Software Development

AlgoriVA

Algorithm Visualization

T3044

S4959; S4955

Orlando, FL TSA National Conference

6-21-2017

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 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.

Dijkstra'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 behaviorally 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

Nowadays, nearly every single thing in this world we live in utilizes computers, either directly, if not, indirectly. Having such an important role, lots and lots of people worked in the field of computers for 4 decades, in order to improve them. This improvement has come in two variations; hardware and software. Improvements in the hardware part are astounding, but still, there is a limit to what extent computer hardware can be improved. Because of this, improvements in the software part are even more important. In order to write the most efficient code for a given task, a programmer must acknowledge the working principle of algorithms. A programmer who wants to be `the one` in her field must know, or at least be aware of the existence of as many algorithms as possible. Through algorithms, the max-min-average running time and memory requirements of a program can be analyzed.

A fundamental 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 of programming, a big majority of students face problems 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 decided to find an easier way of teaching algorithms in order to smooth over the adaptation period for abstract thinking, which is bringing in more visualization in contrast to the abstraction. We made an app that teaches how to program setting sight on algorithms.

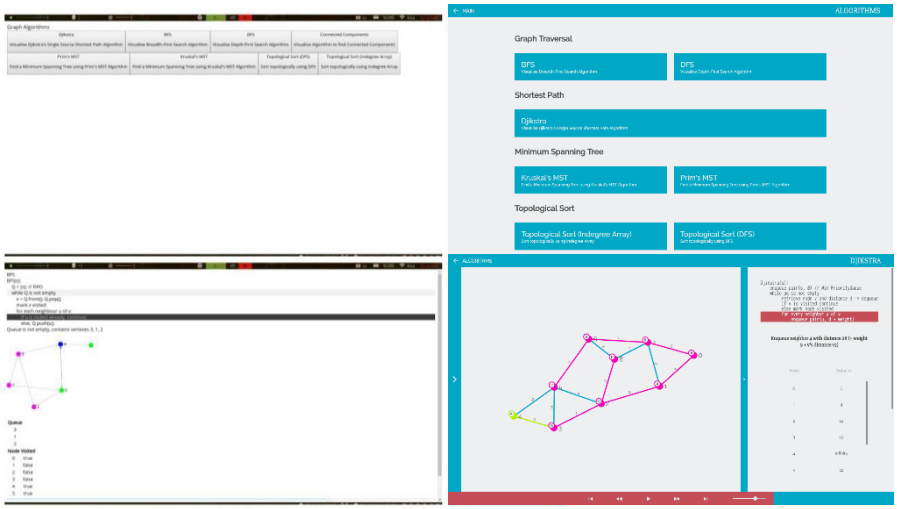


Figure 1: Screenshots from the first iteration (left) and the last prototype (right)

Purpose

AlgoriVA aims to help for programmers who have problems getting the abstract flow of algorithm design by visualizing algorithms. It interacts with the user at every step so that the user can arrange the functions of AlgoriVA in her own way.

As a side goal, AlgoriVA aims to get children familiar with programming as early as possible. It's simple and to the point written pseudo-codes can be understood by anyone with very little computer knowledge.

Scope

There are two main functions of the software. First one is visualizing the pre-defined algorithms, such as BFS, DFS and Topological Sorting. The user chooses one of the example graphs titled by their virtues or creates a new graph, which will be kept on memory by AlgoriVA until it is 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 custom algorithms written by the user. AlgoriVA understands that newly written algorithm and develops a genuine animation for it. This animation is no different than the ones of pre-defined algorithms, there are no limitations. 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 additional information about the algorithm is shown. Moreover, the user can change themes. For example when AlgoriVA is used under low light, one can set one of the dark themes so that eye health is protected.

Intended Audience

The program includes a selected list of pre-defined algorithms, which can be understood by anyone from 7 to 70 with little computer experience. The Custom Code mode, on the other hand, is meant for intermediate programmers, who basically knows fundamental elements of JavaScript language and is able to use an API. Majorly visual learners are recommended to experience AlgoriVA in order to obtain efficiency.

Social and Educational Value

Heading away from the fact that science and technology shape our future, we want to make a contribution to education. According to a survey done by The Telegraph, the most popular subject in the late years is 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 people who face problems learning algorithms. During the development of AlgoriVA, teaching methods such as demonstration and collaboration are considered. Thus AlgoriVA can be used in classrooms as an educational tool. The cross-platform aspect of the software allows portability even on smart boards. 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.

