NEA Computer Science

5 Evaluation:

1. Comparison of project performance against objectives:

Objective 7.1.a - Allow the user to add a node through a node identifier within a given length of characters, with coordinates for its position on the axis on the screen.

This objective has been met, this allows the user to add a node at a valid position on the graph with a node ID of length 1 or 2, displaying the node information. This can show appropriate error messages if invalid node inputs are entered.

Objective 7.1.b - Allow the user to add edges with the mandatory inputs being the minimum and maximum capacities, the node it came from and the node it has gone too.

This objective has been met, this allows the user to enter an edge from 2 existing nodes on the graph, that are distinct, displaying the edge information under the node information in which the node has come from. This can show appropriate error messages if invalid edge inputs are entered.

Objective 7.1.c – Allow the user to add optional minimum capacities with a given weight as well as allow the ability to check for any given node the minimum ingoing/outgoing capacity is less than the outgoing/ingoing total respectively.

This objective has been met, this allows the user to enter minimum capacities as an optional input box, displaying this with the other edge information. This shows appropriate error messages if minimum capacity < maximum capacity of edge.

Objective 7.1.d – Allow the user to add node capacities for given node identifier and weight as an optional input.

This objective has been met, this allows the user to enter node capacities as an optional input box, displaying this with the associated node information.

Objective 7.1.e – Allow the user to add flows into the graph created when given a valid path and flow.

This objective has been met, this allows a user to add a valid flow as long as the path found is connected and the flow is less than the minimum flow left to send down an edge, displaying this in a table format with the path and flow. This shows appropriate error messages if the invalid augmented path and flow inputs are entered.

Objective 7.1.f – Allow the user the ability to delete nodes/edges/flows that are entered incorrectly.

The objective has been met, this allows the user to delete nodes and edges allowing the graph to be amendable, removing the node or edge information from the screen as well as the edge and node on the graph. The flow is deleted by it either being invalid when the graph is solved or an edge or node is deleted in which lies on the path.

Objective 7.1.g - The ability to display the nodes and edges in a clear, coherent manner, grouped together based on a corresponding node.

This objective has been met, displaying the edges and nodes information, with the edges grouped underneath the node information in which the edge has come from, with the information being able to be shifted up or down to view all data if the data needed to be put in the screen exceeds the space given, operated by buttons.

Objective 7.1.h – The ability to display the augmented paths added within a table format in a clear and coherent manner.

This objective has been met, this allows the path and flow to be displayed in different columns. If a path is added that already exists, this will update the flow rather than add a new record within the table.

Objective 7.2.a - The ability to determine based on the incoming and outgoing edges, whether a node is a source or sink.

This objective has been met, identifying if a node is a source and sink.

Objective 7.2.b – If there is multiple sources or sinks, it has the ability to create a new node for the supersource and/or supersink, with valid weight which has no impact on the network flow graph, (greater than outgoing edges of the sources or ingoing edges of sinks).

Objective 7.2.c - Ability to find the position of a given node and edge for a supersource/supersink

Objective 7.2.d - Display the weights on the edges, which is not amendable to the user.

The objectives 7.2.b, 7.2.c and 7.2.d has been met, finding a valid position for the supersource or/and supersink, accounting for the ingoing or outgoing edges and their maximum capacities as well as the positions of the sources/sinks, without limiting the optimal flow through the graph.

Objective 7.2.e - Error should be identified if there isn’t at least both a source and sink.

This objective has been met, displaying an appropriate error message, maintaining the graph in edit phase as graph not solvable.

Objective 7.2.f - Ability to hide supersources and/or supersinks corresponding node and edges using a button.

Objective has been met, this allows the associated nodes and edges of a supersource and supersink to be hidden or shown when the button has been pressed, returning to the supersource and supersink being shown if a cut is being drawn.

Objective 7.2.g - Ability to account for node capacities within the sources and sinks, shifting edges.

This objective has been met, this allows the supersource and supersink edges to be shifted to one the node capacity locations, maintaining the graph as connected, allowing the sources or sinks to have their own node capacity value. This has the ability to swap with the original edge depending on if the node capacities are being shown or hidden.

Objective 7.3.a - Ability to display a coordinate axis onto the area to aid the user entering coordinates for the nodes.

This objective has been met. This displays the coordinate axis acting as an arbitrary scale for the user to enter nodes onto the graph. This is removed when the user enters the solve phase.

Objective 7.3.b – Represent a node as a circle with the node identifier within.

