NEA Computer Science

4 Testing:

1. Test plan:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test No | Purpose | Associated objective number | Test Data | Expected Result | Result |
|  | Checks that the valid parameters for a node can be inputted without a node capacity, resetting the input boxes after entering. Ensuring the correct information is displayed within the scrollbar | 7.1.a, 7.1.d, 7.3.b, 7.3.d | I will enter “A” into the Node ID input box, 1 in the X ordinate input box and 1 into the y ordinate input box and enter the same node with a node ID of “AA” the second time at position (3,3) with a node capacity of 0 | Accepted both, drawing the nodes at arbitrary position (1,1) and (3,3), with information regarding each node appearing in the scrollbar above with node ID “A” or “AA” and node capacity of 0, with the node input boxes being reset | Pass |
|  | Checks that the valid parameters for a node can be inputted with a node capacity, resetting the input boxes after entering. Ensuring the correct information is displayed within the scrollbar | 7.1.a, 7.1.d, 7.3.b, 7.3.d | I will enter “A” into the node ID input box, 2 in the node capacity input box, 1 in the X ordinate input box and 1 into the y ordinate input box and the node ID “AA” at a position of (3,3) with a node capacity of 3 | Accepted both , drawing both nodes at arbitrary position (1,1) and (3,3), with information regarding each node appearing in the scrollbar above with node ID “A” and node capacity of 2 as well as node ID “AA” with a node capacity of 3, with the node input boxes being reset | Pass |
|  | Checks that if Node ID is left empty when inputting node information that an error message is received | 7.1.a, 7.1.d | I will enter 1 into the x ordinate input box and 1 into the y ordinate input box | Rejected inputs, adding a suitable error message within the error scrollbar each time, resetting the node input boxes | Pass |
|  | Checks that if x and y coordinate is left empty when inputting node information, that an error message is received | 7.1.a, 7.1.d | I will enter “A” into the node ID input box and 1 into the y ordinate input box and “A” and 1 into the x ordinate input box | Rejected input, adding a suitable error message within the error scrollbar, resetting the node input boxes for both set of inputs | Pass |
|  | Checks any Node ID, which has a maximum length exceeding 2 | 7.1.a | I will enter “SSS” into the node ID input box with 1 into the x ordinate input box and the y ordinate input box | This should accept the input, however this should only show a node ID of SS as limiting node ID to a length of 2 | Pass |
|  | Checks if any characters that are not numbers can be entered into the node capacity, x and y input box including negatives | 7.1.a, 7.1.d | I will enter “-K” into the node capacity, x and y coordinate input box with each only one “-K” being entered | Rejected input, this should not update the node capacity, x and y coordinate input boxes | Pass |
|  | Checks if node ID input box contains a reserved node ID, e.g for supersource and supersink | 7.1.a | I will enter “S” into the node ID input box, 1 into the x ordinate input box and 1 into y ordinate input box | Rejected input, a suitable error message should be displayed, resetting input boxes | Pass |
|  | Checks if the duplicate node ID will be accepted | 7.1.a | I will create a node with a node ID, node capacity, x and y coordinates, (“A”, 2, 1, 1) and try entering this node ID again at position of (3,3) | Accepts the first input, but rejects the second input, showing a suitable error message, resetting the node input boxes | Pass |
|  | Checks if any x ordinates can be entered outside of the given range on screen | 7.1.a, 7.3.d | I will enter 2 nodes with a node ID, node capacity, x and y coordinates, with inputs (“A”, 2, 0, 1) and (“A”, 1, 14, 1) respectively | Rejects both node inputs, showing 2 suitable error messages within the error scroll bar, resetting the node input boxes | Pass |
|  | Checks if any y ordinates can be entered outside of the given range on screen | 7.1.a, 7.3.d | I will enter 2 nodes with a node ID, node capacity, x and y coordinates, with inputs (“A”, 2, 1, 0) and (“A”, 1, 1, 7) respectively | Rejects both node inputs, showing 2 suitable error messages within the error scroll bar, resetting the node input boxes | Pass |
|  | Checks if multiple decimal points can be placed for given coordinate inputs | 7.1.a | I will enter “..” in the x and y ordinate input boxes respectively | Will display “.” in both input boxes with the last keyboard input not being shown on the screen | Pass |
|  | Checks if the node is accepted if position of new node overlaps existing node or within a certain distance of the node to enable an edge to be drawn | 7.3.b, 7.3.e | I will enter a valid node at position (3, 3) with node ID “A” and another node at positions (2,3), (4,3), (3,2), (3,4) and (5, 5) each with a node ID of “F” | Will display node “A” and reject the first 4 node inputs with a suitable error message appearing in the scrollbar for each invalid node, with the coordinate set being accepted ad drawn onto the screen | Pass |
|  | Checks to see if node is accepted with coordinates which are not integers | 7.1.a, 7.3.a | I will enter a node “A” at (2.2, 3.5) onto the graph | This will display the node at this position on the screen | Pass |
|  | Checks that a valid edge can be added between 2 valid nodes with no minimum capacity across the edge | 7.1.b, 7.1.g  7.3.c | I will enter 9 nodes “A”, “B”, “C”, “D”, “E”, “F”, “G”, “H” and “I” at (1,1), (3,1), (5, 1), (1,3), (5, 3), (1,5), (3, 5), (5, 5) and (3,3), adding an edge going from “I” to all other node as all other nodes to “I” of maximum capacity of 5. | Will display all 16 edges with the correct main, forward flow and backward flow arrows all with a maximum capacity and backward flow of 5, ensuring they are in the correct orientation. This should display the information of the edge directly below the information of the node in which the edge has come from. This should also reset the edge input boxes | Pass |
|  | Checks that a valid edge can be added between 2 valid nodes with minimum capacity across the edge | 7.1.b, 7.1.c, 7.1.g, 7.3.c, 7.3.f | I will enter 2 nodes “A” and “B” at (1,1) and (3,3), adding an edge going from “A” to “B” with maximum capacity of 5 and minimum capacity of 1 | Will display an edge with main arrow going from “A” to “B” displaying the minimum and maximum capacity with forward flow being 0 on the forward arrow and 5 being on the backward arrow, ensuring these are in the correct orientation for the edge, displaying the information of the edge directly below the information of the node in which the edge has come from. This should also reset the edge input boxes | Pass |
|  | Checks that the edge will be accepted only if the minimum capacity is <= the maximum capacity | 7.1.b, 7.1.c, 7.1.g | I will enter an edge between “A” and “B” which are nodes at positions (1,1) and (3,3) respectively, with (2,1), (2,2), (1,2) with the numbers representing the minimum capacity and the maximum capacities respectively of each node | Will display a suitable error message for the first edge inputted, but will accept the second and third edges, drawing these onto the screen, resetting the input boxes each time | Pass |
|  | Checks if the node ID entered for start and end positions of edge exist and are drawn on the graph | 7.1.b | I will enter an edge going from “A” to “B” 3 times, once with no nodes entered, 1 with only “A” and then only “B” inputted as a node onto the screen, all with a maximum capacity of 2 | Will display a suitable error message for all 3 edge inputs as at least one node has not been entered into the input boxes for the graph, resetting the input boxes each time | Pass |
|  | Checks if the 2 node ID’s entered are not the same as an edge can’t go to itself. | 7.1.b | I will enter a valid node “A” onto the screen and input an edge going from “A” to “A” | Will display a suitable error message that an edge can’t go to itself, resetting the input boxes | Pass |
|  | Checks if any of the reserved node ID’s will be accepted as a start or end point of the edge | 7.1.b | I will enter an edge going from “S” to “A”, “A” to “T” and “S” to “T” where “A” is a preset node on the graph with each having a maximum capacity of 1 or 5 | This should reject all 3 inputs, displaying a suitable error message for each incorrect input, resetting the input boxes for the edges each time | Pass |
|  | Checks if the minimum and maximum capacity rejects any character other than a number including “.” | 7.1.b, 7.1.c | I will enter “-G“ into both the minimum and maximum capacity input boxes | This will reject all inputs in both input boxes, adding no characters to either | Pass |
|  | Checks if an edge can be added with data omitted from either the from node, to node and the maximum capacity input boxes | 7.1.b | I will enter nodes “A” and “B” at positions (1,1) and (3,3) respectively with an edge going from “A” to “B” of maximum capacity of 5 being added 3 times with 1 value being omitted each time which is different from the previous time | This will reject all 3 inputs, displaying a suitable error message for each incorrect input, resetting the input boxes each time | Pass |
|  | Checks if an edge is rejected if the edge collides with a node, either the main arrow or the forward and backward arrow | 7.3.b, 7.3.c, 7.3.e | I will enter valid nodes “A”, “B” and “C” at positions (1,1), (3,3), (5,5) respectively. I will enter an edge going from “A” to “C”, before shifting C to (5,4) before re-entering the same edge, with both edges having a maximum capacity of 5 | This will reject both the first and second edge inputs due to the main arrow colliding with the node and the smaller arrow on the latter test | Pass |
|  | Checks if an edge is rejected if the edge collides with another existing edge through the main edge | 7.3.c, 7.3.e | I will enter nodes “A”, “B”, “C” and “D” with positions (1,1), (3,3), (1,3), (3,1) respectively, entering an edge from “A” to “B” and from “C” to “D” of maximum capacity of 2 | This will draw the first edge but reject the second edge due to a collision, displaying an appropriate error message | Pass |
