Software development project

William Roebuck

Contents

[Analysis 1](#_Toc113622736)

[The problem 1](#_Toc113622737)

[Computational methods 2](#_Toc113622738)

[Object oriented programming 2](#_Toc113622739)

[Functional and procedural programming 2](#_Toc113622740)

[Visualisation 2](#_Toc113622741)

[Heuristics and backtracking 2](#_Toc113622742)

[Stakeholders 2](#_Toc113622743)

[Esteban Garcia 2](#_Toc113622744)

[Paula Weston 2](#_Toc113622745)

[Harry Thorne 2](#_Toc113622746)

[Research and existing solutions 2](#_Toc113622747)

[CS academy graph editor 2](#_Toc113622748)

[Graph online 3](#_Toc113622749)

[Essential Features 4](#_Toc113622750)

[Limitations 5](#_Toc113622751)

[Hardware and software requirements 5](#_Toc113622752)

[Success Criteria 5](#_Toc113622753)

[Design 6](#_Toc113622754)

[Decomposition 6](#_Toc113622755)

[Structure 6](#_Toc113622756)

[Algorithms 6](#_Toc113622757)

[Usability Features 6](#_Toc113622758)

[Data Structures 6](#_Toc113622759)

[Test Data 6](#_Toc113622760)

[Post development 6](#_Toc113622761)

[Implementation 6](#_Toc113622762)

[Evaluation 6](#_Toc113622763)

[Testing for function 6](#_Toc113622764)

[Testing for robustness 6](#_Toc113622765)

[Testing for usability 6](#_Toc113622766)

[Success Criteria 6](#_Toc113622767)

[Further development 6](#_Toc113622768)

[Maintenance 6](#_Toc113622769)

# Analysis

## The problem

Graphs are useful for a wide variety of applications, including computer networking and urban planning, however there are very few general-purpose node programs that can be used without significant programming experience. The aim of my program is to be easy to set up and intuitive to use, while still maintaining a high level of functionality.

This program will allow the user to create, edit and view node networks, including algorithms such as Dijkstra’s, A\*, Kruskal’s, Floyd’s, brute force travelling salesman, and the planarity algorithm. The networks can be weighted and/or directional. The program should save graphs to a file and can load graphs from files.

## Computational methods

This problem is suitable for a computational solution mainly to exploit the benefits of abstraction. Since the computation will be done for the user, a high level of expertise is not needed to understand the outputs of the program and visualisation will aid this.

### Object oriented programming

OOP will play a large part in this project’s development as I intend to implement a “Graph” class that will be used for almost all of the program’s features. This has the benefit that the code is reusable if I create a project in future that includes graphs but it is also a useful abstraction, in that it is more intuitive to access the parts of a graph (Graph.Count, Graph.Weight) as attributes rather than as standalone variables (graphCount, graphWeight) which will help with future development as well as making code easier to understand. Furthermore, the class will have many methods, at least one for each graph algorithm that can be used, so that the graph and its data do not have to be passed into a function outside of the class which improves security.

### Functional and procedural programming

Functional and procedural programming will also be important for this project for handling most of the user driven parts of the program, including the UI. These subprograms will allow for event driven code (through windows forms) and make code more human readable.

### Visualisation

Visualisation is essential to the program and is key to making the program user friendly. Graphs will be visualised as soon as they are created/loaded and will update to reflect changes made by the user. It will also be used to show the outputs of algorithms such as Dijkstra’s by highlighting the given route

### Heuristics and backtracking

Heuristics will be used as part of A\* (which improves performance) whereas Dijkstra’s algorithm will use backtracking.

## Stakeholders

### Esteban Garcia

Esteban Garcia is a 57-year-old urban planner who is not fluent in English and requires the program to allow accents such as á, é, í, ñ, ó, and ú as inputs. He is familiar with graph structures but usually uses paper to work with them. He needs the program to be flexible in what it can display and calculate. As he may use the program in his work, he needs it to be robust and save data in the case of a crash, which should be a rare occurrence if it does ever happen.

