Design

Overall system design

The proposed system will be constructed in VB.Net 2010 using a Windows form template; therefore it will be a Windows form application. The system will be constructed on Windows XP computer systems at both Yeovil College and my home. Once my system has been constructed it will be saved onto the Moodle Decision Mathematics 1 course, so that this system can be accessed through Moodle at home, college or anywhere where there is an internet connection.

The proposed system is to be interactive, informative; it should enable students to study D1 on their own and it should teach students how to perform the three algorithms to find either the shortest route or minimum connector of a network. This is what my end user wanted the proposed system to do and I believe that along with what I had originally decided to do and the changes that I have made to it during my design it will allow me to fulfil my end users targets as well as my own.

Also my program will be easy to navigate because it will be very clear and concise; it will also tell the reader what certain options and parts of my program do, thereby making it easy for college students and teachers to understand. Also the lack of green and red in my program will mean that those who suffer from red-green colour deficiency will not be affected. However I have chosen the colour red to display the incorrect statement so that users know when they have clicked on a wrong arc, vertex or table box because I believe this colour is the most appropriate in this case.

I believe that even though the look of my program is important it is not as important as the content of my program or whether it is user friendly. Therefore I will be focusing more on the content of my program and that everything does as it should and that it is user friendly rather than it being stylish. However I will try and make my program look similar to resent Windows applications and make it relatively interesting so that people are not bored by it being to dreary in terms of colour and look.

Description of modular structure of system

Main Menu

Kruskal’s algorithm

Prim’s algorithm network form

Prim’s algorithm tabular form

Dijkstra’s algorithm

Teacher’s Section

Quit

Teacher’s login page

Run

Times achieved

Timed Run

Run

Times achieved

Timed Run

Run

Times achieved

Timed Run

Run

Times achieved

Timed Run

Teacher’s Menu

Forename entry page

Quit to Main Menu

Dijkstra’s algorithm

Prim’s algorithm tabular form

Prim’s algorithm network form

Kruskal’s algorithm

Definition of data requirements (Design data dictionary – from the viewpoint of programmer)

* Description of record structure:

Long term data: The times achieved for timed runs of algorithms, forenames of the students who achieve them and the generated rank number will be stored in binary random files. An example of how data will be stored in the binary random file is like this:

1

John

00:00:00

Where the rank is stored as an Integer, the name is stored as String and the time achieved is stored as DateTime.

This example will then be displayed in the times achieved section for each timed run of an algorithm and the teacher’s section like this:

Rank Name Time

1 John 00:00:00

However the forename will be set to a maximum of 50 characters for each one saved, so that users don’t have too many problems with errors when they enter their forename; meaning that there is less for them to remember. This is due to the fact that the width and length of binary random files must be set for data to be stored in them.

The password for the teacher’s section will be stored in the coding and as long as the password entered in the textbox matches this then the user will be allowed access to the teacher’s section; to make changes to the data stored.

Short term data:

I will store the arc weights of the network in a 2D array, the weights will be decided by a random number generator. This is only to be done while the program is debugging and will therefore not be stored for the long term. This will make it easier to construct the table for Prim’s algorithm tabular form and it will also make it easier for my program to construct the network. Also all of the vertices of the network will be assigned a letter from A-G where only a maximum of one direct route will exist between each vertex. However there will be no loops in my network, therefore e.g.: A won’t have a direct connection to A. Therefore the array will be an 8x8 array because there are 8 destinations and only a maximum of one route between each vertex. Also the workings produced from the run or timed run of an algorithm will not be kept for the long term as they are not needed, however they will need to be stored during the run of an algorithm so that then the program can do the next stage.

* Validation required:

Long term data: When the student enters their forename they will have to enter one as string which contains no spaces. If they enter a forename that is not as string, contains spaces, contains no characters or contains more than 50 characters then the program will flag an error message (once the user has pressed the “OK” button); that requests the user to enter the forename again. It will also tell them what they have done wrong so that they know what to change for it to be accepted. This will be done by using a message box which will appear and say “This is an invalid forename because it either contains an invalid character such as a space or a number, it contains no characters or it contains over 50 characters. Can you please enter your forename again?” Also in the message box it will display a button that says “OK” and the user must press this to continue to enter their forename as a forename must be entered. Once this forename has been entered, the user has pressed “OK” and the forename has been accepted a different message box will appear which says “Your forename and time has been saved to the leader board” where the user will have to press an “OK” button on the message box to go back to the main menu. The time where the timer was stopped will be shown on the screen to the user throughout this process. However when the program produces an error message because the name entered is too long it will only happen if the name is ridiculously long because the limit has been made high to prevent this.

When the timer reaches 59:59:99 the program will stop and ask the user for their forename and after this has been accepted it will save a time of 59:59:99 as their time achieved. This is because in the exam students will not be allowed to take more than 30 minutes on these types of questions, however I have made it so that the student has one hour because they need to learn how to perform the algorithms and therefore may need more time than that is allowed to do this.

There is no limit to the number denoting the rank (which is generated by the bubble sort), however my program must be able to cope with at least 100 data items and store them in order.

For the password that is entered to be valid it must match the password given in the coding, however when the password is entered into the textbox by the user it will appear as black dots so that it can’t be seen by others. Once they have entered the password into the textbox and they have pressed the “Sign in” button the user will gain access to the teacher’s page; if they entered the correct password. If the user enters an incorrect password a message box will appear and it will say “The password you entered was incorrect. Please enter a different password to gain access to the teacher page.” Also the user will have to press the “OK” button to continue entering the password where upon doing this the contents of the textbox will be deleted; so that they can enter another password.

Short term data:

The arc weights must be in the range of values of 0<arc weights<=20 and they must be integers. However this will be implemented using my program so that the random numbers are multiplied by a constant so that what ever comes out is a number in this interval. Also the arc weight will denote the distance between a vertex and another vertex. Also the same arc weight can be used to denote more than one arc. The network will be set meaning that the same vertices will have the same letters for each run or timed run, however the arcs connecting them will have different weights.

* File organisation and processing:

Long term data: The files containing the saved forenames, times and ranks for each timed run of an algorithm will be split into four different binary random files, so that they can be easily updated and easily coded for. One binary random file will contain the times achieved for timed runs of Kruskal’s algorithm, another for Prim’s algorithm network form, another for Prim’s algorithm tabular form and the last for Dijkstra’s algorithm. These will be opened in list boxes in the teacher’s page of my program, therefore only teachers will be allowed to update the data stored in each file.