|  | Checks if the edge is rejected if the small forward and backward arrows collide with the edge | 7.3.c, 7.3.e | I will enter nodes “A”, “B”, “C” and “D” at (3,3), (5,5), (2,1) and (7,6) with an edge going from “A” to “B” and from “C” to “D” with a maximum capacity of 2 | This will draw the first edge but reject the second edge due to an overlap of the forward flow arrow, displaying the appropriate error message | Pass |
|  | Checks if a node is rejected if the node collides with an existing edge when colliding with both the main arrow and the forward and backward flow arrows | 7.3.b, 7.3.c, 7.3.e | I will enter “A” and “B” at positions (1,1) and (5,5) respectively with an edge connected from “A” to “B”. I will enter a node “C” which will have positions (3,3) and (4,3) respectively. Node “B” will then be shifted to (6, 6) and “C” to (4,3) | This should reject C on the first occasion, displaying a suitable collision error message and accept C on the second occasion as this does not collide with forward and backward flows. On the 3rd input, “C” should be rejected due to collision with forward and backward flow arrows | Pass |
|  | Delete a node and corresponding edges from the graph and the scrollbar displaying the information for each node and edge deleted | 7.1.a, 7.1.b, 7.1.f, 7.3.b, 7.3.c | I will enter nodes “A” and “B” at positions (1,1) and (3,3) respectively with an edge going from “A” to “B” of maximum capacity of 5. The delete button for “A” and then “B” will pressed | Both inputs should be accepted, with the first delete button pressed causing the edge and the node ”A” to be removed from both the screen and the displaying information within the scrollbar, with the second delete button removing “B” from both the screen and scrollbar | Pass |
|  | Delete an edge from the screen and the scrollbar displaying the information for each edge | 7.1.b, 7.1.f, 7.3.c | I will enter nodes “A” and “B” at positions (1,1) and (3,3) respectively with an edge going from “A” to “B” of maximum capacity of 5. The delete button for the edge will then be pressed | The input should be accepted with the edge removed from the screen and the scrollbar | Pass |
|  | Checks that valid parameters can be entered for an augmented flow, updating the correct edges | 7.1.e, 7.1.h | I will enter nodes “A” and “B” at positions (1,1) and (3,3) respectively with an edge going from “A” to “B” of maximum capacity of 5. An augmented path of “A,B” with a flow of 2 will be added | This should accept all inputs, adding a forward flow of 2 and reduce the backward flow by 3, displaying the path and flow in a separate scrollbar, resetting the associated input boxes | Pass |
|  | Checks that the valid augmented flow can account for node capacities | 7.1.e, 7.1.f | I will use the same input from test 28 but the node capacity of “A” will be 1 and the augmented flow will be 1 added twice instead of 2 with the same augmented path of “A,B” | This should reject the second augmented flow but accept the first augmented flow due to the limiting capacity of the node capacity, adding a flow of 1 to the path, displaying the path and flow in a separate scrollbar, resetting the associated input boxes | Pass |
|  | Checks that the path input box only accepts valid node ID’s which have an edge | 7.1.e | I will enter nodes “A” and “B” at positions (1,1) and (3,3) respectively with an edge going from “A” to “B” of maximum capacity of 5. Augmented paths of “B,A”, “A,B,A” and “S,A” each with flow of 2 will be added | This should reject all 3 inputs adding no flow to any edges and displaying a suitable error message for each input, resetting the associated input boxes | Pass |
|  | Checks the flow input box to only accept integers | 7.1.e | Add “GH.-“ into the flow input box for the augmented paths | This should not update the input box so that “GH.-“ is omitted as invalid values | Pass |
|  | Checks that the flow sent through the augmented path cannot exceed the maximum capacity of each edge | 7.1.e | I will enter nodes “A” and “B” at positions (1,1) and (3,3) respectively with an edge going from “A” to “B” of maximum capacity of 5. An augmented path of “A,B” with a flow of 7. The node “C” at (5,5) will be added with the edge BC having a maximum capacity of 3. An augmented path of “A,B,C” and flow of 4 will be added | This should reject the first input as edge can’t have a flow of 7 so a suitable error message should be displayed. This should reject the second input as a flow of 4 exceed the maximum capacity of the edge BC but not AB, therefore rejecting this augmented path and flow, displaying an error message also | Pass |
|  | Checks if an augmented path is accepted with 1 of the 2 values omitted | 7.1.e | I will enter nodes “A” and “B” at positions (1,1) and (3,3) respectively with an edge going from “A” to “B” of maximum capacity of 5. An augmented path “A,B” and 2 twice with 1 distinct value omitted each time | This should reject both inputs, displaying a suitable error message, resetting the augmented path input boxes after each input | Pass |
|  | Checks if a node is deleted, this removes the path and flow from each edge in the path | 7.1.e, 7.1.f | I will enter nodes “A” and “B” at positions (1,1) and (3,3) respectively with an edge going from “A” to “B” of maximum capacity of 5. An augmented path of ”A,B” with a flow of 2, will be added. The delete button for “A” will be pressed, the graph reset to previous layout and the edge delete button will be pressed | This should accept both inputs, deleting the augmented path from the nodes and edges as well the maximum flow scrollbars when the node “A” is deleted | Pass |
|  | Checks if an edge is deleted, this removes the path and flow from each edge in the path | 7.1.e, 7.1.f | I will enter nodes “A”, “B” and “C” at positions (1,1), (3,3) and (5,5) respectively with an edge going from “A” to “B” and “B” to “C” of maximum capacity of 5. An augmented path “A,B,C” with flow of 2 will be entered. The edge going from “A” to “B” will then be deleted. The path of “A,B” will then be entered with all nodes and edges reset, with the edge “BC” having a maximum capacity of 3 and a flow of 1 added, with the solve button then being pressed | This should accept the first augmented path. When the edge has been deleted, this should remove the path and flow from the edge “B” to “C”, accepting the inputs and removing the flow. This should accept the second augmented path initially but delete it once the solve button has been pressed as invalid flow | Pass |
|  | Checks that the edges are connected only moving to another node if an edge exists, ensuring it doesn’t accept the reserved node ID’s | 7.1.e | I will enter nodes “A”, “B” and “C” at positions (1,1), (3,3) and (5,5) respectively with an edge going from “A” to “B” of maximum capacity of 5 and “B” to “C” with a maximum capacity of 2. Then an augmented path of “A,B,C” and “B,C,A” each with a flow of 1 | This should accept the first path, updating the edges as required while rejecting the second path as edges are not connected, displaying a suitable error message | Pass |
|  | Checks if a node can move from being identified as a source/sink to a node and back | 7.3.b, 7.3.i | I will enter nodes “A”, “B” and “C” at positions (1,1), (3,3) and (5,5) respectively with an edge going from “A” to “B” and “B” to “C” of maximum capacity of 5. Then the edge BC should be deleted before re-adding the edge with a maximum capacity of 5. Node “A” is then being deleted. | Once all edges have been inputted, this will change node “B” to a node with a single outline rather than 2, and converting back to a node with a single outline after the node has been deleted or the edge has been deleted | Pass |
|  | Checks if the graph is connected and is valid for maximum flows and minimum cuts, checking if there is at least a source and sink and it is connected when converting to a solve phase | 7.2.a | I will enter nodes “A”, “B” and “C” at positions (1,1), (3,3) and (5,5) respectively with an edge going from “A” to “B” of maximum capacity of 5. I will then press the solve phase button when there are no nodes and just “A” on the screen. | This should not accept the graph each time, converting it back to the edit phase, displaying a suitable error message, keeping the phase of the button the same | Pass |
|  | Checks if the edit/solve button will change to the solve phase due to there being a source and sink, ensuring the delete buttons and the graph disappears and reappears when it returns to the edit phase | 7.3.a, | I will enter nodes “A” and “B” at positions (1,1) and (3,3) respectively with an edge going from “A” to “B” of maximum capacity of 5. Solve button will then be pressed | This will accept all inputs converting the button to the solve phase, hiding the graph and changing the message of the button to identify that when pressed. It will return to the edit phase, which when pressed will cause the graph to appear | Pass |
|  | Ability to check if the graph has a minimum flow <= maximum flow for each node | 7.3.a | I will enter nodes “A”, “B” and “C” at positions (1,1), (3,3) and (5,1) respectively with an edge going from “A” to “B” with maximum capacity of 1 and “B” to “C” with a maximum capacity of 5 and minimum capacity of 2. The solve button will then be pressed | This should keep the screen in the edit phase as graph is invalid due to the minimum capacity > maximum capacity | Pass |
|  | Checks if changed to solve phase, and there are more than 1 source, a supersource will appear with appropriate edges of minimum and maximum capacity in a valid space without collisions with existing edges and nodes | 7.2.b, 7.2.c, 7.2.d | I will enter nodes “A”, “B” and “C” at positions (1,1), (3,3) and (5,1) respectively with an edge going from “A” to “B” and “C” to “B” of maximum capacity of 5 and minimum capacity of 2. The solve button will then be pressed | Once the solve button has been pressed, this should display a source of node ID “S” with edges having no impact on the graph flow, connecting the sources together, (“S” to “A” and “S” to “B”) | Pass |