TECHNOLOGY STUDENT ASSOCIATION
PLAN OF WORK

Date	Task	Time involved	Team member responsible	Comments
02-21-2017	Detect a problem Find a solution	16 days	S4959 S4955	Brainstorming with the whole chapter Interviewing with relevant students
03-09-2017	Plan development cycle	110 min	S4959	Determine the language, fundamental libraries, and frameworks to use.
04-09-2017	Initial commit	120 min	S4959	Initial commit with base configuration
04-25-2017	Plan abstract layouts	3 hours	S4959	Modularity, plugins: Adding Modules internal API
05-08-2017	Finish first iteration	95 min	S4959 S4955	BFS Algorithm can be visualized, start writing tests
05-16-2017	Tweak Style, Theming	120 min	S4959 S4955	Add a theming system
05-28-2017	Packaging	80 min	S4959	Package app for Web, OSx, Windows, Linux
06-01-2017	End User Documentation	1 day	S4959 S4955	Finish End User Documentation and thereby the whole portfolio
Advisor signature _____				

Project Requirements

Functional Requirements

The user must 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 must enable the user to write, save, run and visualize her own code. The system must ease the coding process by giving information about the written code constantly and make it easier to debug the code. The user must be able to choose various types of algorithms later to be visualized after clicking on the "Algorithms" section. General information about the algorithm and examples suited for the algorithm must be included. The system must only show the most important visualization aspect of the algorithm on the initial open, though the user must be able to open additional visualizations.

Non-functional Requirements

The system must be compatible with all major operating systems and be 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 share the same infrastructure; predictable because of the fact that solid techniques are used for teaching; reliable by virtue of the successful tests; 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. Linux is supported. Ubuntu 12.04 and later, Fedora 21, Debian 8 are guaranteed to work. Minimum supported OSx version

Hardware: An Intel Pentium 4 processor or later that's SSE2 capable and 512 MB RAM

Cross-Browser/Platform Support (all modern browsers including mobile browsers)

Code Conventions

The source code must be documented, tested and linted. To make it easier for future collaborators to collaborate, “Code Conventions for the JavaScript Programming Language” by Douglas Crockford must be enforced and linted accordingly. Source code must not contain by any means duplicate code, complying with DRY principle. Duplicate code must be found during testing. End-user facing API endpoints must be documented with care in the source code.

High-Level Software Design

Overview

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, and Linux.

System

Cutting 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.

The app also checks for errors in the written code through babel.

Bundling

The app composes multiple modules, both third-party and in-house. 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.

Integration

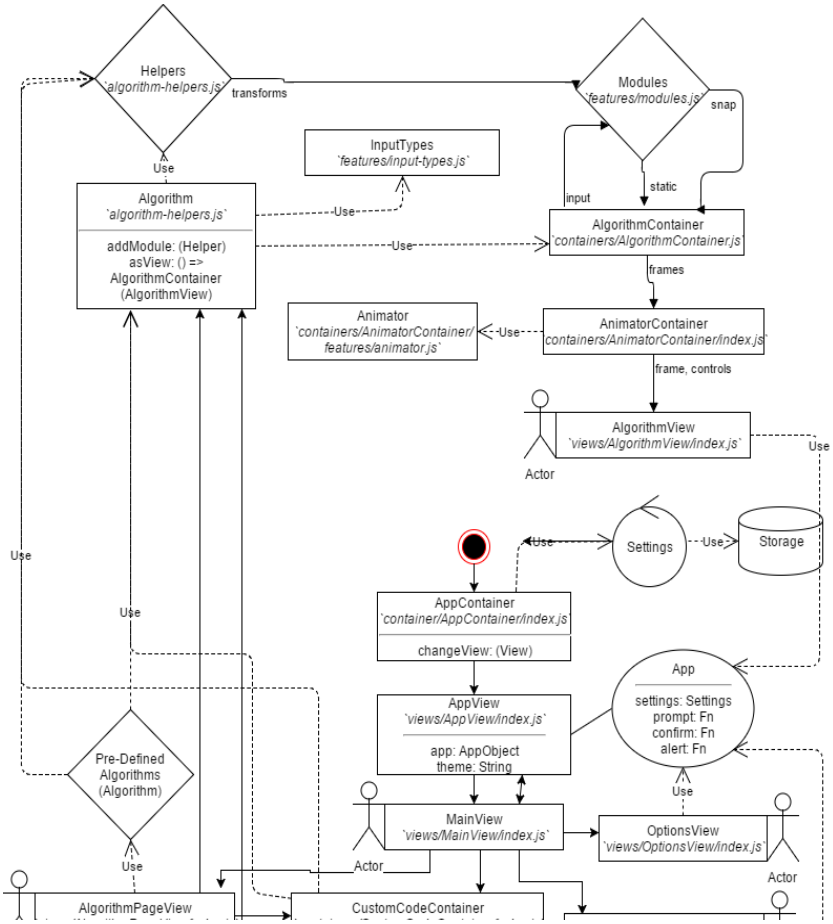


Figure 2: Relationship Graph of some major components

(Diamond: List, Square: Component, Circle: Object)

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 with Javascript standards, and enforce consistent styling; eslint, a pluggable Javascript linter. CSS and SCSS are linted using sass-lint. Duplicate code in the app is found using jsinspect.

There are currently 106 unit tests in the last prototype, 2 of which are failing but have corresponding workarounds. There are no duplicates and only 2 eslint warnings. Vendor code files are not checked.