Also when the user chooses the times achieved option on the main menu for an algorithm the program will access the corresponding file and display the contents in a list box which the user will be able to scroll up or down, or left or right, so that they can see all of the times achieved. This will also be done on the teacher’s page, however in the teacher’s page teachers will be able to update and delete the data in the list boxes and therefore in the files themselves; once they have selected one of the options on the teacher’s menu.

The teacher’s section password will be included in the code so that it is just a case of writing code that accepts the password if the text entered in the textbox equals that in the code.

Short term data:

The arc weights and vertex letters will be contained in a 2D array during debugging and then the arc weights will be deleted once finished. The program will read this array (which has the arc weights contained) line by line.

Identification of appropriate storage media

The program will be stored on my home computer and Yeovil College’s computer system where adaptations will be made during construction; therefore it will be saved into a folder onto a computer’s hard drive. After construction the master program will be stored on the Decision Mathematics 1 course page on Moodle; where any further adaptations will be made and saved. However it may be adapted by Yeovil College’s Maths teachers and lectures on their home computer or a college computer and will then be uploaded onto Moodle again.

The reason for choosing Moodle as my preferred storage media is that it already has a login system, meaning that I can concentrate on coding the algorithms instead of making and coding a login page. Also my program will be stored in the Decision Mathematics 1 section of Yeovil College’s Moodle meaning that only those doing Maths and the D1 module will be using it, therefore I don’t have to worry about whether my program is being used by those who shouldn’t be using it. Also on Moodle my program can only be accessed by those who are enrolled at Yeovil College meaning that those from other colleges can’t use my resource and it allows students to access my program from home or at college.

After my program has been completed it may be adapted so that it can be saved and used on memory sticks and CD’s. If this happens then the master program will have to be recoded so that it will be updated from what has been done by the student at home.

Identification of processes and suitable algorithms for data transformation

The algorithms that my program will be coded for are that of Kruskal’s, Prim’s network and tabular form (minimum connector) and Dijkstra’s algorithm (shortest route). This is to be used and run on a network which contains vertices and arcs, so that it will teach students how to find the minimum connector or shortest route of a network. These algorithms are shown below in a step by step form:

Kruskal’s algorithm:

1. Select the shortest edge in a network.
2. Select the next shortest edge in the network.
3. Select the next shortest edge which doesn’t create a cycle.
4. Repeat step 3 until all the vertices have been connected.

Kruskal’s algorithm has cubic complexity.

Prim’s algorithm network form:

1. Select any vertex.
2. Select the shortest edge directly connected to that vertex.
3. From the vertices that are not already connected select the shortest edge from one of these to a vertex that is connected. This mustn’t create a cycle.
4. Repeat step 3 until all of the vertices have been connected.

Prim’s algorithm tabular form:

1. Choose a column, mark it and cross out its row (The starting column that is chosen is usually given in the question, however if it isn’t then any column can be chosen instead.). Delete the corresponding column’s row.
2. Choose the smallest number in the column and circle it. If there is a choice then choose either.
3. For the number that you have just circled, cross out its row and put a mark above its column at the top of the table.
4. Choose the smallest number not already crossed out from the marked columns and circle it.
5. Repeat steps 3 and 4 until all of the vertices have been included in the tree.

Prim’s algorithm network form and tabular form have cubic complexity.

Dijkstra’s algorithm:

1. Label the start vertex with permanent label 0 and order label 1.
2. Assign temporary labels to all the vertices that can be reached directly from the start vertex.
3. Select the vertex with the smallest temporary label and make its label permanent. Then add the correct order label.
4. Put temporary labels on each vertex that can be reached directly from the vertex you have just made permanent. The temporary label must be equal to the sum of the new permanent label and the direct weight from it. If there is an existing temporary label at a vertex, it should be replaced only if the new sum is smaller.
5. Select the vertex with the smallest temporary label and make its label permanent. Add the correct order label.
6. Repeat until the finishing vertex has a permanent label.
7. To find the shortest paths, trace back from the end vertex to the start vertex. Write the route from start to finish and state the weight.

Dijkstra’s algorithm has quadratic complexity.

Also the bubble sort will be needed to sort the times achieved and forenames of students into order from lowest to highest and then assign each student a rank which depends on the time that they have achieved and the times others have achieved. This is so that the students know where they are in comparison with others and so that the teacher can see how fast someone takes to complete a timed run of an algorithm. I am going to use the bubble sort to sort the data items into order from lowest to highest because I have already coded it before in class, meaning it will take less time for me to implement. Also because my program will have to store at least 100 of these data items then the fact that the bubble sort is slower than the other sorts with a large amount of data items then it won’t make that much difference with such a low amount as 100. The bubble sort will sort the data into order and the results will be saved in the correct binary random file and will then be overwritten and performed again when the user saves more of their times.

The bubble sort in step by step form and pseudo-code form are shown below:

1. On the first pass, the first number in the list is compared with the second, and whichever is the smaller assumes the first position. The second is then compared with the third and the smaller is placed in the second position, and so on through the list. At the end of the first pass the largest number must be at the bottom of the list.
2. For the second pass, the process is repeated, but excluding the last number, and on the third pass the last two numbers are excluded.
3. Passes continue in this way until a pass with no swaps occurs. The list is then sorted.

procedure bubbleSort( A : list of sort able items )

repeat

swapped = false

for i = 1 to length(A) - 1 inclusive do:

if A[i-1] > A[i] then

swap( A[i-1], A[i] )

swapped = true

end if

end for

until not swapped

end procedure

The weights of arcs between two vertices will be decided using a random number generator where the results of this will be stored in a 2D array, so that students get a differently weighted network every time. This will only be the case in timed runs of algorithms because the arc weights for a run of an algorithm will be stored in a 2D array but the values will be set so that the network has the same arc weights every time.

User interface design (HCI) rationale including: sample of planned data capture and entry designs and sample of planned valid output designs

Sample of planned data capture and entry designs:

Some of the data that needs to be obtained from the user is that of their forename after completing a timed run of an algorithm, the time that is achieved by them (which is obtained from the timer), their rank in the leader board (which is obtained from the results of the bubble sort on the data) and the password which gains them access to the teacher’s section. Also the user will have to click on the correct arcs, vertices and table boxes to complete a timed run of an algorithm.