|  | Checks if changed to solve phase, and there are more than 1 source but no valid space for a supersource, error is shown but graph can still be solved | 7.2.b, 7.2.e | I will enter nodes “A”, “B” and “C” at positions (1,1), (3,3) and (3,1) respectively with an edge going from “A” to “B” with minimum and maximum capacity of 2 and 5 respectively and “C” to “B” of maximum capacity of 5. The solve button will then be pressed | This will display a suitable error message as not a valid position of supersource found, with no new nodes or edges appearing | Pass |
|  | The supersource and its associated nodes and edges will be hidden/shown depending on the phase of the button, only appearing if a supersource has a valid position | 7.2.f | Once a valid supersource has been entered, e.g test 41, the hide/show supersource button will be pressed twice | Both inputs will be valid with the first shifting the phase of the button to show and hide all associated edges and nodes with the second input changing the phase of the button again to show and reshow all associated edges and nodes of the supersource | Pass |
|  | Checks if the program is changed to solve phase, and there are more than 1 sink, a supersink will appear with appropriate edges of minimum and maximum capacity in a valid space without collisions with existing edges and nodes | 7.2.b, 7.2.c, 7.2.d | I will enter nodes “A”, “B” and “C” at positions (1,1), (3,3) and (5,1) respectively with an edge going from “A” to “B” with maximum capacity of 5 and minimum capacity of 2 and “A” to “C” with maximum capacity of 5. The solve button will then be pressed | Once the solve button has been pressed, this should display a sink of node ID “T” with edges having no impact on the graph flow, connecting the sinks together, (“B” to “T” and “C” to “T”) | Pass |
|  | Checks if the program is changed to the solve phase, and there are more than 1 sink but no valid space for a supersink, an error is shown but graph can still be solved | 7.2.b, 7.2.e | I will enter nodes “A”, “B” and “C” at positions (1,1), (3,3) and (3,1) respectively with an edge going from “B” to “A” with maximum capacity of 2 and “B” to “C” with a minimum and maximum capacity of 2 and 5 respectively. The solve button will then be pressed | This will display a suitable error message as not a valid position of supersink found, with no new nodes or edges appearing, but remaining in the solve phase. | Pass |
|  | Check to see if a supersink is hidden/shown when correct button is pressed | 7.2.f | Once a valid supersink has been entered, (inputs from test 44 are used), then hide/show supersink button will be pressed twice | Both inputs will be valid with the first shifting the phase of the button to show and hides all associated edges and nodes with the second input changing the phase of the button again to hide and reshowing all associated edges and nodes of the supersink | Pass |
|  | Checks if the graph of a single node with a node capacity is accepted, checking if the associated edges and nodes can be hidden by the show/ hide node capacity button | 7.3.g, 7.3.j | I will enter “A” at position (1,1) with node capacity of 1, the solve button will then be pressed and the show/ hide node capacity button will be pressed twice | Once the solve button has been pressed, this should display 2 nodes of “A-1” and “A-2”with edges having no further impact on the graph flow, connected together by a flow of 1, with the graph being hidden on the first button press and shown again on the second button press | Pass |
|  | Checks if the program is changed to the solve phase, and there is a node with a node capacity, this  will appear with appropriate edges of minimum and maximum capacity in a valid space without collisions with existing edges and nodes | 7.3.g | I will enter nodes “A” with a node capacity of 1, “B” and “C” at positions (1,1), (3,3) and (5,5) respectively with an edge going from “A” to “B” and “B” to “C” of maximum capacity of 5 The solve button will then be pressed | Once the solve button has been pressed, this should display 4 nodes of which 2 are node capacities of “A” with edges having no further impact on the graph flow, connecting the graph together | Pass |
|  | Checks to see if a node capacity and a supersource can be drawn where a source has a node capacity, checking if associated edges are hidden or shown when in solve phase | 7.3.g, 7.3.d, 7.2.g | I will enter nodes “A” with node capacity of 1, “B” and “C” at positions (1,1), (3,3) and (5,1) respectively with an edge going from “A” to “B” and “C” to “B” of maximum capacity of 5 The solve button will then be pressed and then the show/ hide node capacity button will be pressed twice and the show/ hide supersource button twice | Once the solve button has been pressed, this will display the node capacity and supersource at valid positions. This should then hide all associated node capacity edges when the node capacity button is pressed, re showing the associated nodes and edges, with similar occurring for the supersource components when the supersource button is pressed | Pass |
|  | Checks to see if a node capacity and a supersink can be drawn where a sink has a node capacity, checking if associated edges can be hidden or shown when in solve phased | 7.3.g, 7.3.d, 7.2.g | I will enter nodes “A” with node capacity of 1, “B” and “C” at positions (1,1), (3,3) and (5,1) respectively with an edge going from “B” to “A” and “B” to “C” of maximum capacity of 2. The solve button will then be pressed and then the show/ hide node capacity button will be pressed twice and the show/ hide supersink button twice | Once the solve button has been pressed, this will display the node capacity and supersink at valid positions. This should then hide all associated node capacity edges and nodes when the node capacity button is pressed, re showing the original nodes and edges presplit. This is then reversed if the node capacity button is pressed again. The associated nodes and edges for the supersink components should be hidden and then shown when the supersink button is pressed | Pass |
|  | Checks to see if a flow can be created from a simple graph correctly | 7.4.a, 7.4.e, 7.4.f, 7.4.h | 2 nodes of “A” and “B” with positions (1,1) and (3,3) respectively are inputted with an edge going from “A” to “B” with maximum capacity of 3. This is repeated twice with the latter edge also having a minimum capacity of 1. The solve button and then the next button will be pressed | This should create a path of “1,3” with maximum flow of 3 and then minimum flow of 1 and maximum flow of 2 on the second set of inputs, displaying the paths and flows in the table format expected, shifting the tables as required depending on if there is data in 1 or both of the scrollbars used, updating the flow of each edge. This should shade the backward flow in yellow when 0 | Pass |
|  | Checks if the flows found can be added in segments through the next button being pressed | 7.4.e, 7.4.f, 7.4.h, 7.4.i | Continuing with the preset of test 41 as well as with edges of maximum capacity of 3, test 44 with all edges having a minimum and maximum capacity of 2 and 5 respectively, test 48, 49 and 50 with a maximum capacity of 5 for all edges, the next button should add each path and flow individually for each graph, highlighting the backward flows which are 0 if they occur | The next button should only remain functioning while there are paths left to be added, with each path being added individually once the button has been pressed, raising the forward flows of each edge and displaying a new record with the path and flow added, these being the same as mentioned in the test directly above. A maximum flow should be found for each graph coping with different added features | Pass |
|  | Checks if the minimum capacity can temporarily be raised for an edge to allow a minimum flow to be found | 7.4.g | Given nodes “A”, “B” and “C” at positions (1,1), (3,3) and (5,5) respectively with an edge from “A” to “B” with minimum capacity of 1 and maximum capacity of 3 and an edge from “B” to “C” with maximum capacity 4. The solve button and the next button will then be pressed | This should when the flow has been added after the solve button and next button has been pressed, raise the minimum capacity temporarily of edge going from “B” to “C” by 1, highlighting the change, creating a total flow of 3 split between the minimum and maximum flows | Pass |
|  | Checks if maximum flow can be found with a supersource/supersink | 7.4.c | Given nodes “A”, “B”, “C”, “D” and “E” with positions (1,1), (5,1), (3,3), (1,5) and (5,5) respectively with edges “A” to “C”, “B” to “C”, “C” to “D” and “C” to “E” with maximum flows and minimum flows of 3 and 1. The solve button will then be pressed followed by the next button | This should have a total flow of 6 with both the supersource and supersink edges being added with a raised temporary minimum capacity after the minimum flow and maximum flows added | Pass |
|  | Account for minimum flows to be split along the paths | 7.4.d | The nodes “A”, “B”, “C” and “D” will be entered at positions (1,1), (3,3), (5,5) and (5,1) respectively. The edges will also go from “A” to “B” with minimum and maximum capacities of 2 and 3, “B” to “C” and “B” to “D” with the latter 2 edges having a minimum and maximum capacity of 1 and 3. The solve button will be then pressed along with the next button | Once the solve button has been pressed, and the supersink position has been found, this should display the maximum flow of 3, with no raised minimum capacities and a minimum flow of 1 being sent down both “A, B, C, T” and “A, B, D, T” with a maximum flow of 1 being sent down any of the 2 paths | Pass |
|  | Checks an added flow can be split into the minimum and maximum flow when moving from edit phase to solve phase and its flow increased if the path is a duplicate | 7.4.b | With nodes “A” and “B” at (1,1) and (3,3) respectively with an edge between with minimum capacity and maximum capacity of 1 and 3, with an augmented path “A,B” and initial flow of 2. The solve button will then be pressed and then the next button pressed | This should split the flow when solve button is pressed to have 1 in both the maximum and minimum flow tables. When the next button is pressed, the maximum flow will increase by 1 in regards to the additional flow and should be displayed accordingly | Pass |
