.

Saving Your Code

Saved codes can be found in the left hidden bar. You can save as many algorithms as you want. You can rename or delete them.
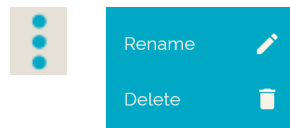
Visualizing Your Code

You will be questioned whether you would like to save your code before visualizing. Then you will be directed to the algorithm page. (see: using algorithm page)

14

USE CASE for CUSTOM CODE

Saved codes can be seen on the left hidden bar by click on the darkest left-hand side of the frame. To save a code the user shall click on the plus symbol, enter a project name in the box enabled by the system, then the window is closed and the focus is returned to the page in the state it was in before this window was displayed. In order to rename or delete a saved code, the user shall click on three points next to the name of the code. The user shall be able to indicate the algorithm type, a starting vertex, and their algorithm's pseudocode from the right hidden bar, which can be opened by a click on the darkest right-hand side of the frame. Buttons below the bar shall run and visualize the program as they are labeled

SELF EVALUATION & FUTURE PROSPECTS

We are a team of hardworking friends chasing the same dream. Our dream is to give hope. We open a window to all the world and his wife because all they need is another perspective to achieve. It is really motivating to feel our business climate because we both have values such as respect and compromise, During the project we always care other's opinions. We know one hand washes the other and together they wash the face. We made our biggest signs of progress when we sat down together and exchange opinions for a while. Despite the fact that we had our own ideas we did not know how to manage them unless both of us added something from oneself. Only then we figured out what to do with our ideas.

HOW DID IT START? HOW IT DEVELOPED AND IMPROVED? WHAT ARE OUR WEAKNESSES? FUTURE PROSPECT: GEOMETRY LIMITS AND EXCLUSIONS

1-Brainstorm, current problems on daily basis
2-

REFERENCES & LICENCES

```
"dependencies": {
    "babel-preset-es2015": "^6.24.1",
    "babel-standalone": "^6.24.2",
    "brace": "^0.10.0",
    "buckets-js": "^1.98.2",
    "chai-jest-snapshot": "^1.0.0",
    "chroma-js": "^1.3.3",
    "classnames": "^2.2.5",
    "dagre": "^0.7.4",
    "electron-reload": "^1.1.0",
    "electron-window-state": "^4.1.1",
    "extract-text-webpack-plugin": "^2.1.2",
    "graphology": "^0.10.0",
    "js-interpreter": "^1.4.3",
    "loaders.css": "^0.1.2",
    "lodash.flattendeep": "^4.4.0",
    "lodash.isequal": "^4.5.0",
    "lodash.isnil": "^4.0.0",
    "lodash.isplainobject": "^4.0.6",
    "lodash.mapvalues": "^4.6.0",
    "lodash.merge": "^4.6.0",
    "lodash.pick": "^4.4.0",
    "lodash.pickby": "^4.6.0",
    "lodash.sample": "^4.2.1",
    "lodash.shuffle": "^4.2.0",
    "lodash.union": "^4.6.0",
    "lodash.without": "^4.4.0",
    "material-ui": "^0.18.0",
    "mocha": "^3.4.2",
    "mocha-webpack": "^0.7.0",
    "node-localstorage": "^1.3.0",
    "prop-types": "^15.5.6",
    "react": "^15.5.4",
    "react-ace": "^5.0.1",
    "react-dom": "^15.5.4",
    "react-hot-loader": "next",
    "react-markdown": "^2.5.0",
    "react-promise": "^1.1.3",
    "react-tap-event-plugin": "^2.0.1",
    "sigma": "^1.2.0"
},
"devDependencies": {
    "babel-core": "^6.24.1",
    "babel-eslint": "^7.2.1",
    "babel-loader": "7",
    "babel-plugin-syntax-dynamic-import": "^6.18.0",
    "babel-plugin-transform-class-properties": "^6.24.1",
    "babel-plugin-transform-object-rest-spread": "^6.23.0",
    "babel-preset-react": "^6.24.1",
    "chai": "^4.0.2",
    "css-loader": "^0.28.0",
    "electron": "^1.6.2",
    "electron-builder": "^18.1.0",
    "electron-builder-squirrel-windows": "^18.0.0",
    "eslint": "^3.19.0",
    "eslint-config-airbnb": "^14.1.0",
    "eslint-import-resolver-webpack": "^0.8.1",
    "eslint-loader": "^1.7.1",
    "eslint-plugin-import": "^2.2.0",
    "eslint-plugin-jsx-a11y": "^4.0.0",
    "eslint-plugin-react": "^6.10.3",
    "exports-loader": "^0.6.4",
    "file-loader": "^0.11.1",
    "html-loader": "^0.4.5",
    "identity-obj-proxy": "^3.0.0",
    "imports-loader": "^0.7.1",
    "inject-loader": "^3.0.0",
    "jsdom": "^11.0.0",
    "jsdom-global": "^3.0.2",
    "jsinspect": "^0.12.4",
    "markdown-loader": "^2.0.0",
    "node-sass": "^4.5.2",
    "node-sass-json-importer": "^3.0.2",
    "null-loader": "^0.1.1",
    "react-test-renderer": "^15.5.4",
    "sass-lint": "^1.10.2",
    "sass-loader": "^6.0.3",
    "sass-material-colors": "^0.0.5",
    "script-loader": "^0.7.0",
    "style-loader": "^0.16.1",
    "url-loader": "^0.5.8",
    "webpack": "^2.3.3",
    "webpack-config": "^7.0.0",
    "webpack-dev-server": "^2.4.2",
    "webpack-node-externals": "^1.6.0"
}
```

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, re\[...\] _____

   If YES, move to question 2A. If NO, \[...\]

   2A) Have you asked for author per\[...\] included that permission (letter/form) in your documentation? If Y\[...\] n (OR use royalty free/your own original graphic) and if permissi\[...\] umentation.

   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.

Technology Student Association (TSA) High School Competitive Events Guide for the 2017 and 2018 National TSA Conferences

10

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)