Testing Output

```
$ 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)
```

JSinspect Output

```
$ jsinspect --ignore "test|min" app

No matches found across 55 files
```

ESLint Output

```
$ eslint app

/builds/omeryagmurlu/algoriv/app/utils.js
  131:2  warning  Unexpected console statement  no-console

/builds/omeryagmurlu/algoriv/app/views/PageViewFactory/index.js
  13:4  warning  Dangerous property 'dangerouslySetInnerHTML' found  react/no-danger

✖ 2 problems (0 errors, 2 warnings)
```

End User Product Documentation

Getting Help

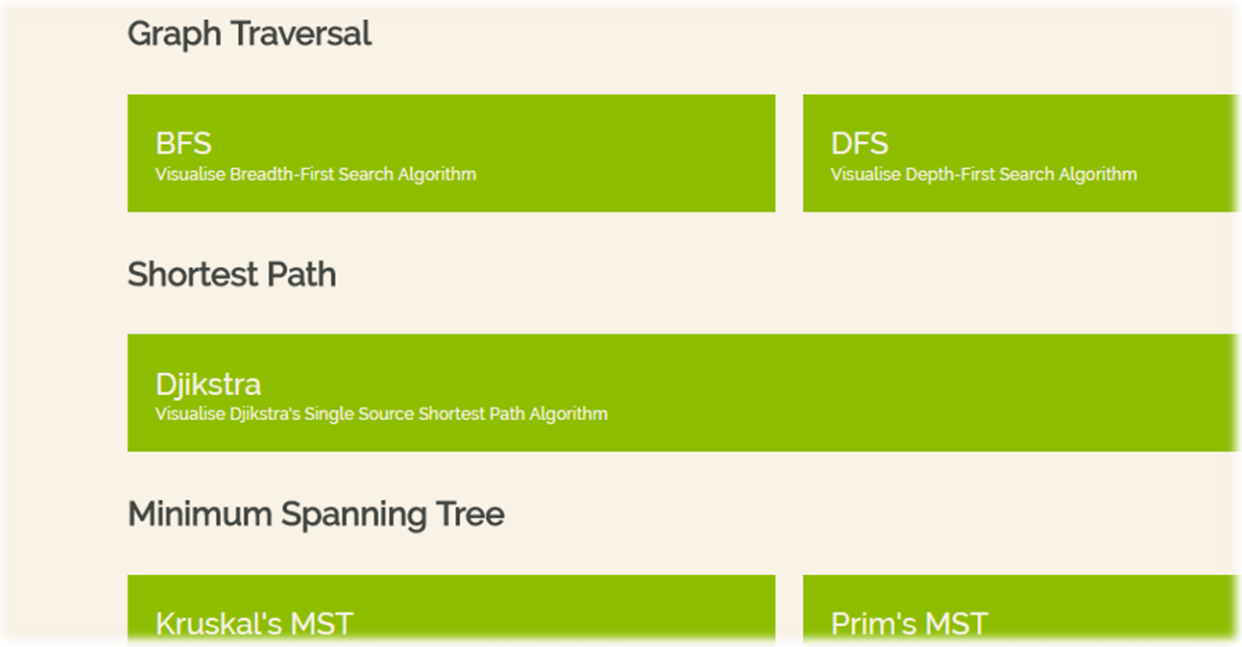
Whenever you need help, you should check ‘Usage’ section on the first page. There you can also find the reference for the Visualization API, which is used in the Custom Code section.

Changing Options

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.

Visualizing a Pre-Defined Algorithm

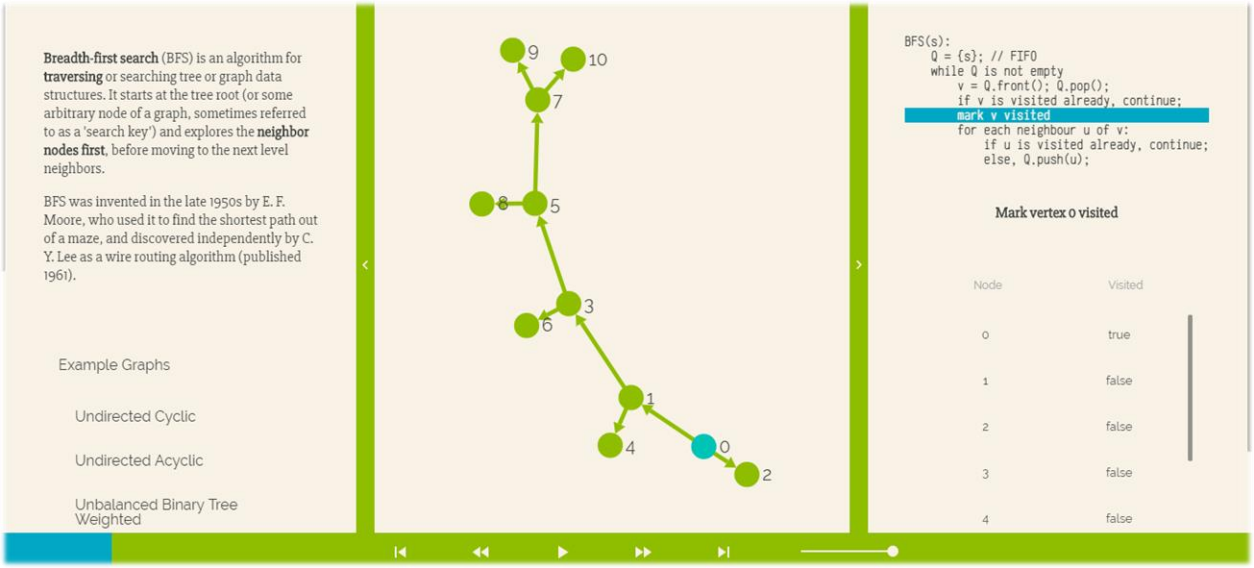
On the main page, clicking on the ‘Algorithms’ section leads you to a list of pre-defined algorithms. There you can choose one of them to visualize it.