The forename will be obtained from the user by sending them to another form immediately after completing a timed run of an algorithm. The user will be sent to this same form every time they complete a timed run of any algorithm, so that they can input their forename. In this form (Forename entry page) the user has to enter their forename in a textbox and then press the “OK” button from here they are sent to the main menu if the forename is accepted. The time achieved by the user is obtained from the timer and is then saved along with the forename into the correct binary random file. This is then used to generate a rank from the bubble sort where the bubble sort puts the data that is saved into the binary random files into order from the lowest time achieved to the highest time achieved and then assigns each person a number based on their time achieved to complete the timed run of the algorithm. Therefore if a student achieves a low time then they will be given a lower rank in the leader board and if they achieve a high time then they will be given a higher rank in the leader board.

When the user presses the “Teacher’s section” button the program displays the teacher login page where the user has to enter the password (that is in the coding) into a textbox and then presses the “OK” button where upon it being accepted the user gains access to the Teacher’s Menu. Where the teacher has to press one of the five buttons which say “Kruskal’s algorithm”, “Prim’s algorithm network form”, “Prim’s algorithm tabular form”, “Dijkstra’s algorithm” and “Quit to Main Menu”. The algorithm pages display the contents of each of the four binary random files in list boxes; so that they can be adapted and updated by the teacher. Also they will contain a “Save changes” and a “Quit to Teacher’s Menu” button so that the teacher can save the changes they have made to the files themselves and go back to the Teacher’s Menu easily.

The user won’t have to enter the shortest route or minimum connector of the network into a textbox because it may be difficult to make it interactive and to integrate it into my program. Examples of a minimum connector and a shortest route are shown below:

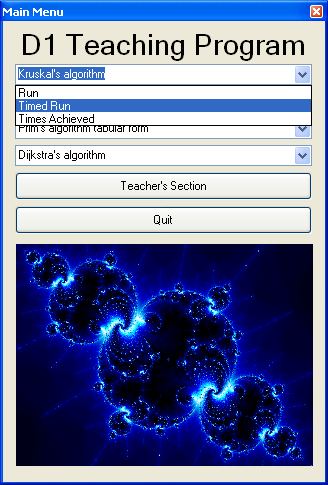
Minimum connector: AB, BE, EF, FG, FC, ED

The shortest route from A-G is: A, C, F, G

Also the user will not have to enter the weight of the shortest route or minimum connector into the program because the weight can be easily worked out from the arcs highlighted in blue. Also for the algorithm to move onto the next stage the user must select the next arc, vertex or table box depending on the timed run of an algorithm chosen. If the correct arc, vertex or table box is chosen the algorithm moves onto the next stage and the arc, vertex or table box turns blue to indicate that it has already been chosen. In timed runs of Kruskal’s algorithm the user must select arcs until the algorithm has found the minimum connector. In timed runs of Prim’s algorithm network form the user must again select arcs until the algorithm has found the minimum connector. In timed runs of Prim’s algorithm tabular form the program will cross out the row of the chosen table box (if it is correct) or the starting vertex’s row at the beginning; which is given in the question. Then it will mark the chosen table boxes’ column at the top of the table with a number. Then the user will select the table box that contains the lowest weight in one of the columns marked where upon doing so the program will circle the weight number inside the table box blue. This will carry on until the algorithm has found the minimum connector. For timed runs of Dijkstra’s algorithm the user will have to select vertices until all the vertices labels have been made permanent, then they will have to select the arcs that make up the shortest route. While the user has been doing this the program is inserting numbers into workings boxes which are next to the vertices; the numbers entered depend on what vertices are to be made permanent to construct the shortest route. Once the user has selected the correct arcs that make up the shortest route the user will be taken to the “Forename entry page”.

Once the user has finished any of these timed runs of algorithms the program will say “You have completed the timed run of the algorithm and you have found the shortest route or minimum connector. Press the Enter key to go to the forename entry page”, this is where the timer will stop and the user will have to press the “Enter” key to go to the “Forename entry page”. This is to give the user a chance to see what has been going on and allow them to understand what they have done without the pressure of the timer. For a run of an algorithm the program will say “You have completed a run of the algorithm and you have found the shortest route or minimum connector. Press the Enter key to go to the Main Menu.” This is to make sure that the user knows that they have completed a run of an algorithm before just suddenly going back to the main menu.

The user also has to choose an option from the main menu, which takes them to other forms these range from buttons that take teachers to the Teacher’s section and another that quits the program. Also they can choose whether to do a run, timed run or see the times achieved by students in timed runs for each algorithm; this is done by pressing the drop down button on the corresponding algorithm’s combo box and then selecting one of the options mentioned. This is shown below:



This is the combo box for Kruskal’s algorithm. After the user has picked an option from the combo box and presses the “Quit to Main Menu” button on the form shown the combo box will still have “Kruskal’s algorithm” in it on the Main Menu instead of the option that was chosen previously. This is the same as with the other combo boxes.

Sample of planned valid output designs:

Below are two designs that I have made for the main menu to help the user navigate through my system, where the first is my original and the second is the new and improved one where I am going to state why I am going to use the second instead of the first:

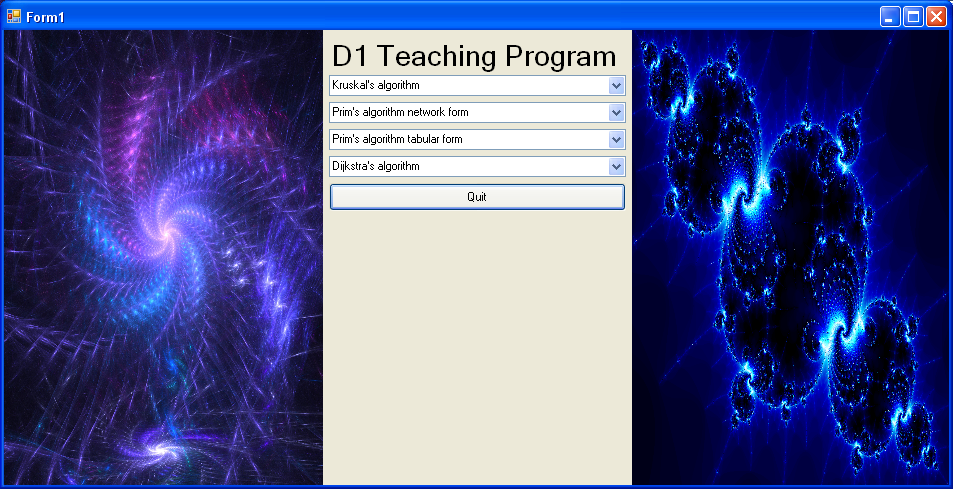
This is a clear and concise menu that clearly shows where the user needs to go to find what they want without looking at the manual.

