NEA Project Analysis

Jacob Halleron
The Sandon School

Background

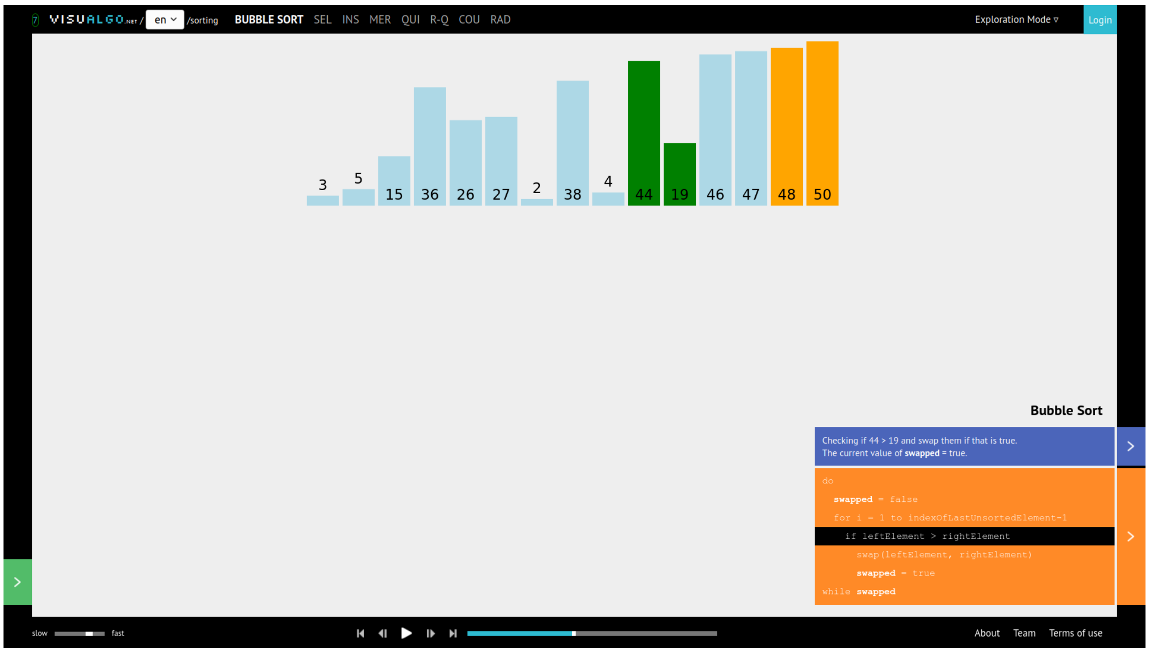
I'll make the assumption that the reader has basic knowledge on what algorithms are and how they process sets of data. Computer Science students always have and always will need to intricately understand how a variety of algorithms and and data structures process data, their pros and cons, how they relate to others, etc. These concepts are usually completely abstract and difficult to understand, especially with all the other topics crammed into students' memory. Teaching of this at GCSE and A-level is currently often limited to text-based explanations and code blocks, which can be hard to understand with more complex algorithms. For example, I failed to understand Dijkstra's Algorithm, which isn't too difficult a concept, until I came across a step-by-step video explaining the process with a graph diagram and code. The same could likely be said with the selection of other algorithms students may need to know.

Nick Miller has asked me to create an online solution that will help him and other CS students to learn these algorithms. Rather than tell the user the answers in an intimidating block of text, the user will teach themselves by following an animated diagram.

Market Research

There are a few other products on the web.

VisuAlgo



Bubble Sort

Checking if 44 > 19 and swap them if that is true.
The current value of **swapped** = true.

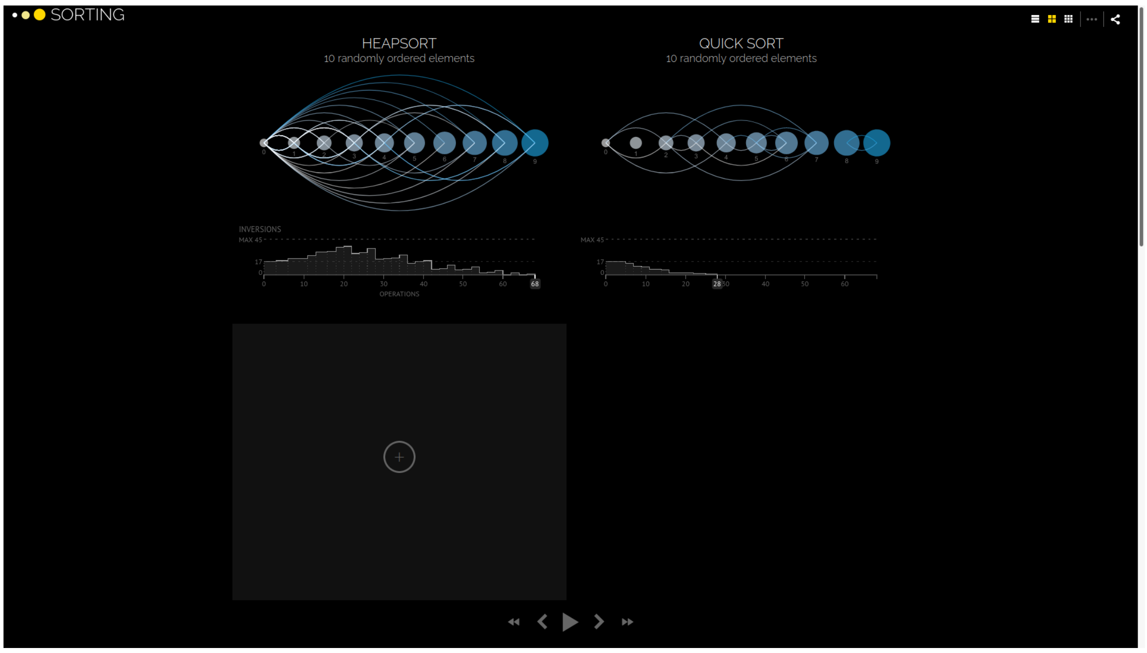
```
do
  swapped = false
  for i = 1 to indexOfLastUnsortedElement-1
    if leftElement > rightElement
      swap(leftElement, rightElement)
      swapped = true
while swapped
```