This objective has been met, this draws any node, if a valid position is entered that means it is in a valid area, drawn as described above with a circular outline if it is a source or sink. As edges are entered, if the source/sink has both an ingoing and outgoing edges, this will convert the node to a circle without the circular outline, identifying it as a node.

Objective 7.3.c - Represent an edge as an arrow from one node to another with the capacity, with 2 smaller arrows, both parallel to the edge representing the forward/backward flow and with its corresponding flow values.

This objective has been met, this draws the edge between nodes, with the main arrow pointing towards the node the edge is going to, with the forward arrow being on the same orientation and the backward arrow being mirrored, with the minimum and maximum capacity being displayed in the middle of the main arrow. The forward flows and backward flows will also be centred on their respective arrows.

Objective 7.3.d - The ability to draw the graph based on the coordinates given by the user as seen on the screen by the grid like structure with supersource and supersinks added if valid positions found.

This objective has been met, this allows the coordinates entered by the user for a node to match the position on the graph, with the supersource and supersink being displayed at a valid position on the graph, with the correct edges and maximum capacities on each edge.

Objective 7.3.e – The ability to check if the new node or edge added is valid and does not intersect another node and edge.

This objective has been met, this allows the node and edge inputs to be rejected, displaying a suitable error message if there is an intersection, maintaining the graph as planar.

Objective 7.3.f – The ability to only display minimum capacities within the graph if and only if there is at least 1 with a minimum capacity greater than 0.

This objective has been met, this allows the user to see a minimum capacity if and only if there is at least one edge which has a minimum capacity in the graph made.

Objective 7.3.g – For any given node capacity, it has the ability to split the node into 2, with a given edge in between, without colliding with other objects on the graph, remaining the node identifier to a suitable alternative.

This objective has been met, if there is a valid position for the node capacity, this splits the node into 2, drawing this onto the graph, shifting the incoming edges to the one of the split nodes and the outgoing edges to the other, displaying when the node capacity is meant to be shown, with an edge connecting the split nodes. This renames the nodes with the node ID’s plus a suffix of “-1” or “-2”.

Objective 7.3.h - Ability to segment the flows into individual steps so can be seen and paused if needed.

This objective has been met, this allows through the next button, the flow to be added in segments, displaying the path and flow in a table format.

Objective 7.3.i - Ability to change any type of source or sink to a different colour/shape to make it more identifiable.

This objective has been met, moving the node from having a circular outline if a source or sink to a circle without that separate circular outline.

Objective 7.3.j – Ability to hide/show node capacities when drawn onto the graph.

This objective has been met, this allows, once the node capacities has been drawn, for the associated nodes and edges to be hidden or shown depending on when the node capacity button is pressed.

Objective 7.4.a - Able to find all possible paths from a source to a sink. If minimum capacities occur, minimum flow should be found before additional flow can be found, accounting for the augmented paths added by the user.

This objective has been met, this finds the maximum flow through the network from source to sink, finding the minimum flow first through the graph, displaying the path and flow in a table format. This can either remove the augmented flow if the augmented flows added do not match the maximum and minimum flows found or add onto the paths if further flow is required. With each flow added, this should update the flow of the edge in the paths.

Objective 7.4.b – Ability for user to preset the flow within the graph, changing the necessary flows within the edges.

This objective has been met, this allows the user to add an augmented flow, changing the flow on the edges in which the path goes through, displaying this in a table format. If the augmented path has flow too large for that path, this displays an error message.

Objective 7.4.c - Ability to account for added supersource/supersinks if the valid position can be found.

This objective has been met, this allows the maximum and minimum flows to be found for multiple sources and sinks, treating each path as having a starting point at the supersource and the end point of the supersink, adding the flow onto the associated edges. If no valid position for either supersource or supersink, flows will start from the sources or end at the sinks.

Objective 7.4.d - Ability to traverse node capacities within the graph, without exceeding the value given.

This objective has been met. This allows the maximum and minimum flows to be found through a node capacity, treating each path as going through the node as if it was split, accounting for the node capacity value if this is contained within the path, adding flow to the associated edges. If a valid position for a node capacity cannot be found, flows will go through the node, accounting for the node capacity despite it not being drawn.

Objective 7.4.e - Ability to display path and there flows in a separate area on screen with the option to split this into flow for minimum capacities and additional flow with a total stated.

This objective has been met. Once a path and flow has been added by the next button, this will display this in a table format in the minimum flow table if the flow is a part of reaching the minimum capacity value of at least one edge in the path or the maximum flow table.