The pictures look distorted as they have been stretched meaning that they don’t as good as they should. Also the picture on the left doesn’t look like a fractal as much as the one on the right.

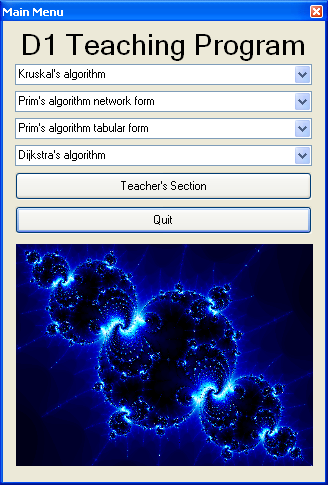
There is lots of wasted space meaning it doesn’t look very professional.

This form has the FormBorderStyle setting set to sizeable, meaning the size of the form can be changed by the user which distorts the imaging and objects contained within the form. Also it has a heading of Form1 which won’t mean anything to users; this may make it difficult for the user to navigate to it.

There is no border which makes the picture difficult to see and makes it look less professional.



This form looks more professional and it tells the user that this is the Main Menu, meaning users will know where the button “Quit to Main Menu” on other forms will take them. Also now that the FormBorderStyle setting is set to FixedToolWindow the size of the form can’t be changed by the user. This makes it look more professional.



This is a clear and concise menu which is easy to read and understand.

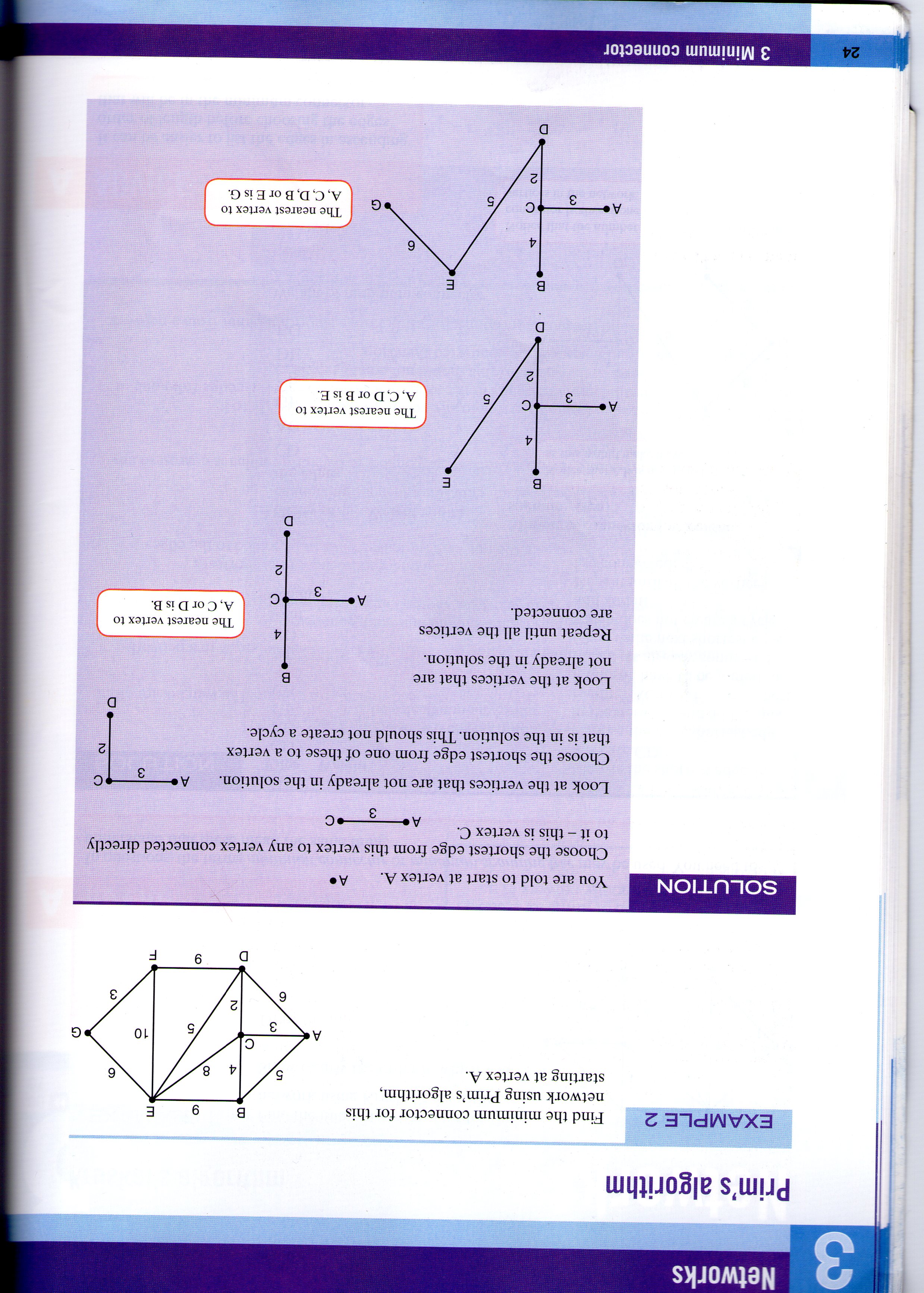
However I thought that it would be good to add a teacher’s section so that teachers could update and delete data that was not needed anymore. This will prevent the files becoming corrupted or any of the system changing from the original.

This is an exit button that closes the program.

The picture looks a lot more professional as the format has been changed. This makes the picture look clearer and better.