|  | Checks when the solve button is pressed if the path is still valid meaning it starts and ends at the source and sink | 7.4.b | By continuing with the graph of test 56 in the edit phase, add a node “C” with an edge between “B” and “C” of maximum capacity of 2. The solve button will then be pressed | This should delete the path and existing flow from the edge going from “A” to “B” as invalid, with a suitable error message | Pass |
|  | Checks that maximum flow accounts for node capacity if it is in the path | 7.4.d | With node capacity “A” of 5 at position (1,1), and nodes at “B” and “C” at positions (3,3) and (5,5) with edges going from “A” to “B” with minimum and maximum capacity of 3 and 6 respectively, “B” to “C” with maximum capacity of 6 and the solve button should be pressed followed by the next button | Once the solve button has been pressed and the valid position is found for the node capacity, the graph creates a minimum flow of 3 and a maximum flow of 2 due to the node capacity acting as a restriction, creating total flow of 5 down “A-1, A-2, B, C” | Pass |
|  | Account for supersource and supersink if not valid as well | 7.4.c | Given nodes “A”, “B”, “C”, “D” and “E” with positions (1,1), (3,1), (3,3), (1,3) and (1,5) respectively with edges “A” to “C”, “B” to “C”, “C” to “D” and “C” to “E” with maximum flows and minimum flows of 3 and 1 respectively | No valid positions for either a supersource or supersink, but maximum flow from the graph should be displaying a value of 6, with paths going from sources and ending at sinks with a minimum flow of 3 and maximum flow of 4 | Pass |
|  | Checks to see if a cut can be drawn if mouse button pressed, and last segment to follow position of mouse | 7.5.a | With nodes “A” and “B” at (1,1) and (3,3) respectively with an edge between the 2 node and a minimum capacity and maximum capacity of 1 and 3. Once in the solve phase and the next button has been pressed, start a cut by clicking on the mouse and tracing the outline of the screen | This should start the cut and the end position of the line should follow the mouse up to the border but no further, with buttons being pressed being independent of the cut. | Pass |
|  | Checks if correct key is used to signal cut is finished, analysing the cut correctly | 7.5.a, 7.5.b, 7.5.d, 7.5.e, | Continuing from test 60, if space bar is pressed, the cut will have ended. 2 cuts will be drawn, 1 going through the valid edge giving a cut of {A}/{B} and the other not | This should display the valid cut, displaying the correct notation of {A}/{B} and cut value of 3 but display a suitable error message for the cut that does not intersect the edge | Pass |
|  | Multiple cuts will be added with the hide and show buttons working despite the use of the scrollbar | 7.5.a, 7.5.g | Continuing from test 60, cuts will be drawn that are valid, until the scrollbar with the cuts information is being shifted downwards. The down and up button should be pressed once | This should change the colour of each cut after each valid cut, with the scrollbar shifting downwards when down is pressed and upward when up button is pressed by 2 records as associated records are adjacent | Pass |
|  | The cuts may also be hidden or shown by a button press, which can hide all cuts no matter whether they exceed the space given | 7.5.f | By continuing from the graph in test 62, hide button is pressed before shifting the records to show a different set of cut ID’s and their corresponding button, and hiding/showing any other cuts | This should change any of the associated cut ID’s from being hidden to seen and vice versa and remain in that phase until the button is pressed again no matter whether the button is displayed in a different position as to before | Pass |
|  | Check collisions of cut drawn | 7.5.a | By following from test 63, try drawing the cuts parallel to the edge, colliding to the edge, colliding with only the forward small arrows of the edge | This should only accept the second input as the other do not separate the source and sink onto separate parts of the graph | Pass |
|  | Adding a cut onto a graph with a supersource, node capacity and/or supersink | 7.5.c | Continuing from graphs in test 41, 44 with all edges having a minimum and maximum capacity of 2 and 5 respectively, test 48, 49, 50 and 54, add a variety of valid and invalid cuts between the sources and supersource, between the sinks and supersink and the node capacities | This should accept or reject each cut depending on if the cut splits the source/supersource and sinks/supersink as well as if the cut does not collide with nodes | Pass |
|  | Account for edges going from the sink side to the source side of the graph | 7.5.b | The nodes “A”, “B”, “C”, “D” and “E” are entered at positions (1,1), (3,3), (7,3), (3,5) and (6, 5) respectively with edges going from “A” to “B”, “A” to “C”, “C” to “B”, “B” to “D”, “B” to “E” and “C” to “E” respectively. Each edge inputted can have any maximum capacity with edge “C” to “B” having a minimum capacity of 0 and then 1. The solve button will then be pressed along with the next button and all valid cuts will be drawn along with the invalid cuts | Once the solve button has been pressed, this should generate a supersink of a valid position and mark the cut as valid if the cut splits the source/ supersource and sinks/ supersink as well as if the cut does not collide with nodes. This should if going through edge CB remove 1 off the cut value when the edge has a minimum capacity of 1, but not when it has a minimum capacity of 0 | Pass |
|  | A suitable message should be displayed if the maximum flow value found is the same as the minimum cut value found | 7.6 | By continuing test 51, after drawing a valid cut, across the edge going from “A” to “B” and next button has no more functionality, (all flows added possible) | Should display a message with the optimal flow of 3, proving the optimal flow in the graph | Pass |
|  | Shows a full run through of a single node with a node cap from entering to finding the optimal flow, showing cuts are valid within a single node capacity | 7.5.c | This adds a node “A” at (1,1) with a node capacity of 1. The solve button is then pressed along with the next button with a cut drawn between “A-1” and “A-2” | This should give an optimal flow of 1 with all flows and cut information within the table displayed correctly as well as on the graph, with the cut being {A-1}/{A-2} | Pass |
|  | Shows a full run through of the problem with a graph that does not have a valid position for a supersource or supersink, showing cuts can be validated without a supersource or supersink present | 7.5.c | By continuing from the graph in test 59, the solve button and the next button will be pressed, with a cut drawn of {A,B}/{C,D,E} and {A,B,C}/{C,D}, while drawing invalid cuts | Given no supersource and supersink position is valid, once the solve button is pressed and the next buttons pressed, the 2 valids cuts ({A,B}/{C,D,E} and {A,B,C}/{C,D}) both have cut value of 6, giving an optimal flow of 6, displaying this within the table in the correct format | Pass |
|  | Shows the ability to account for multiple node capacities connected to each other | 7.3.d, 7.4.d, 7.5.c | This enters the nodes “A” at (1,1) with a node capacity of 2 and “B” at (3,3) with a node capacity of 3, joined by the edge going from “A” to “B” with a maximum capacity of 2. The solve button will then be pressed, followed by the next button with all 3 valid cuts drawn, ({A-1}/{A-2,B-1,B-2}, {A-1,A-2}/{B-1,B-2}, {A-1,A-2,B-1}/{B-2}) | This should draw both node capacities onto the screen once the solve button has been pressed. Once the next button has been pressed a flow of 2 should be added down the path “A-1,A-2,B-1,B-2”, with the cuts {A-1}/{A-2,B-1,B-2} and {A-1,A-2}/{B-1,B-2} having a cut value of 3 and {A-1,A-2,B-1}/{B-2}) having a cut value of 2, giving an optimal flow of 2. This should display the flows and cuts found in a table format | Pass |
|  | Handling a node capacity that does not act as a source or sink | 7.3.d, 7.4.d, 7.5.c | The user enters the nodes “A” and “C” at (1,1) and (5,5) with node “B” of node capacity 2 at (3,3) with edges AB and BC having a maximum capacity of 3. The solve button is then pressed along with the next button being pressed. 3 cuts will also be added, ({A}/{B-1,B-2,C}, {A,B-1}/{B-2, C} and {A,B-1,B-2}/{C}) | This should display a valid position for the node capacity of “B” once the solve button is pressed. A maximum flow of 2 should be seen sent down “A,B-1,B-2,C” and the cuts {A}/{B-1,B-2,C} and {A,B-1,B-2}/{C} both have a cut value of 3 and {A,B-1}/{B-2, C} has a cut value of 2, showing the flow and cuts in a table format. This should highlight the optimal flow as 2 | Pass |
|  | Removing added flows from a graph if the augmented path is initially valid but within the paths for the optimal solution | 7.4.a | The user should enter nodes “A”, “B”, “C”, “D”, “E” and “F” at positions (1,1), (3,3), (3,1), (6,3), (6,1) and (9,2) respectively. All edges AB, AC, BD, BE, CE, EF and DF all have a maximum capacity of 2. An augmented path “A,B,E,F” will then be added with a flow of 2. The solve button will then be pressed followed by the next button | This should keep the augmented path “A,B,E,F” of flow 2, until the next button is pressed in which it will remove the path and 2 further paths of flow 2, giving a total flow of 4, displaying this in a table format | Pass |

1. Evidence:

| Test Number | Video Reference | What is seen |
| --- | --- | --- |
|  | NodeValidation from 0 to 34 seconds | The user is entering a node ID of possible length 1 and 2, with valid positions on the grid displaying the 2 nodes once inputted with the node IDs at the centre, resetting the input boxes back to their default text and the information on the node being displayed in the top left hand corner |
|  | NodeValidation from 34 seconds to 1 minute 7 seconds | The user is entering the same values above but using the optional property of the nodes having a node capacity and therefore accepting this at the valid positions of the grid. This resets the input boxes back to their default text and the information on the node being displayed in the top left hand corner |
