Fun With Algorithms

Armen Ourfalian

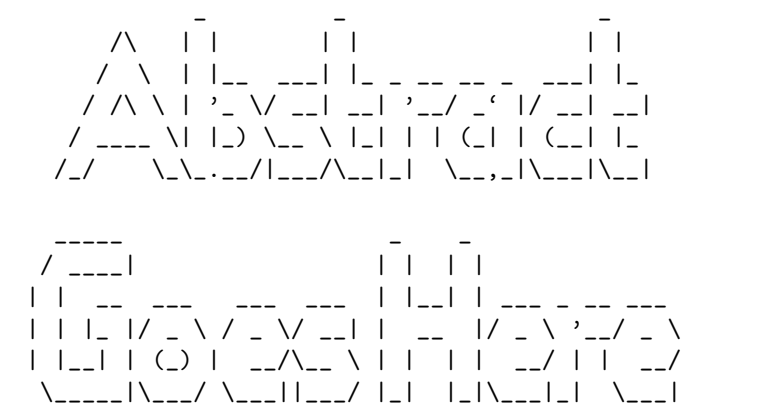
March 31, 2018

Contents

List of Figures

List of Tables

1 Abstract



2 Introduction

2.1 Motivation

When I was a high school Math and Science teacher, I often tried to employ the use of technology in the classroom. I often found it to be a good way to spend less class-time on tasks that do not engage the student. An example of such a task is drawing a graphs (in Algebra class): while I was busy drawing the diagram, students would not engaged in the lesson.

So I started preparing graphs on a computer to display over a projector, and saved. valuable minutes of engaging class time, and a side benefit of using computer-generated images was that they looked far better than any graph I could have drawn by hand. The only problem with this solution was that I had to spend more of my own time outside of class to create these visuals, or find some online.

Due to my background as a teacher, I was drawn towards a thesis project where I could create an educational tool to be used in a classroom to help instructors draw diagrams and display them over a projector instead of having to draw them on the whiteboard. The overall goal of this project is to create a tool that will save time in class by reducing the teacher's non-engaging tasks, while not requiring too much of the teachers' time outside of class for preparation. For the subject matter, I chose Algorithm Visualization.

2.2 Background

Algorithms are a fundamental part of Computer Science education. Just about every introductory CS class discusses the various sorting algorithms, their advantages and disadvantages, followed by some video that graphically shows how each sorting algorithm is implemented such as [1]

Algorithm Visualization (AV) is the use of software to create diagrams, animations, and other visual tools to facilitate the learning of an algorithm, its process (how it works and why it produces the correct result), and its complexity (runtime, required space, etc). There is an abundance of visualizations for sorting algorithms, and a decent amount for other introductory algorithms or data structures (Binary Search Trees, Linked Lists, Kruskal's Algorithm, Primm's Algorithm, etc.), but fairly few and far between for the more intermediate-level algorithms.

Studies have shown the positive effects of various AV's (see **Section 3 Literal Review**), and

yet they are seldom used as part of the teaching process. Some of the advantages Algorithm Visualizations provide are:

- Instructors can display complex diagrams or data structures without spending a lot of time to draw them
  - This would increase the amount of time they spend engaging the class
  - This would also allow instructors to cover more complex examples than they otherwise would be able to
- Students can use the visualizations whenever they like
  - This would allow them to reproduce the problem outside of class
  - This would give them access to a more guided environment to review than just studying on their own

2.3 Objectives

1. Create an AV tool to be used by CSUN Faculty teaching an intermediate Algorithms class during lecture
   (a) Algorithms covered by this project will be drawn from those taught in Comp 482
   (b) The main use case is on a projector display as part of a lecture
2. The tool must be intuitive and easy to use
   (a) Users should be able to use the tool with little or no formal training
   (b) Using the tool to create diagrams should take no more time than drawing a similar diagram by hand
3. The tool will have the following features:
   (a) Allow the user to create instances of a given problem
   (b) Allow the user to simulate the steps of an algorithm and see the solution
   (c) Display visuals to describe how the algorithm is running

   A more detailed list of requirements can be found in **Section 4 Requirements**

2.4 My Approach

The ultimate goal of this project is to create a tool that will actually be used in the classroom. The target users are CSUN faculty, and the target environment is in a classroom during lecture. I kept this in mind when I was planning, designing, and testing my project. Whenever a decision had to be made between making the tool more universally applicable versus making the tool a better fit for the specific target audience, I chose the latter. For example: that is why I chose specific lessons that were covered in CSUN's Comp 482 class.