Using the Algorithm Page

An example suited for the algorithm appears automatically after the algorithm is chosen. Other examples and general information can be found in the left bar. Pseudo-code and additional information can be found in right bar. To open these bars, you can click on the handles on their respective sides. You

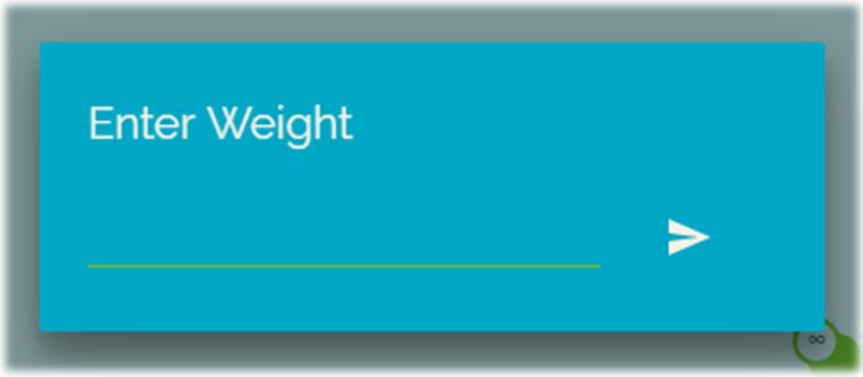
can change the input as you wish.



Changing Graph Input

To change a graph, you must push CTRL/CMD. It acts as a modifier and enables modifying the graph. There are 3 supported actions.

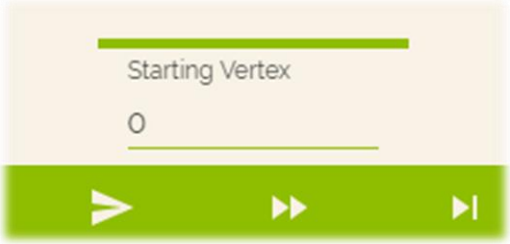
- Add Node: Clicking on the background when the modifier key is pushed down will add a new node.
- Remove Node: Clicking on a node while both the ALT and the modifier key is pushed down will remove that node and connected edges. You cannot remove a node if it is the starting node.
- Add Edge: Clicking on 2 nodes consequently will add an edge from the first to the second. If the graph is a weighted graph, then a prompt asking for weight will pop up.



Control Bar

In the control bar, there are animation controls. There are currently 7 parts of this bar.

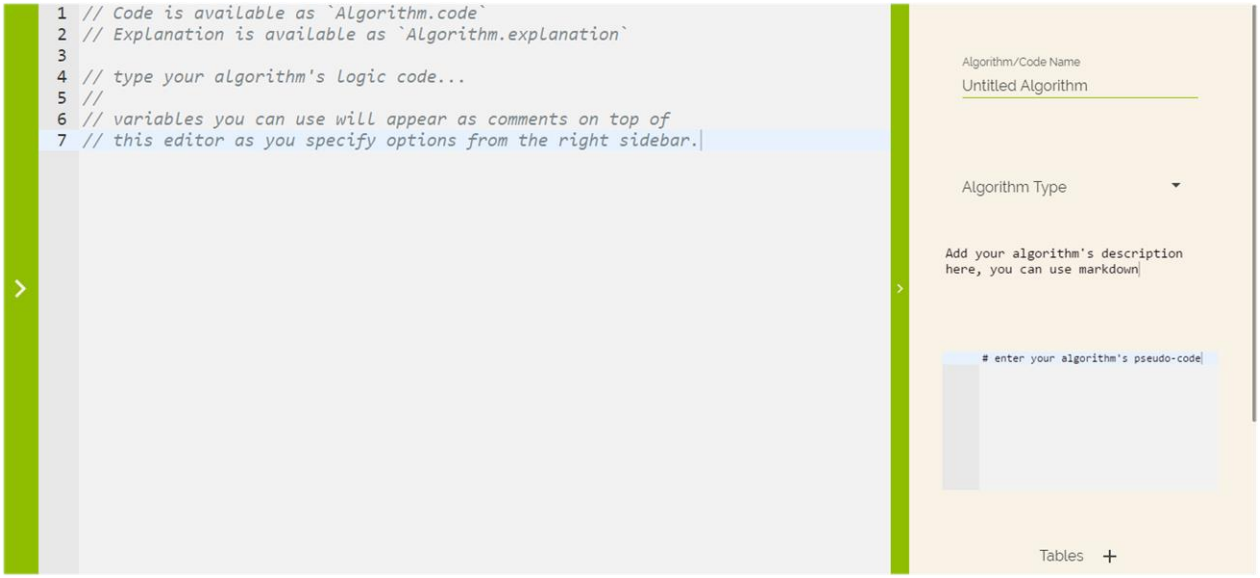
- Play: Starts the animation. Asks additional input if the algorithm needs it.