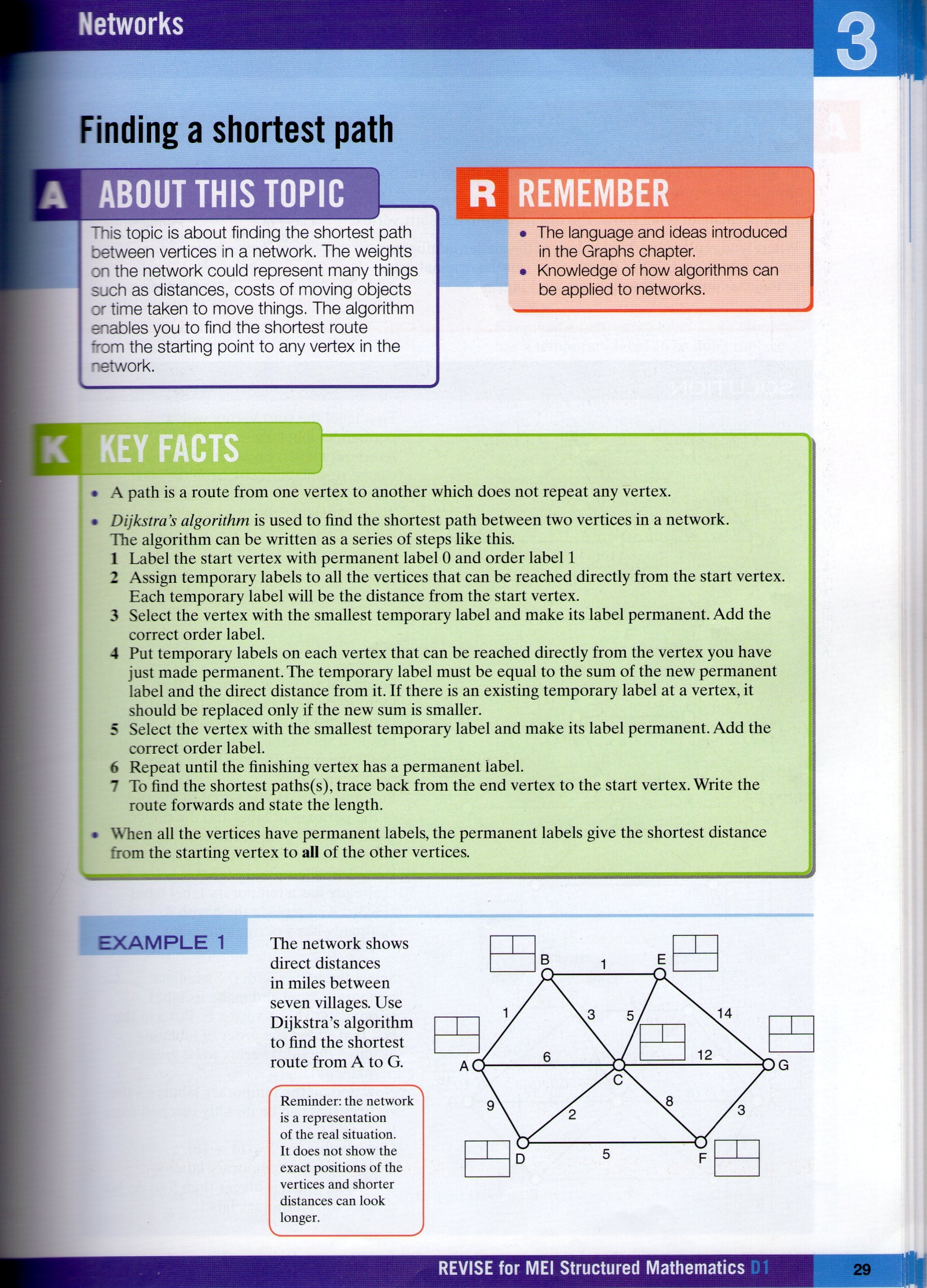
There is no wasted space; however there is some unused space for borders but this makes it look more professional.

The networks or tables for each timed run and run of each algorithm are generated using an 8x8 2D array where the network or table is drawn onto the screen for the student to interact with for timed runs or just to look and watch the algorithm run on them for runs. An example of a network that will be displayed for each run and timed run of an algorithm for Kruskal’s and Prim’s algorithm network form is shown below:



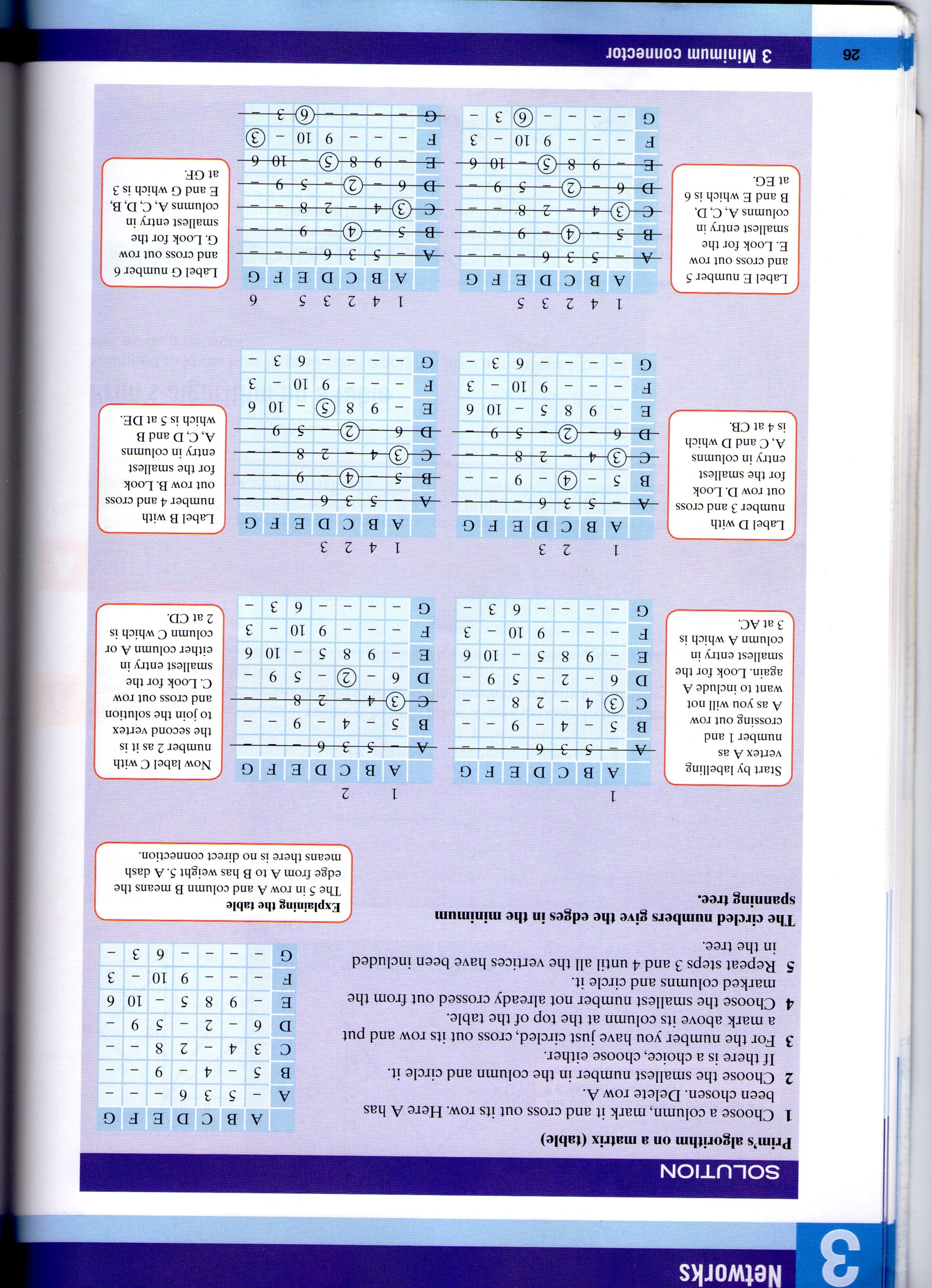
This network has vertices that range from A-G and the network is clear and concise and easy to read. It is also in an appropriate colour that can be easily recognised by the majority of students.

