Analysis

Problem identification

There already exists lots of educational games of different kinds that can give beginners in computer science a brief introduction to what is computer science, but there are very few such games for intermediate or more advanced learners. Despite the fact that most advanced learners are driven by their own motivation rather than the fun from playing games, it would still be useful for them to have some sorts of visualisations of some abstract concepts, as it is often the case that some of the algorithmic ideas in computing are rather abstract and can easily deter a new learner from further study.

Having realised that, my project is going to be writing an educational game using Python and its extension library Pygame. The game will try to illustrate some advanced computing algorithms that may or may not be taught in A-level Computer Science. The idea is that some algorithmic problems all have a very interesting story, behind which lies the complicated algorithms. So, I would like to illustrate these algorithms vividly for learners and hopefully this could encourage more people to learn these algorithms, which might seem quite daunting at first.

This problem is amenable to a computational approach due to two reasons. The first one is the nature of a game. As one of the aims of this project is to present computational ideas graphically and interactively, a computer would be necessary and sufficient to provide a platform for the user because it can output some game information through the monitor and receive input from the user to do the processing. Moreover, as the theme of this game is computing theory, it would make more sense if a player is playing this game on a computer. It could also make it easier for a user to implement these algorithms if he/she is sitting in front of a computer. Being able to access the source code of the game is also a good way to better understand the implementation. Compiling a book about these theories might be an alternative but would be less interactive and interesting to the learner.

Stakeholders

Despite every effort is made to ensure this game is as interesting as possible, this game is not designed for absolute beginners who want to learn more about what computer science is, the reason of which is that this game assumes basic knowledge in computing such as a tree in graph theory and it does not include introductions to these theories. If a user wants to learn more about the implementation of an algorithm, programming knowledge may be needed. However, it is still possible for a learner to get a taste of advanced algorithms, although this game is not designed to do so. In conclusion, there are two main possible types of stakeholders:

1. Those who have already obtained some basic concepts of algorithms and want to study further in this area, then this game would be a good starting point to get a feeling;
2. Teachers might want to utilise this game in lessons while introducing relevant topics in order to add more fun to lessons. As each level in my game will centre around a key algorithm, e.g. dynamic programming, therefore this game can be played when a certain topic will be discussed in a particular lesson.

Research

Coding game

Creating interesting game background to help leaners learn coding from scratch

Competitions similar to OJ but multi-player

Different types of games to improve coding skills

Lacks knowledge of some advanced algorithms

Fancy visual effects are used as an illustration of some rather abstract ideas in programming which makes this platform quite fascinating for new coders who want to make a start. However, most of the content is for beginners who rather than experienced programmers. They also offer multi-player competitions where one can challenge friends, schoolmates or coworkers, which is fun and engaging, but there is less visualisation in some of the higher level problems. In addition, those coding challenges are less difficult than those on lots of OJ platforms.

<https://www.codingame.com/start>



[CS-Playground-React](http://cs-playground-react.surge.sh/)

A simple in-browser JavaScript sandbox for learning and practicing algorithms and data structures, lots of classic sorting algorithms as well as data structures that are frequently used are included in this platform. Some challenges are provided at the end to test one’s knowledge and make it more fun. In addition, it also offers solutions when one gets stuck, and comes chock-full of links to helpful articles, tutorials, and other resources but that is provided as an external link which makes it less interactive. In conclusion, this website does teach you some very important data structures and algorithms but not in a game context.

<http://cs-playground-react.surge.sh/>



Teaching Kids Computer Programming

There are lots of introductory games for kids as a taste of programming, for example, the Scratch programming language. And this website also offers 6 free games for teaching kinds programming. Although these games are very interesting and easy to play, they are designed to be a very brief introduction to programming which is not what I intended to do.

<https://educators.brainpop.com/2014/09/26/6-free-games-teaching-computer-programming-kids/>



Online Judging system

<https://uva.onlinejudge.org/>

<https://leetcode.com/>

<http://codeforces.com/> 

These are all very well-known online judge systems where programmers can practice solving algorithmic problems and submit their solutions to judging systems. Feedback can normally be given in a few seconds. Some competitions may also be held regularly on these platforms. They are all perfect places to improve coding skills, but those problems are all designed by experts and tend to be very challenging. And it is often the case that there is no visualisation of problems available on these websites, which makes solving these problems even harder.

UK Bebras challenge

This challenge is held annually and introduces computational thinking to students in different age groups. The main part of this challenge is to solve puzzles that require logical thinking rather than prior knowledge in computer science. Web-based human interaction is also available so that the participant can interact with the computer to obtain the solution.

<http://www.bebras.uk/students.html>



Ticket to Ride – a board game

*Ticket to Ride* is a cross-country train adventure where players collect cards of various types of train cars that enable them to claim railway routes connecting cities in various countries around the world.

Students through the missions that they choose about connecting one city to another, come across the implementation of some path finding and minimum spanning tree algorithms, such as Kruskal’s, Prim’s and Dijkstra’s algorithm.

<https://www.daysofwonder.com/tickettoride/en/>



An Educational Game for Teaching Search Algorithms

This is an article about using a Pacman game designed by the university team to teach searching algorithms such as DFS, A\*, etc. This game consists of detailed explanations and visualisations of these algorithms in a game context. Students who play this game will also have the chance to apply their knowledge in this game in order to solve some of the challenges in this game.