- 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.
- Speed Slider: The user can adjust how fast the animation is.
- Progress Bar: The whole bar is a progress bar, it will change color as the animation advances.

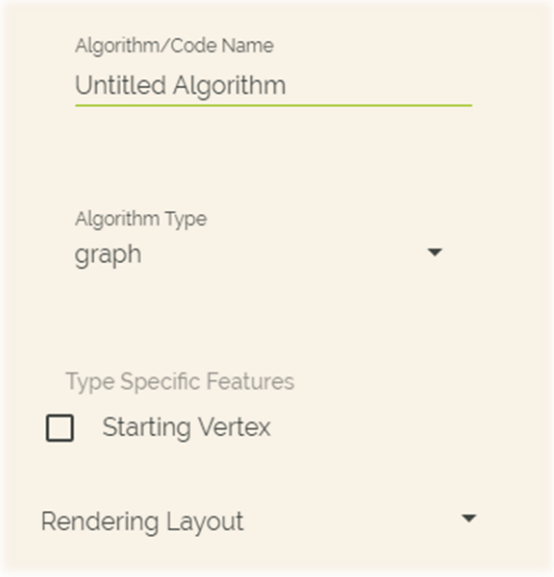
Using the Custom Code Mode

On the main page, clicking 'Custom Code' will lead you to the Custom Code page. To begin with the algorithm, the form on the right bar should be filled first. Filling this form will reveal which Visualization APIs you can use in your code; detailed information about the Visualization API can be found in the 'Usage' section of the app. There is an example of an implementation of DFS already in the app, which can be found on the left sidebar.



Type Specific Features

Selecting the type of the algorithm will add the main visualization component of that type and will also reveal additional more options about that type.

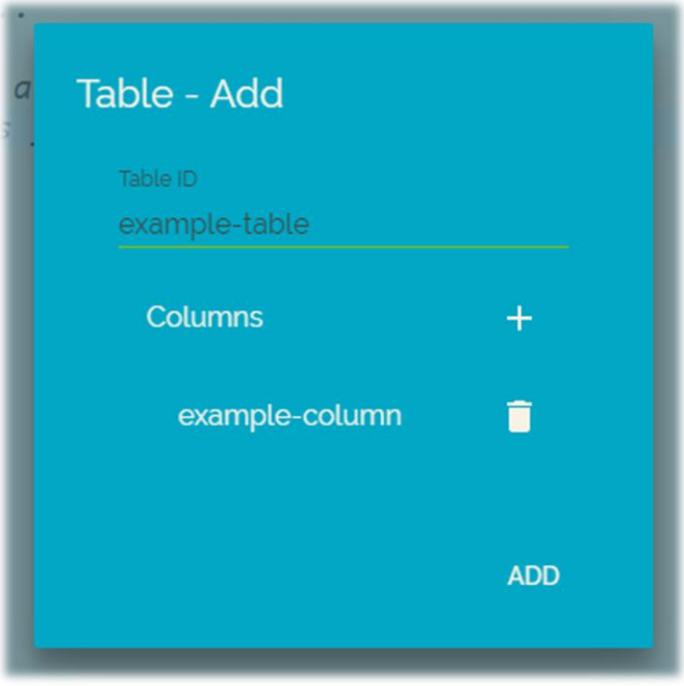


Description and Pseudo-Code

These text boxes are mandatory to fill. Description of your algorithm will be shown on the left sidebar, whereas pseudo-code is shown on the left sidebar.

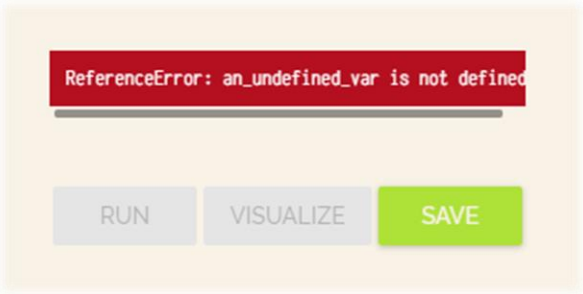
Adding Tables

Clicking on the plus sign right of the 'Tables' text will open a pop-up, in which the id of the table and names of columns must be entered. Clicking on the pencil sign right of a table id will edit that table. For your changes to be saved, you must push the save button.