|  | NodeValidation from 1 minute 7 seconds to 1 minute 22 seconds | The user attempts to add a node onto the graph without entering a suitable node ID, receiving a suitable error message in the bottom right hand corner of “Not all required node inputs were completed” |
|  | NodeValidation from 1 minute 22 seconds to 1 minute 52 seconds | The user attempts to enter a node ID without a suitable x or y coordinate, which is rejected, giving a suitable error message in the bottom right hand corner of “Not all required node inputs were completed” |
|  | NodeValidation from 1 minute 52 seconds to 2 minute 17 seconds | The user attempts to enter a node ID of length greater than 2 with “SSS”, with the input box preventing the user from adding a 3rd character, adding the node “SS” instead |
|  | NodeValidation from 2 minute 17 seconds to 2 mins 53 seconds | The user attempts to add invalid characters into the node capacity, x and y coordinate input boxes, “-K” which due to this only accepting integer inputs, rejects these added characters, without changing the text within the input boxes |
|  | NodeValidation from 3 minute 9 seconds to 3 minute 25 seconds | The user attempts to add a node with a node ID of “S” which is reserved due to this being the node ID used if the graph requires a supersource, giving a suitable error message in the bottom right hand corner of “ID reserved” |
|  | NodeValidation from 3 minute 25 seconds to 3 minute 50 seconds | The user attempts to add a second node with the same node ID as the first node “A”, resulting in confusion within the graph if accepted. This therefore rejects the second node, giving a suitable error message in the bottom rand hand corner of “Node ID can’t be repeated” |
|  | NodeValidation from 3 minute 50 seconds to 4 minute in 35 seconds | The user attempts to place a node “A” where x is 0 and 14, which although on the coordinate graph, it would place a part of the node outside on the area preset for node and edge drawing. This means both node inputs are rejected, displaying a suitable error message in the bottom right hand corner of “Coords are not within given parameter” |
|  | NodeValidation from 4 minute 35 seconds to 5 minute 6 seconds | The user attempts to place a node “A” where y is 0 and 7, which although on the coordinate graph, it would place a part of the node outside on the area preset for node and edge drawing. This means both node inputs are rejected, displaying a suitable error message in the bottom right hand corner of “Coords are not within given parameter” |
|  | NodeValidation from 5 minute 6 seconds to 5 minute 29 seconds | Checks if the user can enter multiple “..” into the coordinate input boxes as these allow decimal values, preventing from invalid numbers from being enters. This omits the character pressed, only adding the single “.” to the input box |
|  | NodeValidation from 5 minute 29 seconds to 7 minute 5 seconds | The user attempts to add a node too close to an existing node so part of the new node overlaps part of the existing node, with all but position (5, 5) being accepted for this reason. A suitable error message is displayed in the bottom right hand corner for all nodes rejected of “Nodes must not touch” |
|  | TestVideo6-part1 from 0 seconds to 18 seconds | The user adds a valid node “A” at position (2.2, 3.5) showing that the coordinate input boxes accept float inputs when within the possible ranges of the graph |
|  | EdgeValidation from 0 seconds to 2 minute 22 seconds | This use “I” as a centre node, validates all positions node positions in relation to another node, showing the correct orientation of all arrows once drawn whether going from or to “I”, adding each edge information under the from node in the top left hand corner, without the minimum capacity being greater than 0 and resetting the edge input boxes each time |
|  | EdgeValidation from 2 minute 22 seconds to 2 minute 56 seconds | This shows the user can enter an edge with an optional minimum capacity of 1, with this value being displayed, both next to the maximum capacity and within the edge information in the top left hand corner of the screen |
|  | EdgeValidation from 2 minute 56 seconds to 3 minute 27 seconds | This shows an edge will only be accepted if the minimum capacity is <= the maximum capacity, due to the initial input being rejected due to 2 > 1. This displays an error message in the bottom right hand corner of “Min capacity must be less than or equal to the maximum capacity”. However the further 2 inputs, satisfy the statement above, therefore accepting this and drawing the edge between the 2 nodes |
|  | EdgeValidation from 3 minute 27 seconds to 4 minutes | This rejects the edge the user has entered when an insufficient number of node ID’s have been entered for the edge, either 1 or none, rejecting the input and displaying an error message in the bottom right hand corner of “Not all required edge inputs were completed” |
|  | EdgeValidation from 4 minutes to 4 minute 13 seconds | The user attempts to input an edge going to and from the same node, this rejects this as this would cause an infinite loop, meaning the graph is no longer simple, displaying an error message of “An edge can’t go to itself” and “1 or more of Node ID’s do not exist” as this only counts distinct node IDs entered |
|  | EdgeValidation from 4 minute 13 seconds to 5 minute 59 seconds | The user tries to add edges between reserved node ID’s and existing node ID’s, which is rejected as the reserved node ID may not have a valid position at the current point in the program. This displays a suitable error message in the bottom right hand corner of “1 or more of Node ID’s do not exist” |
|  | EdgeValidation from 5 minute to 5 minute 24 seconds | The user tries to add “-G” into both the minimum capacity and maximum capacity input boxes to show that only integers are valid, omitting all characters which were entered |
|  | EdgeValidation from 5 minute 24 seconds to 5 minute 51 seconds | The user tries to enter an edge omitting either a node ID for the from or to node input boxes or a maximum capacity, rejecting the edge as not enough information to draw the edge, displaying a message in the bottom right hand corner of “Not all required edge inputs were completed” |
|  | EdgeValidation from 5 minute 51 seconds to 6 minute 36 seconds | The user attempts to send 2 edges through “B”, one where the edge’s main arrow collides directly with the arrow and the second with the smaller forward and backward arrows, rejecting the edge due to the collision with existing nodes, displaying a message in the bottom right hand corner of “Edge collides with other nodes” |
|  | EdgeValidation from 6 minute 36 seconds to 7 minute 7 seconds | The user attempts to draw an arrow over an existing edge “AB” when trying to connect “CD”, this rejects the new edge due to the overlap of the main edge, displaying a suitable error message of “Edge collides with other edges” |
|  | EdgeValidation from 7 minute 7 seconds to 7 minute 57 seconds | This user attempts to send edges parallel to each other in close proximity so that the smaller forward and backward arrows collide with each other, causing a collision. Due to the graph being planar, this mean the second edge is rejected, displaying a suitable error message in the bottom right hand corner of the screen of “Edge collides with other edges” |
|  | EdgeValidation from 7 minute 57 seconds to 8 minute 48 seconds and Flow-Validation from 0 seconds to 35 seconds | The user tried to enter a node overlapping an existing edge as well as a node in close proximity at (4,3). This rejects the first input due to the graph needing to be planar, which due to the collision would no longer make it planar, displaying a suitable error message in the bottom right hand corner of “Node collides with at least 1 edge”. However, the second node is valid due to there being no collisions with existing edges. The third input is also an invalid node, displaying the same message above, however the collision is with the smaller forward and backward arrows instead of the main arrow |
|  | Flow-Validation from 37 seconds to 1 minute 17 seconds | This shows the deletion of nodes to amend the graph shown, first with an edge attached to the node and then a single node, showing that that the edges are grouped with the nodes if deleted as the node ID is no longer valid for part of the edge, shifting up the delete buttons if necessary |
|  | Flow- Validation from 1 minute 17 seconds to 1 minute 45 seconds | This deletes a single edge from the graph, showing that the edges although they can be deleted with a node, can be deleted by themselves |
|  | Flow-Validation from 1 minute 45 seconds to 2 minute 4 seconds | This shows the user adding a valid flow down a graph, displaying this in a table format, and updating the edges so the forward flow is 2 and the backward flow is 3 due to augmented flow of 2 |
|  | Flow-Validation from 2 minute 4 seconds to 2 minute 49 seconds | This shows that a path checks the valid flow against both edges and node capacities, allowing the first flow, but rejecting the second due to the previous flow already being sent down the path preventing any further flow. This was due to after the first augmented flow, there is no more flow left for the second input, displaying a suitable error message of “Additional flow entered is too great” |
|  | Flow-Validation from 2 minute 49 seconds to 3 minute 49 seconds | This shows that only paths which are connected and has each edge along the path existing that are accepted as valid, evident with “B,A”, “A,B,A” being rejected as well as “S,A” due to “S” being a reserved node ID not an existing one, with a suitable error message appearing in the bottom right hand corner of “Atleast one edge does not exist” |
|  | Flow-Validation from 3 minute 49 seconds to 4 minute 28 seconds | This shows the only values entered into the flow input box that is accepted is integers as the user entered “GH.-“, which accounts both for string and float possibilities, is omitted from the input box |