### Paula Weston

Paula Weston is a partially sighted 17-year-old further maths student who requires the program to have a dark colour palette and an option to increase font size for the whole program. She needs to be able to create graphs quickly and wants a detailed output from each algorithm, that shows most of the steps taken by it so that they can be worked through.

### Harry Thorne

Harry Thorne is a 34-year-old professional cyclist who requires the program to be easy to use and understand without studying computer science or further maths. He will use the program to plan his routes and needs the nodes and edges to be able to store and display long multi-word names such as “Barton-upon-Humber” or “Cammeringham Hill”. Harry also needs the program to be able to handle large, detailed networks without becoming unusable or unstable.

## Interview

**Do you have any hardware limitations?**

**Garcia:** I have not upgraded my hardware since 2010 and I am used to it being slow. As long as the program is responsive enough so that I can use it, it will be fine.

**(After some discussion, we came to the conclusion that for his PC, with 4GB of RAM, the response time should be no more than 2 seconds for any graph manipulation)**

**Weston: My PC is built for the latest videogames, so it should be able to run most things, but I dislike when the fan starts making a lot of noise, so I hope the program isn’t too intensive**

**Thorne: I have a somewhat modern laptop; it can run many tabs of Google Chrome without issues but it sometimes slows down during intensive physics simulations**

**How familiar are you with computer interfaces and how do you like to interact with them?**

**Garcia:** I use my company’s software on my computer, and most of it is not very user friendly so it takes a long time to get used to. I would prefer if I could just click buttons and not have to find everything in verbose menus and sub-menus.

**Weston:** I’ve used a lot of different programs for installations and niche applications so I am quite used to confusing user interfaces. I like to use my keyboard to navigate around the interface as its more convenient.

**Thorne:** I mostly use my computer for YouTube and Google but have used some physics software. I prefer visual representations in the user interface since I find icons easier to remember than words.

**At what scale are you likely to use the program?**

**Garcia:** My node networks are often between 25 and 50 nodes however there always far less edges than there could be (more like 300 connections instead of 1225)

**Weston:** I’d only need the program for really small networks; less than 20 nodes and 30 edges

**Thorne:** I don’t think I’d use any more than 15 nodes, but there’d be a lot of connections between them, about 100

**Would you like to name every node and edge?**

**Garcia:** Naming the nodes will be essential but I would only rarely name edges

**Weston:** The names of nodes and edges don’t matter to me; I’d prefer if each node was automatically named A-Z and the edges left blank

**Thorne:** I might not name every edge, but certainly the important ones and each node must have a name

## Research and existing solutions

### CS academy graph editor

<https://csacademy.com/app/graph_editor/>



This web-based solution is an effective visualisation tool for node graphs, allowing the user to “draw” a graph by clicking and dragging on nodes. Adding nodes is efficient and intuitive as one must click an empty space and then simply click and drag to create a connection between them. Additional details such as weight and direction can be added later by clicking on the connection. The program is somewhat lacking in functionality in that it can only display a graph, not make any calculations using one.

The program also has a range of customisation features, including the ability to change node colour, size, and other options, however there is no way to increase font size.

Graphs can only be saved and loaded by copying the data textbox on the left into a file which makes managing a large group of graphs difficult.

From this I will incorporate the click and drag functionality and the elegant interface that becomes more complex as more information is added

### Graph online

https://graphonline.ru/en/



Graph online is another web-based solution for working with graphs but with a focus less on visualisation and more on algorithms. It offers 20 different useful algorithms that can be performed on your graph, selectable from a dropdown above the graph.

However, the program is hard to use in that to add a new vertex one must first select “Add vertex”, click on the graph, then “Connect vertices” and click on the two vertices. Then, a popup box appears that must be filled in each time a connection is made. In comparison to the CS academy graph editor, it is far slower and more tedious to use.

I will try to avoid using dialogue boxes in my program as they interrupt the creation process.