Debugging

Your code is checked for errors constantly as you type it. These errors are located in the right sidebar. To run your code without visualizing, you may click on the run button. The run button will also enable special debug bindings, for example, you can `log(something)` in your code to have something printed out in a console on the right sidebar.

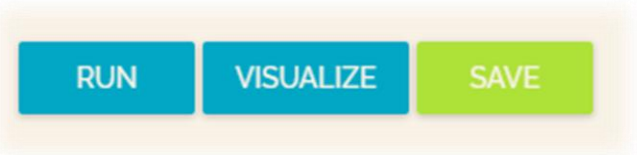


Saving and Loading Algorithms

Once you have progressed enough with your code, you will want to save it in order to use or develop it later again. Clicking on the 'Save' button saves your algorithm to your computer. You can load it back from the right sidebar.

Visualizing

When you are ready to visualize your algorithm, hit the 'Visualize' button. Doing so will ask you whether you want to save your changes if you changed your algorithm but haven't saved it yet.



Self Evaluation and 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 cared 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.

References and Resources

Book

- Data Structures and Algorithms, Rifat Colkesen, Papatya Publishing, October 2010

Websites

- Student life: Top ten most popular subjects
  www.telegraph.co.uk/education/educationpicturegalleries/10643255/Student-life-top-ten-most-popular-subjects.html
- Hacking Knowledge: 77 Ways to Learn Faster, a. (2006)
  http://oedb.org/ilibrarian/hacking-knowledge/
- React - A JavaScript library for building user interfaces
  https://facebook.github.io/react/electron.atom.io/
- Babel · The compiler for writing next generation JavaScript
  https://babeljs.io/
- JS-Interpreter Documentation
  https://neil.fraser.name/software/JS-Interpreter/docs.html
- ESLint - Pluggable JavaScript linter
  http://eslint.org/
- danielstjules/jsinspect. (2017) Github
  https://github.com/danielstjules/jsinspect
- webpack
  https://webpack.js.org/
- Mocha - the fun, simple, flexible JavaScript test framework.
  https://mochajs.org/
- lukehoban/es6features. (2017). GitHub
  https://github.com/lukehoban/es6features
- ECMAScript® 2016 Language Specification. (2017). Ecma-international.org
  http://ecma-international.org/ecma-262/7.0/#sec-samevaluezero
- Linkurious/linkurious.js. (2017). GitHub
  https://github.com/Linkurious/linkurious.js
- JavaScript reference. (2017). Mozilla Developer Network.
  https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference
- Publishing with electron-builder. (2016). Electron Rocks!
  http://electron.rocks/electron-builder-explained/
- verekia/js-stack-from-scratch. (2017). GitHub
  https://github.com/verekia/js-stack-from-scratch
- VisuAlgo - visualising data structures and algorithms through animation.
  https://visualgo.net/en
- Using loaders with webpack – VAMSI DEEPAK AMPOLU – Medium.
  https://medium.com/@deeepakampolu/using-loaders-with-webpack-f724e8d6469b

- Lecture 20: Topo-Sort and Dijkstra's Greedy Idea
  https://courses.cs.washington.edu/courses/cse326/03wi/lectures/RaoLect20.pdf
- What is a Failure in software testing?
  http://istqbexamcertification.com/what-is-a-failure-in-software-testing/
- Faster CI Debugging with GitlabCI | Bryce Fisher-Fleig
  https://bryce.fisher-fleig.org/blog/faster-ci-debugging-with-gitlabci/index.html
- Mistaeks I Hav Made: Refactoring JavaScript from Sync to Async in Safe Baby-Steps
  http://www.natpryce.com/articles/000812.htmlhttps://bost.ocks.org/mike/algorithms/

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.