Visualgo is a website which shows algorithms, but also functions for controlling data structures.

This one has options to visualise almost any algorithm or data structure, all accessible from an extensive main menu. When I clicked on an animation for bubble sort, a large pop-up block of text told all about sorting and how it works, but the average user probably wouldn't read all that, only skip to the animation itself. When running, there is a handy pseudocode block on the right, and the speed of the loop can be changed, stopped or made to go step-by-step on click. However, there are a range of features which are confusing and hard to scan, like the navbar containing a list of other algorithms at the top, or the strange yellow box on the left whose controls a typical student wouldn't understand or need to use.

Charts and diagrams in this product are done with an SVG image, and rectangle elements within it are moved around by scripts altering the `transform` attribute. The site also uses the JQuery library, which I have elected to not use.

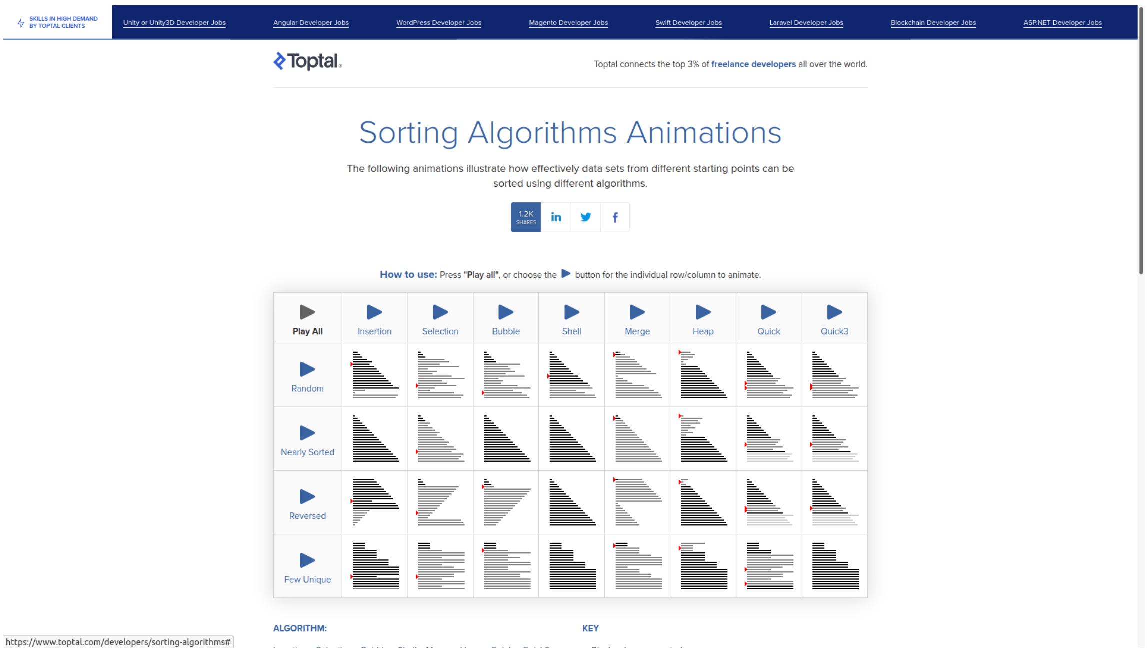
SORTING



This one is a more aesthetic product which only shows sorting algorithms. It's purpose is mainly to look pretty rather than be educational.

The site does achieve its goal of looking nice and artistic, with its dark theming and extensive whitespace. It also has the option to speed up or slow down the tick rate, and you can also compare different algorithms in real-time. However, its not particularly functional as data sets of more than about 20 are very hard to read, and the text can be very small, especially in the unlabelled charts, which I think is supposed to plot the number of inversions over time (It's not the most user-friendly).

Toptal


Toptal connects the top 3% of freelance developers all over the world.

Sorting Algorithms Animations

The following animations illustrate how effectively data sets from different starting points can be sorted using different algorithms.