The same process is used to produce the network below for Dijkstra’s algorithm except that other tables need to be added to work out the shortest route between two vertices, an example of which is shown below:



This network contains other tables which are necessary to perform Dijkstra’s algorithm, however in runs and timed runs users will not have to type in any of the temporary or permanent labels because the program will do it instead. This is because it may be difficult to code and it could cause boredom having to type them in. This is because I don’t think that it would enable students to use it in bite sized chunks or be interactive.

Below is an example of a table that will be drawn and then displayed to screen when someone performs a run or timed run of Prim’s algorithm tabular form:



A table similar to this will be produced by my program using an 8x8 2D array for runs and timed runs of Prim’s algorithm tabular form.

All of the forms in my program will contain a “Quit to Main Menu” button which takes the user back to the main menu where they started from where ever they are in my program, so that if the user wants to do something else or take a break then they can do so quickly, thereby creating a shortcut. However the main menu will contain a “Quit” button instead which exits the user from the program. Also the forms which display the times achieved in each algorithm for the teacher’s to update or change will contain a different button this being a “Quit to Teacher’s Menu” button, which closes the current form and instead displays the teacher’s menu.

All of the runs and timed runs of algorithms will contain and display a help button which will be the colour purple and will produce a drop down box that contains the stages of the algorithm to help the student. Also when the student chooses a wrong arc, vertex or table box a hint appears underneath the statement “The arc, vertex or table box you chose is incorrect. Please choose another and use the hint below for help” (in red), this hint will be based on the algorithm itself and will hopefully help the user figure out what they have done wrong; without wasting time by clicking the help button which is a “?” inside a circle. When the user has to click on arcs to find the shortest route at the end of a timed run of Dijkstra’s algorithm, the program will instead of saying the incorrect statement at the bottom of the screen it will put a tick or a cross next to the arcs that they have chosen; depending on whether they are in the shortest route or not. This is so that the user knows which arcs they have chosen are in the shortest route of the network and which ones are not.

When users enter an incorrect forename the program will display an error message which helps the user to correct what they have done wrong, so that they can enter a forename that is accepted. Also when the forename is accepted the program will display an error message that tells the user what has happened to the data that they have entered. The data is saved to the leader board for timed runs of the corresponding algorithm, where the user can see how they compare to other users times in the times achieved section of the corresponding algorithm.

When users enter an incorrect password to gain access to the teacher’s section of the program an error message will appear which tells them why the password that they entered was rejected. However an acceptance message that tells the user that they have gained access to the teacher’s section will not be included, because I believe that it is not needed because the user will already know that they have entered the correct password because the program will display the teacher’s menu.

When a run or timed run of an algorithm page is shown the question is displayed to the user at the start at the top of the page along with the network or table but not the arc weights because they will be shown once the user presses the “Start” button, so that they don’t cheat. When the user clicks on the correct arc or vertex it will turn blue or if they click on the correct table box then the weight inside it will be circled blue and if the user gets it wrong then it will remain as it is until the user selects it at the right time.

In runs and timed runs of algorithms the minimum connector or shortest route and weight will not be shown because the user should be able to work these out for themselves from the arcs, table boxes and vertices they have chosen and other workings. However it may be added to my program if I feel that it is necessary to do so to make my program more informative. Also during runs of algorithms the program will run the algorithm slower along with the workings out, so that students can learn how to perform the algorithm properly. During runs the program will tell the user what they will need to do for timed runs, what they will need to do in the exam and what the algorithm is doing at certain stages; this will be based on the step by step method of the algorithm. It will also tell them that they need to press the “Enter” key on the keyboard to move onto the next stage of the algorithm and that they need to press the “Backspace” key on the keyboard to go back a stage of the algorithm. In timed runs the program will tell the user what they need to do to move onto the next stage of the algorithm which will say for Kruskal’s algorithm “Click on the next arc you think is in the minimum connector.” For Prim’s algorithm network form it will say the same thing and for Prim’s algorithm tabular form it will say “Click on the next table box that contains the arc weight that you think is in the minimum connector”. Finally for Dijkstra’s algorithm it will say “Click on the next vertex you think should have its label made permanent” and then when the user has to find the shortest route by selecting arcs it will say “Click on the next arc you think is in the shortest route”. All of these will be displayed on their respective pages to the user above the picture of the network (once they have pressed the “Start” button). This is so that the user knows what they are doing and what they need to do to complete a run or timed run of an algorithm; without looking at the manual. Also the timer will be displayed all the way through timed runs of algorithms in the colour black, so that the user knows how long they have taken so that they can stick to a time constraint and so that those with red-green colour deficiency don’t suffer.

In the times achieved pages and the teacher’s section the times achieved at timed runs of algorithms by users will be displayed along with their forename and rank in a leader board; which will be output to the user in list boxes. However in the teacher’s section the leader boards will be able to be changed by teachers but the leader boards in the times achieved pages will not be able to be changed by any users. Also in the algorithm pages in the teacher’s section the leader boards will be accompanied by a “Save” button which saves the changes made by the teacher to the leader boards in the files.

Description of measures planned for security and integrity of data and description of measures planned for system security

The proposed system will not be coded for any extra security as such because I believe that if it can only be accessed through Moodle then it could only be accessed by those who are enrolled in the D1 course at Yeovil College. Therefore it will make it difficult for someone that is not using it for educational purposes to take or use my program.