Brief conclusion:

Having done the research, I realised that there are some similar platforms and projects that are committing to the visualisation of algorithmic problems, though with various approaches and levels. For instance, coding game has fascinating game context for new coders and makes it much more appealing to beginners, while online judging systems provides much more challenging and well-designed problems and feedback on users’ solutions after they submit their codes. If we go back to my initial thought, I intended to combine interesting stories and abstract computational ideas together to make the learning process less painful. This idea is similar to the game mentioned in the article An Educational Game for Teaching Search Algorithms. The game comes with explanations of great details and vivid illustration. Human interaction is achieved by asking students to apply what they learned in another mode of the game. This is the closest solution to my problem and thought, but there is still something to add.

Essential features

In the project mentioned above, only searching algorithms are included and they are all demonstrated in the same game. (which does have some advantages) I want to add more algorithms of different categories and different games might be used to introduce different algorithms. It will consist of three features:

1. The game is a combination of OJ and the game “Pacman for teaching search algorithms”, two of the existing solutions to my problem. Therefore, most of my games will be based on problems from online judging systems (might be simplified or decomposed). In other words, OJ problems will be visualised in my game. This is due to the fact that lots of problems from OJ are described as games or puzzles themselves, which already adds fun to my game and makes it easier to adapt them. In addition, qualities of problems from OJ are generally very good and I will also try my best to select the most classic ones to ensure leaners can have a good learning experience.
2. There are a range of games included in my project and most of them will have different levels of difficulty. Brute force may be able to address some of the easiest levels but will become too inefficient to be used while coping with harder levels. This design is based on the idea that efficiency is usually the key part of an algorithm design when the data scale becomes very large, which is often the case in the real world.
3. Last but not least, this game will be able to run offline. Bizarrely, nearly all existing solutions require users to have a stable Internet connection. Although publishing a game online will make it accessible to more people, being able to run offline gives users the opportunity to access it anywhere and anytime they like. And the game could run more smoothly on a local computer.

Limitations:

There are two main limitations of my solution. The first one is that the formal theories behind each game is not included in this game and users will have to find out themselves how exactly a problem is solved by a computational method and how this method is developed step by step. The reason why theories are not included is that there are already lots of excellent books or websites available, designed by expert computer scientists, and they tend to be very good resources for systematic learning. The main focus of this game is to provide visualisation which can assist learners in their learning process and it is not meant to be a replacement of further study on those theories. This game is therefore not suitable to be used as a full learning guide.

The second limitation is that the visual effect of this game will not be as good as that on some of other platforms which appeared in the research. Although effort will be made to demonstrate the functionality of algorithms as much as possible, fancy visual effect is not the main concentration of this game. Some of the games may have already done the abstraction and may be presented to the user in a different way from how its original description is. This could potentially make the game less interesting to play for some of the users.

Requirements

Hardware

A computer that is able to run this game would be sufficient. As this game does not take many CPU resources, nearly all computers nowadays will be able to run this game.

Software

Windows, macOS or Linux installed

These three operating systems support Python which is what this game is written, therefore it would be essential for a device to have one of these operating systems installed

Python interpreter

This is used to run the source code of this game.

Pygame library

This library provides lots of functionality that enable a programmer to write a game. My game will be mainly based on this library.

Success Criteria

|  |  |
| --- | --- |
| Criterion | Comments |
| Design | |
| Clear instruction on how to play the game | So that a user can easily start to crack on the game |
| Problem illustration as straightforward as possible | Many of games in my project will be a simplified version of some rather complicated problems, so it is important to give clear illustration |
| Having seen a similar problem is helpful but not essential to be able to play | A game level might not contain detailed explanation or tutorials such as examples on how to operate, but operations will be designed so that most users can figure them out intuitively |
| Extra guidance provided if a user does not understand the problem | If a user is having difficulty on how to play the game or understanding the problem, there will be extra help available |
|  |  |
| User experience | |
| Game can run smoothly and provides good user experience | Smooth running can give the user the willingness to keep playing |
| Program does not contain serious bugs | So that the game will not crush unexpectedly in most situations |
| Essential functions are implemented and available to the user | e.g. the user does not need to start the game again if he/she took a wrong step as undo button is provided |
|  |  |
| Educational aspect | |
| The user needs to think strategically in order to pass the game | As the primary purpose of this game to help user learn knowledge |
| The game is as interactive as possible | So that user will not give up the game easily if he/she is stuck |
| This game can encourage people to learn more about these algorithms rather than deter them | Another main focus of this project is to make the learning process of some complex algorithms more enjoyable, therefore the game is designed to motivate learners |
| Solutions are straightforward for players to follow | Although explanations on how a solution is achieved will not be included, the correct answer will be displayed and made easy for a user to follow |
| Principles behind games are logically strict | Since this game has educational purposes for advanced intermediate and advanced learners, it is vital to ensure the principles are accurate and logical |
| Stories are interesting to most players | Again, this can make a user enjoy more about the learning process |
|  |  |
| Code itself | |
| Idea of OO is used properly throughout | There are lots of ideas in object-orientated programming, e.g. class, inheritance, private, etc. These ideas should be used accurately and correctly in the code. |
| Code is as elegant as possible | So that is would be easier for others and to read |
| Code is nicely commented | To make review process easier and more efficient |
|  |  |

//Visualizations can demonstrate the operational functionality of algorithms and are designed in line with the principles of student’s active learning