How to use: Press "Play all", or choose the ▶ button for the individual row/column to animate.

Toptal is a website for finding freelance developers, but it also has a page for a neat animated table showing sorting algorithms.

Like sorting.at, it only shows sorting algorithms. It can also show multiple at once, but unfortunately there's no option to view just one algorithm. It also has very limited information on how they work, their efficiency or how they compare to others.

If you look at the inner workings, there aren't actually any algorithms running; instead the diagrams are pre-rendered images. That means the data will always look the same, nothing is randomised, and there is no option to control speed and change the size of the data set. It has no step-by-step explanation.

My solution will fix these problems with a clean, simple interface and the ability to view different types of algorithm. There will also be the option to compare different algorithms.

End User / Target Audience

The end user and target audience are the same for this. The product would be targeted at English-speaking A-level and GCSE Computer Science students, so age 14-18. Any device should work as long as it can run a web browser with scripting enabled. Most can, but school computers are infamous for being slow

Research

I've emailed Nick (the client) a few questions about the project and the objectives I can derive from it. They're fairly closed questions, but they will certainly prove valuable for planning and organisation. Below is our conversation.

```
1. Do you struggle with learning algorithms in GCSE or A-level -- Ye
2. How capable is your hardware and OS? -- My system is very good,
3. What platform would you prefer the software run on? -- Windows
4. Is the user interface important to you? -- Yes. If there is a bad
5. Would it be important to compare and contrast animations with eac
6. Is it important to have a code block or explanation of the proces
7. Would you want to be able to control the speed of the animations,
```

Objectives

1. The product must effectively teach students how the algorithms work. This one's fairly vague but the others should provide some detail.

2. It should be able to run on almost all computers. The client's system is very above average, and isn't a typical setup the average student would have, but slowed-down bubble sort shouldn't be too resource-intensive.

3. It must work on most operating systems. Windows is the main one, but web browsers tend to be the same on all platforms. I'll test it on Chromium, Safari and Firefox since almost everyone uses them or a derivative. Since it's made for the web, I'll also need to think about server-side stuff.

4. The UI must be simple and intuitive. It's aimed at minimum 14 year olds, who might not have advanced technical expertise. I think my CSS skills are good enough; my aim is a grid layout with some fairly plain-looking grey boxes and a very colourful diagram so that the focus is on the animation rather than unimportant details.

5. There must be a feature to compare multiple animations side-by-side in parallel. I'm not yet sure if the code blocks will still be visible, or if the controls will be universal for every animation or specific to each.

6. A code block must be available beside the animation, ideally with the current step highlighted. This will aid the learner. For readability, I'll probably write it in Python or pseudocode so it's easier to read so it will be separate from the actual internal JS.

7. The speed of the animation will be adjustable. I'll put a separate control box below the diagram, which will contain a play/pause button, next/previous buttons, an input box for the tick rate, and a reset button.

8. This isn't based on his answers, but I'd like there to be some data on each algorithm, i.e. the big O and efficiency with different given inputs.

Proposed Solution

The solution will be a website with a complex client-server model. Javascript, HTML and CSS will be used, along with some basic Node.js for the server. In terms of development software a browser, text editor and terminal should do. I'm using a Node.js server running the Express framework, as I've had some experience with it and found it to be simple and capable. The only real way to do the front-end development is vanilla Javascript with HTML & CSS, so that's what I'll use. Here are the planned stages of my design:

1. Express.js server process. I'm still a bit conflicted over Node/Express vs. Apache/PHP, as the latter is specifically designed for server-side web development and the project isn't too complicated when it comes to the server. I'll only need some basic templating for the repeat elements (header, footer, etc.) and the animations, so I can essentially have the same page but with different data and imported scripts.

2. Site design using HTML/CSS. This has much less priority as I'm not actually being graded on how aesthetic my web design is. I might skip it and come back later.

3. Homepage / Menu. There must be a page where users can select an algorithm, which I'll do as a big unordered list with images. They might even animate on hover if I'm feeling spicy.

4. Resizeable box to show data structure, different for arrays, trees or graphs. It can be updated to render an object in the JS runtime and this will happen every time the algorithm ticks. This will be the most time-consuming part of the project. For increased immersion, I'll also have sound effects on the sorting algorithms in the form of a beep with pitch corresponding to array index.

5. Control box which alters the tick rate of whatever algorithm is running. I might need to do some funky coding in the JavaScript realm so that each algorithm's speed can be controlled, probably using a variable to show the current step of the loop.

6. Algorithms themselves:

- Sort (array)
- Pathfind (graph)
- Search (array)
- Graph/tree traversal

   Each one will probably be a single function that executes one step of the process, unless i decide to go with a for-loop that just runs until it's done.

Model

Improper sitemap: - Menu page - Single simulation page - Diagram - Controls: - Play/Pause button - Next/Prev buttons - Tick speed input box - Reset (randomise) - Local variables - Code Block - Big O and data, potentially with chart - Multiple simulation page - List of diagrams - Step counter - Controls - Data on each of their speeds - About page