The students will be able to access the program through Moodle by entering their own username and password. After this has been accepted an encryption key is needed to be enrolled in the Decision Mathematics 1 course. When the student is in this course the program will be able to be opened on the computer and be saved onto so that the changes to the times and leader boards are made permanent after the user has done a timed run of an algorithm. This is so that the updated leader board is saved to the program so that other students that go into it will see them updated.

Once the student has finished a timed run of an algorithm they will have to enter their forename where they can enter any forename they want, however it must be able to be stored as string and it must not contain spaces, 0 characters or more than 50 characters. This means that they can’t enter their surname as well; meaning that if someone does get hold of the program then there is no chance of identity theft or any breaches of the data protection act.

Also the data and files on Moodle are backed up every 24 hours meaning that it will be easier to keep the program safe and protect the data stored in it on Moodle. Also the data about the students contained in the program will only be stored for the duration the student remains at Yeovil College; usually being a minimum of 2 years. This will be enforced by a Maths teacher at Yeovil College (more likely my end user); they will be able to do this via the teacher’s section where the changes will be saved to the corresponding file. Therefore it will comply with the data protection act which states that “personal data processed for any purpose shall not be kept for longer than is necessary for that purpose”.

Also the times achieved, forenames and ranks will be stored in four separate binary random files so that the data will be more secure and students will not be able to update or delete data. This will be achieved by adding a teacher’s section to my program where a password is required by the teacher. I will make it so that only those who enter the exact password which is in the coding will be able to go further. This password will be chosen by teachers so that only teachers will know what the password is, also when the teacher enters the password into the text box the password will appear as black dots so that other people can’t see the password being entered.

Overall test strategy in relation to the problem being solved and tested

My method of development is top-down, however I will be using all of the testing methods because some are better at testing certain parts of my program than others. This is because I have started developing and making the main menu of my program first and then the other forms and modules that lead on from this. Also black box testing will be used to test the user interface, so that there aren’t many problems for those who are trying to navigate through the system.

|  |  |
| --- | --- |
| Test Series | Purpose of test series |
| 1 | Test the flow of control: Does the user interface work as it should? Does it take users to the correct form or option? (top-down testing) |
| 2 | Validation of input data performed correctly(bottom-up testing) |
| 3 | Iterations, decisions, calculations, searches and sorts performed correctly.(white box testing and desk checking) |
| 4 | Data is saved into the correct files.(system testing) |
| 5 | The system produces the required results as per the specification.(black box testing) |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test series and number | Purpose | Description | Test data | Expected result | Actual Result | Evidence in system testing appendix |
| 1.1 | To check to see if the menu choices in the main menu take the user to the desired form.(top- down testing) | The menu choices must take the user to where they want to go, so that it is easier for the user to navigate and understand. | Button pressed = Quit  Button pressed = Teacher’s Section  Kruskal’s algorithm combo box item chosen = Run  Kruskal’s algorithm combo box item chosen = Timed run  Kruskal’s algorithm combo box item chosen = Times achieved  Prim’s algorithm network form combo box item chosen = Run  Prim’s algorithm network form combo box item chosen = Timed run  Prim’s algorithm network form combo box item chosen = Times achieved  Prim’s algorithm tabular form combo box item chosen = Run  Prim’s algorithm tabular form combo box item chosen = Timed run  Prim’s algorithm tabular form combo box item chosen = Times achieved  Dijkstra’s algorithm combo box item chosen = Run  Dijkstra’s algorithm combo box item chosen = Timed run  Dijkstra’s algorithm combo box item chosen = Times achieved | The program should end.  The program should show the teacher’s login page so that one can gain access to the teacher’s section.  Displays the form containing the run page for Kruskal’s algorithm.  Displays the form containing the timed run page for Kruskal’s algorithm.  Displays the form containing the times achieved page for the corresponding algorithm.  Displays the form containing the run page for Prim’s algorithm network form.  Displays the form containing the timed run page for Prim’s algorithm network form.  Displays the form containing the times achieved page for Prim’s algorithm network form.  Displays the form containing the run page for Prim’s algorithm tabular form.  Displays the form containing the timed run page for Prim’s algorithm tabular form.  Displays the form containing the times achieved page for Prim’s algorithm tabular form.  Displays the form containing the run page for Dijkstra’s algorithm.  Displays the form containing the timed run page for Dijkstra’s algorithm.  Displays the form containing the times achieved page for  Dijkstra’s algorithm. |  |  |
| 1.2 | To check to see if the menu choices in the teacher’s menu take the user to the desired form.(top-down testing) | The menu choices must take the teacher to where they want to go, so that it is easier for the teacher to navigate and understand. | Button pressed = Kruskal’s algorithm  Button pressed = Prim’s algorithm network form  Button pressed = Prim’s algorithm tabular form  Button pressed = Dijkstra’s algorithm  Button pressed = Log out | The program should take the teacher to the page where the times achieved by students in timed runs of Kruskal’s algorithm are displayed and are able to be adapted.  The program should take the teacher to the page where the times achieved by students in timed runs of Prim’s algorithm network form are displayed and are able to be adapted.  The program should take the teacher to the page where the times achieved by students in timed runs of Prim’s algorithm tabular form are displayed and are able to be adapted.  The program should take the teacher to the page where the times achieved by students in timed runs of Dijkstra’s algorithm are displayed and are able to be adapted.  The program should display the main menu. |  |  |
| 2.1 | Validate the forename that is entered by the user. This is to see if an acceptance message or an error message needs to be displayed.(bottom-up testing) | The program should only not accept forenames if they can’t be stored as string, they contain no characters, have spaces or contain more than 50 characters. | John  Abigail  1  John Every  John.Every  Enters no characters. | Accepted  Accepted  Error  Error  Accepted  Error |  |  |
| 2.2 | To confirm the identity of the user being a teacher by means of validating a password entered by them.(bottom-up testing) | The program should only allow teachers to access the teacher’s section by means of entering a password and it matching the password contained in the code. The password is entered after the user has pressed the “OK” button. | Enters no characters  User enters incorrect password.  User enters correct password. | Error (error message will appear)  Error(error message will appear)  Accepted (proceeds to the teacher’s menu.) |  |  |
| 3.1 | To check if the arc weights produced by the program are correct.(white box testing) | The arc weights for each arc of the network are produced from random numbers and then multiplied by a constant. The arc weights must be integers and in the range of 0<arc weights<=20. | 5  1  0  -1  19  20  1.1 | Accepted  Accepted  Error  Error  Accepted  Accepted  Error |  |  |
| 4.1 | The data is saved into the correct binary random files.(system testing) | Once the acceptance message has appeared the student’s forenames and times achieved are sorted into order using the bubble sort, where a rank is generated for each student. The results are then saved into the corresponding algorithm’s file for timed runs. These need to be saved each time after some one completes a timed run of an algorithm. Where they need to be loaded and displayed in other forms. | Forename, time achieved and generated rank. | Looking in the corresponding times achieved section of an algorithm or the corresponding algorithm’s page in the teacher’s section. The forename is there the same as it was entered. Along with the time achieved and the correct generated rank. |  |  |
| 5.1 | An error message is displayed after entering an invalid forename in the textbox.(black-box testing) | The program should display an error message that states the possibilities for what they have done wrong and that they need to enter another name. | The result after entering a forename is error. | This should always produce an error message. |  |  |
| 5.2 | An acceptance message is displayed after entering a valid forename into the text box.(black-box testing) | The program should display an acceptance message that states that the user’s forename, time and rank have been saved to the leader board. | The result after entering a forename is accepted. | This should always produce an acceptance message. |  |  |
| 5.3 | Timer works correctly, so that it stops at 59:59:99.(black box testing) | The program should only time up until the timer reads 59:59:99 and it must not go beyond this. Once it has reached this time it must stop and ask the user for their forename.  Also if the user completes a timed run of an algorithm then the timer must stop and the program must ask the user to enter their forename. | Letting the timer run until 59:59:99.  The user completes the timed run of an algorithm between 15 and 16 minutes.  Letting the timer run until 20:00:00  Presses the “Quit to Main Menu” button at any time. | Stops and asks the user to enter their forename.  Stops and asks the user to enter their forename.  The timer will continue.  Stops, doesn’t ask for the user’s forename and returns the user to the main menu. |  |  |
| 5.4 | To check to see if the program recognises when the correct arc, vertex or table box is chosen.(black box testing) | When the user clicks on the correct arc, vertex or table box the algorithm moves onto the next stage, thereby allowing the user to complete the timed run. This is so that if the wrong one is chosen then an error message is displayed that tells the user that it was the wrong one. | Correct arc, vertex or table box is chosen by the user.  The incorrect arc, vertex or table box is chosen by the user. | Accepted  Error(the incorrect message and the hint is displayed) |  |  |