Objective 7.4.f - Ability to update the screen in regards to forward and backward flows for each node.

This objective has been met, updating the flows of the edge when the next button is pressed.

Objective 7.4.g - Ability to make edges with a raised minimum capacity more identifiable once it has changed.

This objective has been met. If a path is a minimum path, but not all edges have any flow to add that is below the minimum capacity, this can raise the minimum flow temporarily to accommodate minimum flow through graph, highlighting this with a background of yellow. The minimum capacity will be reset once the program has been returned to the edit phase.

Objective 7.4.h - Allows backward flow if 0 to be more identifiable for better visualisation of remaining paths left to augment or minimum cut once maximum flow is found.

This objective has been met, highlighting the background of the backward flow when it is 0, meaning no more flow can be sent down that edge, making a more identifiable optimum cut.

Objective 7.4.i - Ability to do the algorithm in segments, adding the flow and paths one at a time.

This objective has been met. Through the next button, this allows the flow and paths for the maximum flow to be added individually, displaying this in a table format, updating the edges as required.

Objective 7.5.a - The ability to allow the user to draw their own cut onto the graph, automatically labelling it if valid.

This objective has been met, allowing cuts to be drawn by the mouse being pressed and the space bar being pressed to terminate the cut, labelling the cut C followed by the previous number incremented by 1, starting at C1. The colour of the cut changes for every cut, rotating after each one is drawn.

Objective 7.5.b - Ability to determine and analyse the cut drawn and find cut value, ignoring or subtracting the minimum capacities if going from sink 🡺 source.

This objective has been met, by using collisions with the edges to determine whether a node is on the source side or sink side, adding the edges maximum capacities if going from the source to the sink side and removing the minimum capacities if going from the sink side to the source side. This displays an error message when the cut drawn is invalid, removing that cut from the screen.

Objective 7.5.c – Ability to account for supersource, supersinks and node capacities when calculating the cut value.

This objective has been met. This allows the user to separate the split nodes of the node capacity onto separate side of the graph, updating the cut value as required, with the supersources and supersinks have no effect on the cut value due to the connection with sources and sinks.

Objective 7.5.d - The ability to display a cut using the specific notation on the screen, based on the users cut.

This objective has been met. Using the results from objective of 7.5.b, this displays a cut in the format {}/{} with source side being on the left and the sink side being on the right, displaying this as part of a table for each cut.

Objective 7.5.e – Ability to display calculation to find the cut value.

This objective has been met. This is displayed as part of the table for each cut, showing the sums of and totals for the maximum capacities going from the source side to the sink side and the minimum capacities for the edges going from the sink side to the source side, finding the cut value by subtracting the smaller value from the larger value.

Objective 7.5.f - Ability to hide the drawn cuts from the screen.

This objective has been met. Through a show/hide button for each valid cut drawn, this hides/shows the associated cut when the button is pressed.

Objective 7.5.g - Ability to randomly generate colour of line to make it easier to identify.

This objective has been met. This changes the colour after each cut is drawn, rotating the colour as the next cut colour, repeating after 5 cuts.

Objective 7.6 - Compare the maximum flow and minimum cut, displaying an appropriate message to show optimum flow has been found.

This objective has been met. This displays a message of the optimal flow value, given the maximum flow and a cut value are equal.

1. Self – evaluation:

This project has been a great learning experience not only developing a better understanding of the topic network flows, but also in terms of OOP. By developing my use of association and the importance of both using interfaces as well as creating private, protected and public attributes, this has enhanced my understanding of key structures and relationships within the classes.

In my program, I am very pleased with the functional tools used throughout the user interface, particularly the scrollbar class which enables full access to the displayed data no matter if the data exceeds the given space, allowing a more versatile displaying method. This is particularly due to there being a variable number of nodes and edges on the screen for any one graph. Similarly, the dynamic generation of lines to be displayed once the graph is in the table format to separate the records, enables an easier to understand format of the data, specifically if the table has multiple records related to each other. The scrollbar, combined with the other classes acting as widgets such as buttons, enable an easier graphical user interface to understand, with the formatting being similar for each component.