In order to make the the tool easily accessible (by faculty as well as students), it was designed as a web app, more precisely as a front-end single-page application:

- Front-end: the app runs on the client's local machine
- Single-Page application: when the user interacts with the app, it changes the page dynamically without having to reload or refresh
- Application: computer program that accomplishes some task

Each algorithm within the app is a self-contained module (ie does not rely on other modules) comprised of

- An Instance Maker: allows the user to create instances of a given problem.
- A Solver: allows the user to run the algorithm.
- A Display: allows the user to see the various aspects of the problem while the algorithm is being performed.

I used a JavaScript framework called Vue with an MVC approach. I chose Vue because it is a relatively new framework and its popularity is on the rise [2]. Furthermore, I used a popular Vue library called Vuex, which is a state-management library [4] I hosted the app on Heroku, because they offer free hosting for hobby-level apps. The app can currently be accessed by going to the following webpage:

https://fun-with-algorithms.herokuapp.com

3 Literature Review

(in progress)

- Discuss various AV products available, what their strengths/limitations are
  - Academic Studies → Also discuss why they were made/how they were used in the study
    - HalVis
    - ANIMAL
    - BRIDGES
    - Mention a few others
  - Available on the web
    - Dijkstra's Algorithm
    - Sorting out Sorting
    - https://visualgo.net/en
    - https://www.cs.usfca.edu/~galles/visualization/Algorithms.html

4 Requirements

1. Create a web-app for Algorithm Visualization
   (a) The app will be a Single-Page Application
   (b) The app will run in the front-end
   (c) The app will run in the Google Chrome web browser
2. The target users of The app will be CSUN faculty who are teaching an intermediate-level algorithms class (such as Comp 482)
3. The app must be easy to use by its target users
   (a) Users should be able to interact with the app with little to no formal training
   (b) Where necessary, the app must provide instructions on any controls that do not meet the above criteria
4. The app will provide AV's for 3-5 algorithms covered in such a class, including:

(a) Stable Matching

(b) Interval Scheduling (unweighted)

(c) Closest Pair of Points

5. For each algorithm, an AV will have the following:

   (a) An interface to create/modify instances of the problem

      i. User controls displayed on the webpage (text boxes, buttons, etc.)

      ii. Loading an instance from a .txt file

      iii. Saving an instance into a .txt file

      iv. Loading an instance from a database

   (b) An interface to solve the created instances (ie run the algorithm)

      i. Perform a single step of the algorithm at a time

      ii. Perform the entire algorithm automatically.

   (c) A visual diagram of an instance of the problem

      i. The diagram must be organized and easy to understand

      ii. The diagram must be appropriately sized to be displayed on a projector in front of students

      iii. The display must update as the solver is performing the algorithm

5 Development Process

Since I was working by myself on this project, I did not adopt a formal development process, but it was iterative, with multiple deliveries, and continued improvement. The nature of this project is very modular. It consists of various algorithms which do not rely on one another. Thus I worked on each algorithm individually from conception to completion and published it to the live website before moving on to the next algorithm. The process of creating an AV for a given problem went (roughly) as follows:

- Decide acceptable ranges and restrictions for inputs and problem size
  - These restrictions ensure that the app will run smoothly and not slow down
  - The restrictions also make sure the visualization is easy to understand, and fits on the target screen size
  - The ranges should be larger than if the problem were being solved by hand
- Create the data model
  - What data structures are required for a given type of problem
  - What properties and methods are required to run an algorithm
  - Which parts of the data should be made available to the display
- Create the Instance Maker
  - User interface (text fields, buttons, sliders, etc.) to construct an instance manually
  - Save/Load interface through which the user can upload/download a text file to create an instance
  - Predefined example instances that can be loaded without manually being inputted (this is a nice-to-have)
  - Update the data model as the user makes changes to the problem instance

- Create the Display
  - Visual components that represent the data model
  - Must respond to changes in the data model
- Create the Solver
  - User interface to run the given algorithm
  - Update the data model as the algorithm is performed
- Show Professor Noga the progress
  - Discuss various decisions/restrictions/compromises made and whether they are acceptable
  - Make any necessary changes

6 Tools and Technologies

(in progress)

6.1 Vue js

- What is it? (front end framework that utilizes virtual DOM and creates modular reactive components)
- What alternatives are there? Why did I choose this one?
- Examples of how it works
- How I organized my project

6.2 Vuex

- What is it? (state management library for Vue)
- Why did I use it?
- How it works (state, getters, mutations, actions, modules)
- How I utilized it

6.3 Heroku

- What is it? (web hosting service)
- Why did I use it? (it's free)

6.4 Database

- Why would I need a database? (store problem instances)
- Mention it's not a requirement, it's a nice-to-have
- Mention how I used a database
- Mention other ways a database could be useful for this project (user/login system that allows users to save instances on database instead of relying on text files)

7 Design

(in progress)

7.1 Architecture

- Discuss the use of Vue to organize the web app
- Discuss the use of Vuex to keep the model/controller and the view separate

7.2 Algorithm Pages

- For each algorithm in the project:
  - Discuss the design of the page
  - Discuss the properties (state)
  - Discuss the private methods (mutations)
  - Discuss the public methods (actions)
  - Discuss the view (components)

8 Testing

(in progress)
Mention the amount of testing done was limited due to time constraints

8.1 Unit Testing

Discuss the usefulness of unit testing and which parts of the app were tested

8.2 In-Class Presentations

Discuss the in-class demos and how students (and Professor Noga) responded to it

9 Guide

(in progress)
Discuss how the app is used, some hidden features etc.

- For each algorithm in the project:
  - Show images of the webpage (locked and unlocked mode)
  - Discuss what the various controls do
  - Discuss how the page is supposed to be used
  - Discuss how the save/load functionality works
  - Discuss any other interesting things about it

10 Conclusions

Conclusions goes here

11 Sources

1. Sorting out Sorting Video
2. Top JS libraries (https://medium.com/javascript-scene/top-javascript-libraries-tech-to-learn-in-2018-c38028e028e6)
3. Vue website
4. Vuex website
5. John Domingue, Erkki Sutinen. **Software Visualization**. *Journal of Visual Languages and Computing*. 2002.
6. Tobias Lauer. **Reevaluating and Refining the Engagement Taxonomy**. *ITiCSE*. 2008.
7. Christopher Hundhausen, Sarah Douglas. **Shifting from "High Fidelity" to "Low Fidelity" Algorithm Visualization Technology**. *Conference on Human Factors in Computing Systems*. 2000.
8. Guido Rling, Thomas L. Naps. **A Testbed for Pedagogical Requirements in Algorithm Visualization**. *ITiCSE*. 2002.
9. Guido Rssling, Bernd Freisleben. **ANIMAL: A System for Supporting Multiple Roles in Algorithm Animation**. *Journal of Visual Languages and Computing*. 2002.
10. Steven Hansen, N. Hari Narayanan, Mary Hegarty. **Designing Educationally Effective Algorithm Visualizations**. *Journal of Visual Languages and Computing*. 2002.
11. Christopher Hundhausen, Sarah Douglas, John Stasko. **A Meta-Study of Algorithm Visualization Effectiveness**. *Journal of Visual Languages and Computing*. 2002.

12 Appendices

Appendices goes here