|  | Flow-Validation from 4 minute 28 seconds to 4 minute 53 seconds  testsContinuations (Called test 29 continues) from 0 seconds to 47 seconds | This shows that an augmented path will only be accepted if the path is valid for that amount of flow, evident with path “A,B” where the limit is 5 and the flow is 7, (7 > 5). This displays a suitable message in the bottom right hand corner of “Additonal flow entered is too great”. This also shows in the second graph that the augmented path is rejected due to the path “A,B,C” having a limiting flow of 3 due to the edge BC so the augmented flow added of 4 is invalid, displaying the same error message as before |
|  | Flow-Validation from 4 minute 53 seconds to 5 minutes | The user enters a path “A,B” that already has an augmented flow of 1, meaning when the path is repeated, this increments the flow value by the new flow wanting to be sent down if the added flow is valid, resulting in the flow being displayed moving from 1 🡺 2 |
|  | Flow-Validation from 5 minutes to 5 minute 18 seconds | The user deletes a node in the top left hand corner, removing the edge, node and flow, from the graph and the table where the augmented paths and flows are shown |
|  | Flow-Validation from 5 minute 18 seconds to 6 minute 29 seconds  testsContinuations from 47 seconds to 1 minute 22 seconds | This accepts the first path “A,B,C” with a flow 2, displaying this in a table format in the bottom right hand left hand corner of the screen , until the edge AB is deleted which removes the path from the table, removing the flow from the edges which was added. The second path added “A,B” of flow 1 is initially accepted due to the graph still being changed, displaying this in a table format, however once the solve button is pressed, this will be deleted as “B” is not a sink therefore not a valid augmented flow, removing both from the table and the flow from the edge |
|  | SolvePhase from 0 seconds to 44 seconds | The user tries to enter 2 paths, “A,B,C” which is valid and “B,C,A” to validate that a path has to be connected. Although BC is an edge, CA is not so this augmented path is rejected, showing an error message in the bottom right hand corner of “At least one edge does not exist” |
|  | SolvePhase from 44 seconds to 58 seconds | This shows node B going from a node due to the ingoing edge AB and an outgoing edge BC, to a sink once BC has been deleted due to it only having an ingoing edge of AB, back to a node with the same edges as before and then to a source by deleting the node A, meaning B only has one outgoing edge. This is identified by the black circular outline of the node appearing and disappearing |
|  | SolvePhase from 58 seconds to 1 minute 19 seconds | This shows the user can only enter the solve phase if the graph is connected and there is more than one node entered as a single node does not have a distinct source and sink, displaying an error message of “Graph not connected” in the bottom right hand corner |
|  | SolvePhase from 1 minute 19 seconds to 1 minute 36 seconds | This shows if the user enters a valid graph, this enables the program to enter the solve phase where the graph can be solved |
|  | SolvePhase from 1 minute 36 seconds 2 minute 27 seconds | This rejects the graph as the minimum capacity of edge BC is 2 which is > the maximum capacity of AB which is 1, meaning as no valid flow can validate the path, the graph is invalid, displaying an error message in the bottom right hand corner of “Graph drawn is invalid” |
|  | SolvePhase from 2 minute 27 seconds to 3 minute 16 seconds | This shows the supersource has a valid position, drawing this onto the graph with the reserved node ID “S”, with the added edges and nodes having no collisions with existing edges and nodes, with edges “SA” and “SC” having a minimum and maximum capacities of 2 and 5 respectively matching edges “AB” and “CB” |
|  | SolvePhase from 3 minute 21 seconds to 4 minutes | This shows a graph which no valid position of supersource can be found between A and C, as any node and edge position causes a collision with existing nodes and edges, displaying an error message in the bottom right hand corner of “No valid supersource position found”. This still changes to the solve phase as the graph can still be solved without the visual feature |
|  | SolvePhase from 3 minute 16 seconds to 3 minute 21 seconds | This shows the supersource and its associated edges are hidden/ shown once the button is pressed, changing the text on the button to accommodate the change in functionality of the button |
|  | SolvePhase from 4 minutes to 4 minute 37 seconds | This shows the supersink has a valid position, drawing this onto the graph with the reserved node ID “T”, with the added edges and nodes having no collisions with existing edges and nodes, with edges “BT” and “CT” having a minimum and maximum capacities of 2 and 5 respectively matching edges “AB” and “AC” |
|  | SolvePhase from 4 minute 44 seconds to 5 minute 49 seconds | This shows a graph which no valid position of supersink can be found between A and C, as any node and edge position cause a collision with existing nodes and edges, displaying an error message in the bottom right hand corner of “No valid supersink position found”. This still changes to the solve phase as the graph can still be solved without the visual feature |
|  | SolvePhase from 4 minute 37 seconds to 4 minute 44 seconds | This shows the supersink and its associated edges are hidden/ shown once the button is pressed, changing the text on the button to accommodate the change in functionality of the button |
|  | SolvePhase 5 minute 49 seconds to 6 minute 13 seconds | This shows a graph is accepted with a single node if and only if that node has a node capacity of any value greater than 0, showing that the node can be split, moving from a graph of “A” to “A-1,A-2” once the node capacity button has been pressed |
|  | SolvePhase from 6 minute 13 seconds to 7 minute 5 seconds | This shows the node capacity has a valid position, drawing this onto the graph with node IDs “A-1, A-2” or “A”, with the added edges and nodes having no collisions with existing edges and nodes, with edge “A-1A-2” having a maximum capacity of 1. This maintains the graph as connected |
|  | SolvePhase from 7 minute 5 seconds to 7 minute 49 seconds | This shows that supersources can be handled if a source/node has a node capacity which is “A”, creating an additional edge to move from the supersource to “A-1”, allowing for a potential shift in starting position while maintaining the maximum capacity of 5 of the original edge. The associated nodes and edges are also seen to be hidden or shown correlating to the correct button, in all possible permutations of the 2 button phases |
|  | maxFlows from 0 seconds to 55 seconds | This shows that supersinks can be handled if a sink/node has a node capacity which is “A” in this case, creating an additional edge to move from the “A-2” to the supersink, allowing for a potential shift in starting position while maintaining the maximum capacity of 5 of the original edge. The associated nodes and edges are also seen to be hidden or shown correlating to the correct button, in all possible permutations of the 2 button phases |
|  | maxFlows from 55 seconds to 1 minute 59 seconds | The first maximum flow found of the graph once the next button is pressed, set the maximum flow of 3 on the path “A,B”, displaying this in a table format in the bottom left hand corner, with the total flow of 3. The second maximum flow still finds a total flow of 3, but due to the edge “A,B” having a minimum and maximum capacity of 1 and 3 respectively, this means the flow was split into a minimum flow of 1 and maximum flow of 2, accounting for the minimum flows separately evident due to the path being in a separate table in the bottom left hand corner. Both graphs also update the edge based on the path and flow just entered into the table through the next button being pressed, incrementing the forward flow and decrementing the backward flow by 3 in the first graph and by 1 and then 2 in the second graph, highlighting the backward flow is 0 when no more flow can be found along that edge |
|  | maxFlows from 1 minute 59 seconds to 6 minute 4 seconds   * maxFlows from graph in test 41 – 1 minute 59 seconds to 3 minute 14 seconds * maxFlows from graph in test 44 – 3 minute 14 seconds to 3 minute 45 seconds * maxFlows from graph in test 48 from 3 minute 45 seconds to 4 minute 43 seconds * maxFlows from graph in test 49 from 4 minute 43 seconds to 5 minute 17 seconds * maxFlows from graph in test 50 from 5 minute 17 seconds to 6 minute 4 seconds | * Continuation from graph in test 41 – This shows a graph with a valid supersource position found, first with edges AB and CB of maximum capacity of 3, giving a total flow of 6 through the paths “S,A,B” and “S,C,B”, which is displayed in the bottom left hand corner of the screen and then the same edges with a minimum capacity of 2 and a maximum capacity of 5, meaning a minimum flow of 4 is found, with 2 in each of the paths above and 6 in the total maximum flow, with 3 in the each of the above paths giving a total flow of 10 as seen in the bottom left hand corner. Both times, when the next path and flow is added to the graph, the flow is also shown to be added to the supersource edges SA and SC * Continuation from graph in test 44 – This shows a graph with a valid supersink position found, with edges AB and AC of minimum capacity of 2 and a maximum capacity of 5 This means a minimum flow of 4 is found, with 2 in each of the paths “A,B,T” and “A,C,T” and 6 in the total maximum flow, with 3 in the each of the above paths giving a total flow of 10 as seen in the bottom left hand corner. When the next path and flow is added to the graph, the flow is also shown to be added to the supersink edges BT and CT * Continuation from graph in test 48 – This graph shows a maximum flow being found with a node capacity “A” of 1, limiting the maximum flow within the graph. This produces a total flow of 1 down “A-1,A-2,B,C”, shown in the bottom left hand corner within the table format shown. When the next path and flow is added to the graph, this updates the node capacity edges as well as the normal edges by a flow of 1 * Continuation from graph in test 49 – This shows the graphs can account for both a supersource and a node capacity if one of the sources had a node capacity value and there are valid positions for both. This creates a flow of 1 in the path “S,A-1,A-2,2,B” due to the limiting flow of the node capacity of “A”, while handling the path “S,C,B” normally with a flow of 5, resulting in the total flow of 6 through the graph. This is displayed in the bottom left hand corner of the screen, with each supersource and node capacity edge having its forward flow raised and backward flow decreased by the flow added into the table * Continuation from graph in test 50 – This shows the graphs can account for both a supersink and a node capacity if one of the sinks had a node capacity value and there are valid positions for both. This creates a flow of 1 in the path “B,A-1,A-2,T” due to the limiting flow of the node capacity of “A”, while handling the path “B,C,T” normally with a flow of 5, resulting in the total flow of 6 through the graph. This is displayed in the bottom left hand corner of the screen, with each supersink and node capacity edge having its forward flow raised and backward flow decreased by the flow added into the table |