I am also pleased with the ability to find the maximum and minimum flow by accounting for the augmented paths already entered. Through the ability to split the augmented paths into the maximum and minimum flows, or remove these if these are not in the optimal flow, combined with the ability to split minimum flows to maintain the minimum flow to be as small as possible and raising minimum capacities if required, this enables a solution to be identified which is clearly displayed. This can be accessed step-by-step using the next button. This enables the path, once displayed, to be explained without confusion to which path the explanation refers to. While the raised temporary minimum components when the backward flow is 0 is identified by a yellow background, this could aid the ability to predict the pattern when drawing optimal cuts.

I am also pleased with the ability to build, analyse and hide/show cuts. This enables the cuts to be built in line segments, allowing a variable shape to collide with the edges wanted to by the user. This allows a generation of all possible cuts to be shown, allowing a greater versatility when teaching, enabling other cuts to be added and explained other than the optimal cut. By analysing the cut, this enables the cut to be determined if valid and allows the cut notation to be determined, allowing notation and values to be associated with the cut, splitting the cut value into the impacts of each collided edge. Combined with the ability to hide the cuts, this enables a greater focus on the cut the teacher wants rather than all of them. This also helps simplify the graph if loads of cuts were added and it was hard to determine which is which.

I am pleased with the generation of the supersources, supersinks and node capacities, only generating a valid position if there is space and no collisions. However, this is not always the case, meaning the ability for the program to handle these features without a valid position, enables the maximum flow minimum cut theorem to be still proved. By starting the paths with multiple sources instead of a supersource and ends at multiple sinks instead of a supersink, with the notation matching this, this enables a greater versatility of graphs to be shown independent on the user’s inputs for positions of nodes and edges. Combined with the ability to hide and show these added features, this enables a comparison between the original graph to enable the impact to be shown. Although this has the ability to cope with these features not having a valid position, in very specific situations, if there are multiple added features, this can prevent other added features to be added visually due to their positions causing collisions. Therefore, the ability to store the valid positions and use the one with the most added features drawn would be advantageous visually or the ability to in a phase between the edit and the solve phase, to be able to drag and drop the current nodes on screen to enable further features to be made, reverting back to the original if there are collisions once the solve phase has been entered. This allows easier amendments to the graph to enable such features to be drawn to aid the problem visually.

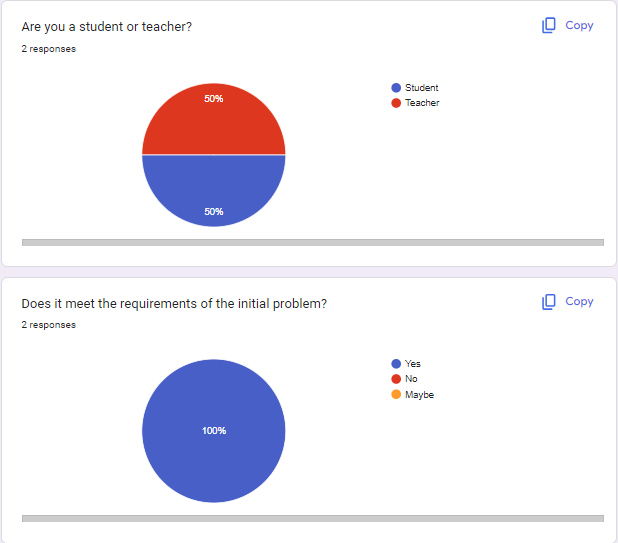
If I was going to re-implement the same program, I would utilise the use of dictionaries to get the node and edge index’s from when the nodes and edges are first being entered, rather than once the graph reaches the solve phase, allowing greater time efficiency within the start of the program. Although the graph is limited to a small number of nodes specified in my limitations, this change should enable little impact to the run time even on larger graphs.

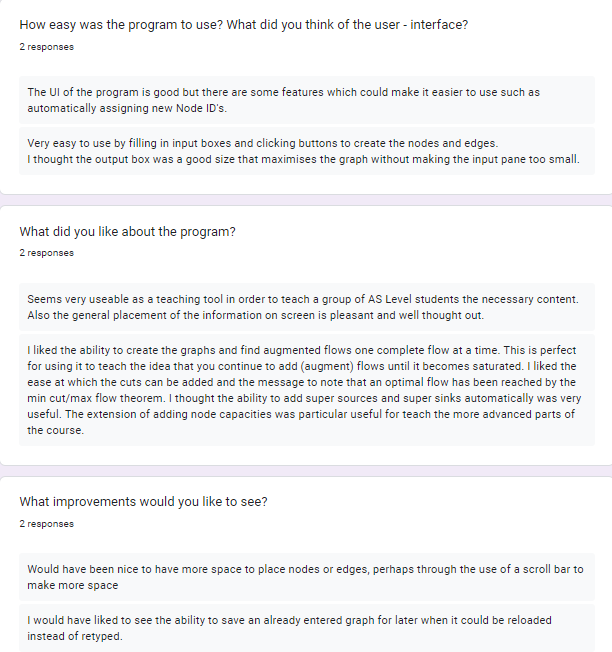
I would also enable the feature to delete augmented flows in the edit phase, through the use of a delete button adjacent to the scrollbar, allowing a more time efficient way of removing the flows without having to remove an edge or node.