Have a look at Gephi

## Essential Features

From my research and discussions with stakeholders, I have decided on these essential features:

* UI accessible algorithms to be performed on user’s graphs: Dijkstra’s, A\*, Kruskal’s, brute force travelling salesman, Floyd’s and the planarity algorithm – These are important to the functionality of the program so that it can be used in the wide range of applications desired by my stakeholders. I have chosen to include Dijkstra’s **and** A\* because for some applications such as city planning, a large number of nodes will be required and Dijkstra’s may be too slow to remain effective at these scales.
* Ability to create, save, name, rename, and delete graphs (storing this data on file(s)) – This is important so that users can work on multiple projects concurrently
* Graph manipulation capability: Adding, removing, moving, and naming nodes; connecting nodes, naming, removing, and automatic/manual weighting connections. – Important for visualisation and the function of the program
* Accessibility settings: Font size, colour scheme

## Limitations

Some solutions that I researched offered a “gravity” feature, where the nodes in the network automatically arrange themselves in a geometric way so that they are evenly spaced out. While this is useful for abstract applications where the physical layout of the nodes is not important, my program is primarily concerned with problems in the real world so will not include this.

One accessibility option that I have not accounted for is a screen reader to explain what the program is doing for blind users. I have decided not to implement this because one of the main features of the program is visualisation and there are other solutions which would be better suited to a purely numeric implementation. Furthermore, Windows 10, the operating system the program is designed for, comes with a screen reader so an additional one is not necessary.

Another potential feature would be language options; however, these should not be necessary as the program’s main purpose is visualisation so there will be more symbols than English language. Furthermore, all the algorithms have names or are named after somebody so will not need to be localised. The program will support user inputs from other languages, so the user can type with their language if they wish

## Hardware and software requirements

To compile the program, I will need:

* Microsoft Visual Studio (which requires windows 10+)

The user will need:

* Windows 10[[1]](#footnote-1)
* 4 GB or more of RAM[[2]](#footnote-2)
* 1 GHz or faster processor[[3]](#footnote-3)
* Built in microsoft libraries

## Success Criteria

Note to self: Specific, measurable, achievable, realistic, timely

|  |  |  |
| --- | --- | --- |
| No. | Criteria | Justification |
| 1 | The user can create and manipulate graphs with GUI controls | This is an essential feature for the stakeholders’ use of the program |
| 2 | The user can run algorithms on their graphs that display an output |
| 3 | Initially, nodes should be named A-Z but edges should be left blank |
| 4 | The user can save and load graphs to/from files | A stakeholder has specified that the program must be suitable for long term, multi-project use |
| 5 | The user can name edges and nodes | Stakeholders require this feature as it is important for the program’s use to model real situations |
| 6 | Users can input Unicode which will be displayed with the same font, colour, size and spacing | A stakeholder has specified that the program must support languages with extended alphabets such as Spanish |
| 7 | If an error occurs, the user should be shown what has caused it | A stakeholder has said that the program may not be perfect, but that the user should be able to see what has gone wrong to avoid future errors |
| 8 | The program must have accessibility options to ensure suitability for partially sighted users | A stakeholder requires these features to use the program |
| 9 | In the event of a crash, the program should be able to recover a recent copy of the most recent graph if it has not been saved | This is important to stakeholders as the program must be reliable so that it can be used in industry |
| 10 | The program must be sufficiently stable and responsive (no more than 2 second delay) with large networks up to 50 nodes and 300 edges in size | Stakeholders may use the program for very large projects so it must be functional as scale grows |
| 11 | The program must be usable without any English language, (excluding algorithm names). I will ask my stakeholders to evaluate whether this criterion is met | Not all of my stakeholders are fluent in English, so I should ensure that the program can be used without relying on English |

# Design

## Decomposition

## Structure

## Algorithms

## Usability Features

## Data Structures

## Test Data

## Post development

# Implementation

# Evaluation

## Testing for function

## Testing for robustness

## Testing for usability

## Success Criteria

## Further development

## Maintenance

1. The program will be tested on Windows 10 so although the program will most likely work from Windows XP and up I cannot be sure without testing for each operating system [↑](#footnote-ref-1)
2. This is the smallest amount of total RAM out of my stakeholders [↑](#footnote-ref-2)
3. Source: <https://support.microsoft.com/en-us/windows/windows-10-system-requirements-6d4e9a79-66bf-7950-467c-795cf0386715> [↑](#footnote-ref-3)