Open Source Licenses

```
├─ babel-core@6.24.1
│  ├─ License: MIT
│  └─ URL: https://github.com/babel/babel/tree/master/packages/babel-core
├─ babel-eslint@7.2.3
│  ├─ License: MIT
│  └─ URL: https://github.com/babel/babel-eslint.git
├─ babel-loader@7.0.0
│  ├─ License: MIT
│  └─ URL: https://github.com/babel/babel-loader.git
├─ babel-polyfill@6.23.0
│  ├─ License: MIT
│  └─ URL: https://github.com/babel/babel/tree/master/packages/babel-polyfill
├─ babel-preset-es2015@6.24.1
│  ├─ License: MIT
│  └─ URL: https://github.com/babel/babel/tree/master/packages/babel-preset-es2015
├─ babel-preset-react@6.24.1
│  ├─ License: MIT
│  └─ URL: https://github.com/babel/babel/tree/master/packages/babel-preset-react
├─ babel-standalone@6.24.2
│  ├─ License: MIT
│  └─ URL: git+https://github.com/Daniel15/babel-standalone.git
├─ brace@0.10.0
│  ├─ License: MIT
│  └─ URL: git://github.com/thlorenz/brace.git
├─ chai-jest-snapshot@1.0.0
│  ├─ License: MIT
│  └─ URL: https://github.com/suchipi/chai-jest-snapshot
├─ chai@4.0.2
│  ├─ License: MIT
│  └─ URL: https://github.com/chaijs/chai
├─ classnames@2.2.5
│  ├─ License: MIT
│  └─ URL: https://github.com/JedWatson/classnames.git
├─ css-loader@0.28.1
│  ├─ License: MIT
│  └─ URL: git@github.com:webpack/css-loader.git
├─ dagre@0.7.4
│  ├─ License: MIT
│  └─ URL: https://github.com/cpettitt/dagre.git
├─ electron-builder-squirrel-windows@18.0.0
│  ├─ License: MIT
│  └─ URL: https://github.com/electron-userland/electron-builder.git
├─ electron-builder@18.1.0
│  ├─ License: MIT
```

```
|  └─ URL: https://github.com/electron-userland/electron-builder.git
├─ electron-reload@1.1.0
|  ├─ License: MIT
|  └─ URL: https://github.com/yan-foto/electron-reload.git
├─ electron-window-state@4.1.1
|  ├─ License: MIT
|  └─ URL: https://github.com/mawie81/electron-window-state.git
├─ electron@1.7.0
|  ├─ License: MIT
|  └─ URL: https://github.com/electron-userland/electron-prebuilt
├─ eslint-config-airbnb@14.1.0
|  ├─ License: MIT
|  └─ URL: https://github.com/airbnb/javascript
├─ eslint-import-resolver-node@0.2.3
|  ├─ License: MIT
|  └─ URL: https://github.com/benmosher/eslint-plugin-import
├─ eslint-import-resolver-webpack@0.8.1
|  ├─ License: MIT
|  └─ URL: git+https://github.com/benmosher/eslint-plugin-import.git
├─ eslint-loader@1.7.1
|  ├─ License: MIT
|  └─ URL: https://github.com/MoOx/eslint-loader.git
├─ eslint-plugin-import@2.2.0
|  ├─ License: MIT
|  └─ URL: https://github.com/benmosher/eslint-plugin-import
├─ eslint-plugin-jsx-a11y@4.0.0
|  ├─ License: MIT
|  └─ URL: https://github.com/evcohen/eslint-plugin-jsx-a11y
├─ eslint-plugin-react@6.10.3
|  ├─ License: MIT
|  └─ URL: https://github.com/yannickcr/eslint-plugin-react
├─ eslint@3.19.0
|  ├─ License: MIT
|  └─ URL: https://github.com/eslint/eslint.git
├─ exports-loader@0.6.4
|  ├─ License: MIT
|  └─ URL: git@github.com:webpack/exports-loader.git
├─ extract-text-webpack-plugin@2.1.2
|  ├─ License: MIT
|  └─ URL: http://github.com/webpack-contrib/extract-text-webpack-plugin.git
├─ file-loader@0.11.1
|  ├─ License: MIT
|  └─ URL: https://github.com/webpack/file-loader.git
├─ graphology@0.10.0
|  ├─ License: MIT
|  └─ URL: git+https://github.com/graphology/graphology.git
├─ html-loader@0.4.5
|  ├─ License: MIT
```

```
│  └─ URL: git@github.com:webpack/html-loader.git
├─ identity-obj-proxy@3.0.0
│  ├─ License: MIT
│  └─ URL: git+https://github.com/keyanzhang/identity-obj-proxy.git
├─ imports-loader@0.7.1
│  ├─ License: MIT
│  └─ URL: https://github.com/webpack/imports-loader.git
├─ inject-loader@3.0.0
│  ├─ License: MIT
│  └─ URL: git@github.com:plasticine/inject-loader.git
├─ jsdom-global@3.0.2
│  ├─ License: MIT
│  └─ URL: git+https://github.com/rstacruz/jsdom-global.git
├─ jsdom@11.0.0
│  ├─ License: MIT
│  └─ URL: https://github.com/tmpvar/jsdom.git
├─ jsinspect@0.12.4
│  ├─ License: MIT
│  └─ URL: https://github.com/danielstjules/jsinspect.git
├─ loaders.css@0.1.2
│  ├─ License: MIT
│  └─ URL: git@github.com:ConnorAtherton/loaders.css.git
├─ lodash.flattendeep@4.4.0
│  ├─ License: MIT
│  └─ URL: https://github.com/lodash/lodash.git
├─ lodash.isequal@4.5.0
│  ├─ License: MIT
│  └─ URL: https://github.com/lodash/lodash.git
├─ lodash.mapvalues@4.6.0