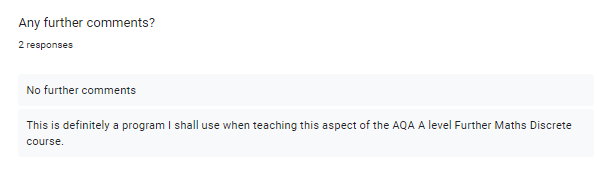
I would also reduce the time complexity of the collision algorithms, evident due to the large number of collision checks when adding a node capacity particularly if not valid as getting 2 lists of positions and checking pair of possibilities. This could be implemented by storing a list of all nodes and all edges once the solve phase has been entered, meaning this can be accessed each time rather than building up the list.

Although the changes suggested in this section are not within the objectives, these are features that would enable further functionality within the program which may be useful and aid the user’s interaction with the program.

1. User feedback/ Analysis of feedback:







This is feedback from both a student and the teacher within the further maths in relation to the use of the program as a teaching resource, with the first response being related to the student who does further maths and the second response to the teacher, who gave overall positive feedback.

The student’s response analysed:

He liked the simplicity of the design of the user interface, particularly the similar formatting of all the buttons and scrollbar. He also liked the use of the default text to prompt a certain type of input when building up a graph, allowing a focus on the inputs being expected for the required components. Although it did require an initial minute to explain the different phases of the program and the keys for building up cuts, he found most of it to be self-explanatory, enabling him to explore the program further, aiding his current understanding of the topic.

He liked the idea that it was versatile when covering the A-level content, allowing a variety of graphs to be drawn, coping with the different aspects of the topic from node capacities to maximum and minimum flows. He also liked the format of all the different sections, with information being useful in the different aspects of the program, with the ability to format the graph similar to any graph when being replicated.

He thought that although the space to draw the graph was quite big, the ability to extend the area to draw the graph would be useful, particularly when finding the optimal flow to graphs which may need more space. This would also enable a greater possibility of their being a valid position for the supersource, supersink and node capacity, meaning the graph will be more visually representative of all features, despite having the ability to cope if there is not a valid position. This was suggested through a scrollbar, which has the functionality to zoom or pan, making the nodes and edges variable in size.

The teacher’s response analysed:

The teacher liked the distribution of sections and the proportions within the screen, particularly in relation to the input boxes and the graph being drawn on, allowing both to be large enough to be visual even at the back of a classroom, while not taking away from the graph component of the program. The teacher also liked the simplistic design to input the graph, with each component’s inputs being grouped adjacent to each other, which is submitted with the button.

In terms of the features within the graph, the teacher liked the ability to augment flows before solving the graph to show the effect on the edges along the path, with the ability to format and change the shape of the graph to match a diagram, building it up from each node and edge. This allows therefore a more versatile graph to be made. The teacher also liked the ability to give the maximum flows in segments, step-by-step, until it becomes saturated, meaning no more flow can be sent down from the source to sink along a path, enabling this to teach the idea that flows are added individually. Combined with the ability to create a cut matching the needs of the user, this enables greater adaptability, particularly as the cut is user defined, with the ability to hide and show cuts being mentioned specifically when walking through the program. Furthermore, the teacher was also pleased with the added features of adding the supersource, supersink and node capacity being auto generated enabling a visual aid to the more advanced areas of the topic, particularly with the node capacity which may further limit the flow.

However, the main user would have liked to see the ability to save and reload the graph rather than being retyped, which would be time efficient rather than re-entering the same graph. This would be particularly useful for the teacher if they taught multiple classes or were entering large graphs, meaning a greater time can be spent on teaching and could work with minor impacts to the current program.

Overall, the main user was pleased with the program as acting as a teaching aid for the network flows, covering all the specification for all aspects of the topic.