|  | maxFlows from 6 minute 4 seconds to 6 minute 47 seconds | The edge AB acts as both a limiting edge in terms of the flow and allows a raised minimum capacity as seen when the next button is pressed initially, raising the minimum capacity of BC to 1 until the program is entered back into the edit phase, giving a minimum flow of 1 along the path “A,B,C” and a maximum flow of 2. This gives a total flow of 3 through the graph. This displays this in the bottom left hand corner of the screen, highlighting with a yellow background the minimum capacity which is temporarily raised, updating the forward flows on each edge |
|  | maxFlows from 6 minute 47 seconds to 8 minute 31 seconds | This creates a graph which accounts for both a supersource and supersink when finding maximum flows with each edge AC, BC, CD and CE having a minimum and maximum capacity of 1 and 3 respectively giving a minimum flow of 2. This is by sending 1 flow down the paths “S,A,C,D,T” and “S,B,C,E,T” and maximum flow of 4, with 2 being sent down these paths. This handles multiple possible paths through a single node C, displaying this in the bottom left hand corner in a table format giving a total flow of 6 |
|  | maxFlows from 8 minute 31 seconds to 9 minute 28 seconds | This forces the minimum flow to be split up in order to find an optimum flow value due to the edge AB having a minimum capacity of 2 and edges BC and BD have a minimum capacity of 1. Therefore the optimum flow as seen in the bottom left hand corner is a minimum flow of 1 down paths “A,B,C,T” and “A,B,D,T”, before sending the maximum flow of 1 down any 1 on these paths, (in this case “A,B,C,T”), giving a total flow of 3, updating the forward flows of each edge. |
|  | TestVideo6-part1 from 18 seconds to 52 seconds | This shows the user entering a path “A,B” of flow 2 which is valid but greater than the minimum capacity of 1, therefore splitting the flows into a minimum flow and maximum flow of 1 with the same path, splitting as seen in the bottom left hand corner with the tables, without affecting the forward flow of the edge |
|  | TestVideo6-part1 from 52 seconds to 2 minute 4 seconds | Adding on the node C with the edge BC, means the augmented path “A,B” of flow 2 is invalid as this no longer starts and ends at a sink, meaning when the graph shifts to the solve phase, this deleted this augmented path, removing this from the forward flow of the edge AB |
|  | TestVideo6-part1 from 2 minute 4 seconds to 2 minute 52 seconds | This shows a valid node capacity position being found of node “A”, showing the edge BC having a raised temporary minimum capacity of 3, giving a minimum flow of 3 in the minimum path “A-1,A-2,B,C” and a maximum flow of 2 down the same path, giving a total flow of 5. This is displayed in the bottom left hand corner, raising the forward flow of all edges including the node capacity edges as shown when the node capacity button is pressed, showing/ hiding the node cap |
|  | TestVideo6-part2 from 0 seconds to 26 seconds | This shows a graph where no valid supersource and supersink positions, giving the minimum flow of 2, handling each source as a start point, (A and B) and handling each sink separately (D and E). This sends down a flow of 1 down each path of “A,C,D” and “B,C,E”. This also creates a maximum flow of 4, 2 in each path, giving a total flow of 6. This shows this in the bottom left hand corner, without exceeding the maximum capacity of the edges. |
|  | TestVideo6-part2 from 26 seconds to 1 minute 18 seconds | This shows a cut with a cut ID of “C1” with the mouse click both initialising a new cut or a segment within that cut, with the end of last segment following the mouse position until the edge of the valid area. This no longer updates the end position of the line segment of the cut if the mouse is outside of the area |
|  | TestVideo6-part2 from 1 minute 18 seconds to 1 minute 42 seconds | This shows 2 cuts being drawn, which can be set with the press of the space bar key. The first one which is invalid as this would result in the source and sink being on the same side as the cut due to the not colliding with any edges, displaying an error message in the bottom right hand corner of “Cut drawn is invalid”. However, the second cut which also has a cut ID of “C1” due to the last being invalid, split the graph {A}/{B} with a cut value of 3 as seen by the bottom middle of the screen, maintaining the drawn cut on the screen once analysed with a Cut ID and a show button appearing in the bottom left hand corner |
|  | TestVideo6-part2 from 1 minute 42 seconds to 2 minute | This shows multiple cuts being drawn, with cut ID being incremented by 1 showing distinct cut IDs ranging from “C1:C7” in the test, displaying each cut in the table in the bottom centre of the screen, scrolling down the table to ensure most recent cut added can be seen. Each edge also varies in colour, making it easier to identify a cut, with colours being blue, orange, pink, green and purple, rotating after each cut drawn |
|  | TestVideo6-part2 from 2 minute to 2 minute 30 seconds  testsContinuations from 1 minute 43 seconds to 2 minute 40 seconds | This shows each cut being hidden once the associated button has been pressed which are located in the bottom right hand corner of the screen, without affecting the table format. These are shown to be hidden and shown as required with scrolling up and down having no effect on the function of that area as shown with cut C3 |
|  | TestVideo6-part2 from 2 minute 30 seconds to 2 minute 47 seconds | This shows 3 cuts being found as invalid, the first showing the lack of collision with the main edge despite its close proximity due to it being parallel, the second shows that a collision with the smaller forward and backward arrows do not count as a collision with an edge, and the third shows that a collision with a node also invalidates a cut being as a node can’t be split without a node capacity. This displays an error message in the bottom right hand corner of “Cut drawn is invalid” for each invalid input, removing the invalid cut from the screen |
|  | TestVideo6-part2 from 2 minute 47 seconds to 8 minute 51 seconds and testsContinuations from 2 minute 40 seconds to 3 minute 50 seconds   * TestVideo6-part2 from graph in test 41 – 2 minute 47 seconds to 3 minute 57 seconds * TestVideo6-part2 from graph in test 44 – 3 minute 57 seconds to 4 minute 36 seconds * TestVideo6-part2 from graph in test 48 from 4 minute 36 seconds to 6 minute 11 seconds * TestVideo6-part2 from graph in test 49 from 6 minute 11 seconds to 7 minute 26 seconds * TestVideo6-part2 from graph in test 50 from 7 minute 26 seconds to 8 minute 51 seconds * testsContinuations from graph in test 54 from 2 minute 40 seconds to 3 minute 50 seconds | * Continuation from graph in test 41 – This shows a cut with a graph which has a supersource position valid and drawn on the screen. This accepts the first cut of {S,A,C}/{B} of cut value 10, 5 from each edge AB and CB, displaying this in the bottom centre of the screen. The cut {S}/{A,B,C} is seen to be rejected as although this splits the supersource to the other side as the sink, but the sources “A” and “C” are on the same side as the sink, therefore rejecting the cut. This also rejects the cut {S,C,B}/{A} and {S,A,B}/{C} due to the invalid split of sources and sinks. Similarly, the final cut seen collides with B invalidating the cut. This displays the error message in the bottom right hand corner of “Cut drawn is invalid” * Continuation from test 44 - This shows a cut with a graph which has a supersink position valid and drawn on the screen. This accepts the first cut of {A}/{B,C,T} of cut value 10, 5 from each edge AB and AC, displaying this in the bottom centre of the screen. The cut {A,B,C }/{T } is seen to be rejected as although this splits the supersink to the other side as the source A, the sinks “B” and “C” are on the same side as the source, therefore rejecting the cut. Similarly, the final cut seen collides with the supersink T invalidating the cut. This displays the error message in the bottom right hand corner of “Cut drawn is invalid” * Continuation from test 48 – This shows cuts with a node capacity within the given graph with the cuts {A-1, A-2}/{B,C} and {A-1,A-2,B}/{C} having a cut value of 5 and the cut {A-1}/{A-2,B,C} having a cut value 1. This displays this in the bottom centre of the screen. This also rejects the cut {A-1,B,C}/{A-2} due to the double collision resulting in the sources and sinks being on the same side, displaying a suitable error message in the bottom right hand corner * Continuation from test 49 – This shows cuts in a graph with a valid position for a supersource and node capacity A. This accepts 2 cuts, a cut {S,A-1,C}/{A-2,B} of cut value 6 and a cut of {S,A-1,A-2,C}/{B} of cut value 10, displaying both of these in the bottom centre of the screen. However, this rejects the cuts {S,A-1}/{A-2,C,B} and {S}/{A-1,A-2,C,B} due to the sources and sink not being on separate sides of the cut, invalidating it and displaying an error message in the bottom right hand corner * Continuation from test 50 – This cuts in a graph with a valid position for a supersink and node capacity A. This accepts 2 cuts, a cut {B }/{A-1,A-2,C,T} of cut value 10 and a cut of {B,A-1}/{C,A-2,T} of cut value 6, displaying both of these in the bottom centre of the screen. However, this rejects the cut {B,A-1,A-2,C}/{T} as the sinks and source are not on separate sides of the cut, invalidating it and displaying an error message in the bottom right hand corner * Continuation with graph from test 54 – This shows cuts in a graph which has both a supersource and supersink. This accepts 2 cuts, a cut {S,A,B,C}/{D,E,T} and cut {S,A,B}/{C,D,E,T} both of cut value 6, displaying both of these in the bottom centre of the screen. However, this rejects the cuts {S}/{A,B,C,D,E,T}, {S,A,B,C,D,E}/{T}, {S,A,C,D,T}/{B,E}, {S,B,C,E,T}/{A,D} and {S,A,B,D,E,T}/{C} as these do not split the sources and sinks onto separate sides of the graph, displaying an error message in the bottom right hand corner |