│  ├─ License: MIT
│  └─ URL: https://github.com/lodash/lodash.git
├─ lodash.pick@4.4.0
│  ├─ License: MIT
│  └─ URL: https://github.com/lodash/lodash.git
├─ lodash.pickby@4.6.0
│  ├─ License: MIT
│  └─ URL: https://github.com/lodash/lodash.git
├─ lodash.sample@4.2.1
│  ├─ License: MIT
│  └─ URL: https://github.com/lodash/lodash.git
├─ lodash.shuffle@4.2.0
│  ├─ License: MIT
│  └─ URL: https://github.com/lodash/lodash.git
├─ lodash.union@4.6.0
│  ├─ License: MIT
│  └─ URL: https://github.com/lodash/lodash.git
├─ lodash.uniq@4.5.0
│  ├─ License: MIT
```

```
│  └─ URL: https://github.com/lodash/lodash.git
├─ lodash.values@4.3.0
│  ├─ License: MIT
│  └─ URL: https://github.com/lodash/lodash.git
├─ lodash.without@4.4.0
│  ├─ License: MIT
│  └─ URL: https://github.com/lodash/lodash.git
├─ markdown-loader@2.0.0
│  ├─ License: MIT
│  └─ URL: https://github.com/peerigon/markdown-loader
├─ material-ui@0.18.0
│  ├─ License: MIT
│  └─ URL: https://github.com/callemall/material-ui.git
├─ mocha-webpack@0.7.0
│  ├─ License: MIT
│  └─ URL: https://github.com/zinserjan/mocha-webpack
├─ mocha@3.4.2
│  ├─ License: MIT
│  └─ URL: https://github.com/mochajs/mocha.git
├─ node-localstorage@1.3.0
│  ├─ License: MIT
│  └─ URL: http://github.com/lmaccherone/node-localstorage.git
├─ node-sass-json-importer@3.0.2
│  ├─ License: MIT
│  └─ URL: https://github.com/Updater/node-sass-json-importer.git
├─ node-sass@4.5.2
│  ├─ License: MIT
│  └─ URL: https://github.com/sass/node-sass
├─ null-loader@0.1.1
│  ├─ License: MIT
│  └─ URL: https://github.com/webpack/null-loader.git
├─ prop-types@15.5.9
│  ├─ License: BSD-3-Clause
│  └─ URL: https://github.com/reactjs/prop-types.git
├─ react-ace@5.0.1
│  ├─ License: MIT
│  └─ URL: http://github.com/securingsincity/react-ace.git
├─ react-dom@15.5.4
│  ├─ License: BSD-3-Clause
│  └─ URL: https://github.com/facebook/react.git
├─ react-hot-loader@3.0.0-beta.7
│  ├─ License: MIT
│  └─ URL: https://github.com/gaearon/react-hot-loader.git
├─ react-markdown@2.5.0
│  ├─ License: MIT
│  └─ URL: git@github.com:rexxars/react-markdown.git
├─ react-promise@1.1.3
│  ├─ License: MIT
```

```
│  └─ URL: git+https://github.com/capaj/react-async.git
├─ react-test-renderer@15.5.4
│  ├─ License: BSD-3-Clause
│  └─ URL: https://github.com/facebook/react.git
├─ react@15.5.4
│  ├─ License: BSD-3-Clause
│  └─ URL: https://github.com/facebook/react.git
├─ sass-lint@1.10.2
│  ├─ License: MIT
│  └─ URL: https://github.com/sasstools/sass-lint.git
├─ sass-loader@6.0.5
│  ├─ License: MIT
│  └─ URL: git://github.com/webpack-contrib/sass-loader.git
├─ sass-material-colors@0.0.5
│  ├─ License: MIT
│  └─ URL: https://github.com/minusfive/sass-material-colors.git
├─ sigma@1.2.0
│  ├─ License: MIT
│  └─ URL: http://github.com/jacomyal/sigma.js.git
├─ style-loader@0.16.1
│  ├─ License: MIT
│  └─ URL: git@github.com:webpack/style-loader.git
├─ url-loader@0.5.8
│  ├─ License: MIT
│  └─ URL: git@github.com:webpack/url-loader.git
├─ webpack-config@7.0.0
│  ├─ License: Apache-2.0
│  └─ URL: https://github.com/Fitbit/webpack-config.git
├─ webpack-dev-server@2.4.5
│  ├─ License: MIT
│  └─ URL: git://github.com/webpack/webpack-dev-server.git
├─ webpack-node-externals@1.6.0
│  ├─ License: MIT
│  └─ URL: https://github.com/liady/webpack-node-externals.git
├─ webpack@2.5.1
│  ├─ License: MIT
│  └─ URL: https://github.com/webpack/webpack.git
```

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)
Attended a local JavaScript developers meeting, learned there the importance of using a bundler to bundle the source code, and made connections. (Know)
Learned how to use Continuous Integration in order to automate tasks like packaging. (Know)
Learned necessary procedures about how to package an app for 4 different platforms using electron, Web, Windows, Linux and OSx (Know)
Organized a live testing session for AlgoriVA to the members of schools Olympiad and Robotics Club. (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)
Attended Startup Carnivals at Istanbul Technical University and Bogazici University. (Know)
Encouraged other state members to attend trainings this year. (Do)
Leadership Roles
Presidents of robotics club and science club. (Be)