The changes to the proposed system since the Analysis section

* Instead of my main menu having 10 options which are assigned a button each, I will have a button for each algorithm and quit, therefore only five buttons will be displayed these being “Kruskal’s algorithm”, “Prim’s algorithm network form”, “Prim’s algorithm tabular form”, “Dijkstra’s algorithm” and “Quit”. This is to make the program look better and to make it simpler to navigate and understand.
* I have decided to change the buttons for the algorithms to combo boxes that have a drop down menu that will include options for them to select, these allow the user to choose to do a run of the algorithm, a timed run of the algorithm or see the times achieved by students to complete each algorithm. When the user has selected the option and it is displayed in the box it will display the corresponding choice’s form. This is to make the menu more interesting and to keep all the options for each algorithm together, thereby making it simpler. Also it lowers the number of forms that are needed, thereby decreasing processing time.
* Also the combo boxes will have the corresponding algorithm in them before the user chooses; to aid understanding. Also when the user chooses one of the options on the main menu and then clicks the “Quit to Main Menu” button the combo boxes will still display the algorithms names in their boxes and not the choice that they chose.
* I will also put images of fractals in my menu to make it look more interesting and inviting.
* I have decided to not allow my program to be saved onto memory sticks and CD’s as it could cause unwanted security issues and problems when uploading the scores someone has done at home to the master program. Also if it is on Moodle then it can already be accessed by students at any time, therefore saving the program onto a memory stick or CD is rendered pointless. Also I would have to rely on those who have done it at home to upload it to the master copy themselves which may cause problems as they may not know how to do it and they may not have enough time to do it.
* The bubble sort will be used to sort the student’s times from lowest to highest and then generate each student a rank depending on their time, so that the leader board can be created.
* I have decided to notify the user when their forename and time has been saved by displaying a message box that will appear and say “Your forename and time achieved has been saved to the leader board” where the user will have to press an “OK” button on the message box to go back to the main menu. This will be displayed once the forename has been entered, the user has pressed “OK” and the forename has been accepted.
* I have decided to instead save the times, forenames and ranks of students in 4 separate binary random files instead of text files. This means that the data will be more secure and I will be able to control who can update and change the data. I will do this by including a teacher’s section in my program which will be able to be accessed through the main menu and will require a password from the user. This will be decided by the end user or the other maths teachers, so that only teachers can update and remove data. If the user enters an incorrect password then a message box will appear that states that they have entered the wrong password and need to enter another.
* I have decided to change my program so that instead of a 20 second countdown before the user starts the question the user will have to press a button which says “Start”. This will make it easier for the student and will allow them to start when they want to.
* The forename will be set to a maximum of 50 characters for each one saved, so that users don’t have too many problems with errors when they enter their forename; meaning that there is less for them to remember.
* I have decided that when the correct vertex or arc is selected by the user it will turn blue to indicate to the user that it was correct and that it has already been chosen. Also when the user chooses a correct table box the arc weight inside it will be circled blue. Because it is a colour that is liked by a lot of people and it doesn’t affect users who have red-green colour deficiency.
* Once the user has completed a timed run or a run of an algorithm the program will say (for a timed run) “You have completed a timed run of the algorithm and you have found the shortest route or minimum connector. Press the Enter key to go to the forename entry page”, the timer will stop immediately at the same time. This is so that the user knows that they have completed the timed run correctly and that they now know what they need to do next. For a run the program will say “You have completed a run of the algorithm and you have found the shortest route or minimum connector. Press the Enter key to go to the Main Menu.”
* I have decided that the timer will be displayed in black font instead of red and then green, so that those with red-green colour deficiency don’t have the overall experience of my resource tarnished by it including these colours.
* I have decided that the user will not enter the numbers above columns or cross rows out because it may cause the program to be difficult to use. Instead the program will do this so that the user can do the other stages of the algorithm.
* I have decided that both the runs and timed runs will not display any extra workings that can be involved in performing each algorithm. However runs will include more information and more workings than timed runs but neither will display the minimum connector or shortest route along with the weight.