|  | TestVideo7 from 45 seconds to 2 minute 28 seconds | This shows 3 valid cuts on the initial graph, showing that the temporary minimum capacities have no effect on the cut value if going from sink side to source side. Along with other valid cuts of {A}/{B,C,D,E,T} and {A,B,C}/{D,E,T} of cut values 4 and 7 respectively, the cut {A,B}/{C,D,E,T} of cut value 7 is valid. Despite this having edge CB going from the sink side to the source side with a temporary minimum capacity of 1, 0 flow is removed due to minimum capacity being 0. Once the edge CB has been updated to have a minimum and maximum capacity of 1, the cut {A,B}/{C,D,E,T} will have a cut value of 6 as the minimum capacity from the edge CB is removed from the cut value. This is all displayed in the bottom centre of the screen in a table format. The invalid cut {A,B,C,D,E}/{T} is rejected as this does not separate the sinks and sources, with the other 2 cuts being rejected for colliding with the node B, displaying an error message in the bottom right hand corner of the screen of “Cut drawn is invalid” |
|  | testsContinuations from 3 minute 50 seconds to 4 minute 18 seconds | This shows once a minimum cut and maximum flow value is found, a message stating the optimal flow is displayed, in this case a flow of 3 is displayed in the top left hand corner |
|  | testsContinuations from 4 minute 18 seconds to 4 minute 34 seconds | This shows both a cut of {A-1}/{A-2} of value 1 being valid as well as the maximum flow being 1, giving an optimal flow of 1 through the graph, displaying the cut and flow in table format |
|  | testsContinuations from 4 minute 34 seconds to 6 minute 32 seconds | Despite not having a valid supersource or supersink position this finds both a flow of 6 through the graph on paths “A,C,D” and “B,C,E” with a minimum flow of 1 and maximum flow of 2 down each path, which is displayed in the bottom left hand corner, giving a total flow of 6. 2 valid cuts are also shown, ({A,B}/{C,D,E} and {A,B,C}/{D,E}), both with a cut value of 6, giving an optimal flow of 6. This also shows invalid cuts {A,B,C,D}/{C}, {A,B,D}/{C,E} due to the sources and sinks not being on separate sides, displaying an error message in the bottom right hand corner of the screen |
|  | testsContinuations from 6 minute 32 seconds to 7 minute 22 seconds | This shows a graph where both “A” and “B” are node capacities of value 2 and 3 respectively, meaning this accommodated for all movement of nodes in the graph. This finds a flow of 2 with a path “A-1,A-2,B-1,B-2”, as displayed in the bottom left hand corner in table format, with the flow being updated on all edges. There is 3 valid cuts that are accepted, ({A-1}/{A-2,B-1,B-2}, {A-1,A-2}/{B-1,B-2} and {A-1,A-2,B-1}/{B-2}) with the first 2 having a cut value of 2 and the last having a cut value of 3, displaying this in the bottom centre of the screen. This gives an optimal flow of 2 |
|  | testsContinuations from 7 minute 22 seconds to 8 minute 12 seconds | This shows a graph with a node capacity that is neither a source or a sink which has a valid position as seen by “B” being split with a node capacity of 2. This shows a maximum flow of 2, limited by the node capacity through the path “A,B,C” as seen in the table in the bottom left hand corner. 3 cuts are also shown to be valid, ({A}/{B-1,B-2,C}, {A,B-1}/{B-2,C} and {A,B-1,B-2}/{C}), with the first and third cut having a cut value of 3 due to the intersection with either AB-1 or B-2C, and the middle cut having a cut value of 2, giving an optimal flow of 2. These cuts are displayed in a table format in the bottom centre of the screen |
|  | testsContinuations-part2 from 0 seconds to 1 minute 56 seconds | This shows a graph in which the path “A,B,E,F” of flow 2 limits the maximum flow, reducing it by 2, meaning the optimal flow is reduced due to AB and EF edges having no more forward flow. This means once the solve button is pressed, as the next button is pressed, this first removes the augmented path added before paths “A,B,D,F” and “A,C,E,F” were added with a flow of 2, resulting in an optimal flow of 4, as displayed in the bottom left hand corner of the screen |

1. Corrections and errors found:

* Test 47 – Removing the error message when a single node with a node capacity is entered into the graph. Corrected by adding an if statement to check if the single node met the requirements of being a node capacity. This is corrected as shown in testsContinuations from 1 minute 22 seconds to 1 minute 43 seconds with no error message being displayed in the bottom right hand corner
* Test 64/65 – The cut ID button for showing and hiding was not resetting properly meaning the phase of the button didn’t match if cuts were hidden and the user goes from solve phase to edit phase to solve phase and re-adds valid cuts. Corrected by resetting the array when entering the solve phase. This is shown to be fixed in TestVideo7 from 0 seconds to 40 seconds with the text matching with the correct hidden or show state of the cut
* Correcting the minimum capacity of supersources and supersinks as these must always be 0. Corrected by setting the node argument to 0 for a supersource and supersink, despite ingoing or outgoing minimum capacities. This is corrected as shown in testsContinuations-part2 from 1 minute 55 seconds to 3 minute 1 second, with the minimum capacity being shown as 0, with the minimum flows raising the temporary minimum capacity of edges SA, SB, DT and ET.
* Although the correction tests do not necessarily build the same graphs, the show the correction on a suitable alternative, rather than redoing multiple tests for the same error.

1. Additional tests added:

Additional test 1: This test checks to see if the graph is connected with there only being a single valid graph being accepted. This adds 2 independent graphs to the screen, one of nodes “1” and “3” at positions (1,1) and (3,3) respectively with an edge between them of maximum capacity of 3 and a similar graph with nodes “2” and “4” at positions (5,1) and (7,3) respectively. The solve button is then pressed. This passed, expecting the graph to be rejected as this is not connected, displaying a suitable error message. This is seen in AdditionalTests1 from 0 seconds to 35 seconds where an error message is displayed of “Graph not connected” in the bottom right hand corner once the solve button has been pressed.

Additional test 2: This shows the ability of the program to cope with node capacities that have no valid position and the node capacity is at the start or end of the path. This adds nodes 1, 2, 3, 4, 5, 6, 7, 8, 9 with node 9 having a node capacity of 3, with positions of each node being (1,1), (3,1), (5,1), (5,3), (5,5), (3,5), (1,5), (1,3) and (3,3) respectively. This has edges of maximum capacity 2 for the edges 12, 23, 34, 45, 56, 67, 78, 89. The solve button is then pressed, followed by the next button and 2 cuts are drawn, one intersecting node 9 and the other intersecting node 8. This passes, with the result expected to be the first cut being accepted and the second being rejected. This is seen in AdditonalTests1 from 35 seconds to 2 minute 7 seconds where an error message is displayed in the bottom right hand corner of “No valid node capacity position” once the solve button has been pressed. The maximum flow is 2 down “1,2,3,4,5,6,7,8,9”. The cut {1,2,3,4,5,6,7,8,9-1}/{9,2} of cut value 3 is accepted despite colliding with the node. This is allowed as no node capacity position is found meaning an optimum cut can still be accessed if a part of the optimal flow, meaning this can account for no valid positions. This is represented by the cut notation splitting the node, despite it not being on the screen to shown where the cut value is coming from. Due to the node 8 not having a node capacity, the collision is accepted, making the cut invalid.

Additional test 3: This shows the ability of the program to cope with node capacities that have no valid position and the node capacity is in the middle of the path. This follows a similar format to additional test 2 with nodes 1, 2, 3, 4, 5, 6, 7, 8, 9 and node 5 having the node capacity of 3, with positions of each node being (1,1), (3,1), (5,1), (5,3), (3,3), (1,3), (1,5), (3,5), and (5,5). The edges go to and from the same node in the additional test 2 with the same maximum capacity. The solve button is then pressed, followed by the next button and 2 cuts are drawn, one intersecting node 5 and the other intersecting node 6. This passes, with the result expected to be the first cut being accepted and the second being rejected. This is seen in AdditonialTests2 from 0 seconds to 1 minute 2 seconds where an error message is displayed in the bottom right hand corner of “No valid node capacity position” once the solve button has been pressed. The cut {1,2,3,4,5-1}/{5-2,6,7,8,9} of cut value 3 is accepted despite colliding with the node. This is allowed as no node capacity position is found meaning an optimum cut can still be accessed if a part of the optimal flow, accounting for if the node capacity is in the middle of path, meaning this can account for no valid positions. This is represented by the cut notation splitting the node, despite it not being on the screen to shown where the cut value is coming from. Due to the node 6 not having a node capacity, the collision is accepted, making the cut invalid.