1. Possible extensions:

As mentioned directly above in the teacher’s feedback, the ability to save and load existing graphs would be particularly useful, being time efficient when teaching, particularly if multiple classes were being taught per year. This would enable graphs to be pre-saved, acting as existing examples rather than the graph being retyped. This would also require the ability to save or view existing files that can be accessed, possibly by further increasing the width of the screen, using a scrollbar to view all file names. Although this would create another section on the window, this could by default also lead to an even bigger graph area. This means little referencing or remembering the graph needed when retyping. This would be implemented using a normalised database using SQL. An example layout is shown below within the database and the primary and composite keys being underlined,

FileInfo(FileID, FileName)

NodeInfo(Node ID, FileID, X-ordinate, Y-ordinate, nodeCap, type)

EdgeInfo(EdgeID, FileID, FromNodeID, ToNodeID, minCap, maxCap)

FlowInfo(FileID, flow, path)

Being as cuts and flows can be generated quite quickly, the database would be used to store the values in the edit phase rather than the solve phase, particularly as these can be generated also.

Similarly, as mentioned in the student response above, the ability to add a scrollbar to the graph, allowing the graph to be zoomed in or out, or even panned to enable a shift vertically or horizontally to account for super long or tall graphs. This would be useful, enabling a greater area for nodes and edges to be displayed. This would be useful for the text on the nodes and edges to maintain readable by changing the font size as required as well as ensuring the full graph is visible.

As mentioned in my self-evaluation, the ability to add a phase in between the movement from edit phase to solve phase, to enable movement of nodes through a possible drag and drop feature. This would be useful both to quickly amend the positions of the nodes as well as to enable the maximum number of supersources, supersinks and node capacities if required in the graph. Although this could also have the option to remove the changes if collisions occur with the original graph from the edit phase. This is useful to allow greater versatility to changing the shape of the graph and would be implemented by checking collisions when clicked with nodes, acting similar to the algorithm to shift the end of the line for the cuts when they are being drawn. It might also be useful to add the added features to be seen if there is a valid position so the user can view how many of the features are viewable.

Another extension which is also mentioned in the self-evaluation is the ability to delete the augmented flows from the scrollbar using a button, similar to the way nodes and edges are currently deleted. This button would only be displayed in association to the record of augmented flows and would not be visible in the solve phase, meaning this is only amendable in the edit phase. This would be more accessible for augmented flows initially to be changed depending on possible changing graphs in questions.

Another feature could be to highlight the path in which has just been added when the next button has been pressed, possibly in a different colour, allowing the user to focus on the path and flow just added rather than the graph as a whole. This could be implemented similar to when the temporary minimum capacity is raised to a higher value than the minimum capacity. This would be useful in identifying parts of the graph, with the possibility of this disappearing once all flows have been visited to allow a focus on the graph as a whole rather than individual paths. Furthermore, this can be added to cuts, which possibly with the press of a button to highlight the collided edges of the cut, with varying colours depending on whether the edge is going from the source to sink or sink to source. This change in colours would act as a reference point to focus in on the edges that affect the cut value.

Although one of my acceptable limitations is to have no cycles within the graph, the ability to cope with this would be useful to allow a broader range of graphs, which could be implemented by adding a check and accounting for visited nodes in the depth first search algorithm.

1. Conclusions of project:

From the feedback given, it is clear the program would be useful in lessons in all sub-topics within the network flows component of the discrete further maths course, being particularly useful when the students are learning the topic. Although there are small changes such as maximising the number of added features drawn, due to this having the ability to cope without a valid position, this has minimal impact on the user interaction with the program. There are also some ideas that can be added to the system, allowing greater time efficiency when using the program like having the ability to save and load previous graphs, which would be beneficial particularly for the teacher in the long term.

References:

Accessed 28th April: <https://www.aqa.org.uk/subjects/mathematics/as-and-a-level/further-mathematics-7367/specification-at-a-glance> - P 29

Accessed 29th April: AQA A-level Further Maths Year 1 + Year 2 Student Book (Brian Jefferson, David Bowles, Eddie Mullan etc.) (Z-library)

Accessed 29th April: AQA A-level Further Mathematics Discrete – Geere, Nick – 2018 – Hodder Education

Accessed 30th April: [TD-SD1-05 Network Flows.pdf](file:///I:\MATHS\Further%20Maths\D1%20Graphs%20and%20Algorithms\TD-SD1-05%20Network%20Flows.pdf)

Accessed 30th April: FM Discrete Teaching Guidance – P36🡺 66

If the resource is a pdf document, this is within the reference resources folder on the USB.