BFS Algorithm

Causes		Value	1	2	3	4
C1	Source node valid	Y, N	N	N	Y	Y
C2	Destination node valid	Y, N	N	Y	N	Y
Effects						
E1	Shortest path					X
E2	Not shortest path		X	X	X	

Reduced table for BFS algorithm

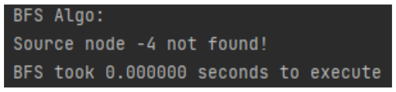
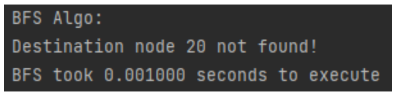
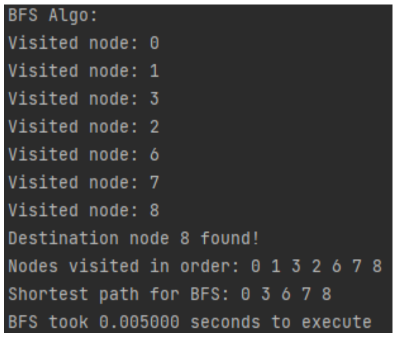
Causes		Value	1-2	3	4
C1	Source node valid	Y, N	N	Y	Y
C2	Destination node valid	Y, N	-	N	Y
Effects					
E1	Shortest path				X
E2	Not shortest path		X	X	

Test scenario for BFS algorithm

Test scenario	Test case	Pre-condition	Test steps	Test data	Expected result	Actual result	Pass/Fail
Check functionality of BFS algorithm (C1: Source node not valid - E2: Not shortest path)	Check whether shortest path is obtained when source node is invalid.		1. Run the codes 2. Check if the output is as expected under BFS algorithm	Source node: -4 Destination node: 6	“Source node -4 not found” message should be printed	“Source node -4 not found” message is printed	Pass

Check functionality of BFS algorithm (C1: Source node valid, C2: Destination node invalid - E2: Not shortest path)	Check whether shortest path is obtained when source node is valid but destination node is invalid.		1. Run the codes 2. Check if the output is as expected under BFS algorithm	Source node: 0 Destination node: 20	“Destination node 20 not found” message should be printed	“Destination node 20 not found” message is printed	Pass
Check functionality of BFS algorithm (C1: Source node valid, C2: Destination node valid - E1: Shortest path)	Check whether shortest path is obtained when both source and destination nodes are valid.		1. Run the codes 2. Check if the output is as expected under BFS algorithm	Source node: 0 Destination node: 8	“Shortest path for BFS: 0 3 6 7 8” should be printed	“Shortest path for BFS: 0 3 6 7 8” is printed	Pass

Testing results for BFS

Source node invalid	Destination node invalid	Source and destination nodes valid
 BFS Algo: Source node -4 not found! BFS took 0.000000 seconds to execute	 BFS Algo: Destination node 20 not found! BFS took 0.001000 seconds to execute	 BFS Algo: Visited node: 0 Visited node: 1 Visited node: 3 Visited node: 2 Visited node: 6 Visited node: 7 Visited node: 8 Destination node 8 found! Nodes visited in order: 0 1 3 2 6 7 8 Shortest path for BFS: 0 3 6 7 8 BFS took 0.005000 seconds to execute

DFS Algo

Causes		Value	1	2	3	4	5	6	7	8
C1	All edges have been added	Y, N	Y	Y	Y	Y	N	N	N	N
C2	Source node valid	Y, N	Y	Y	N	N	Y	Y	N	N
C3	Destination node valid	Y, N	Y	N	Y	N	Y	N	Y	N
Effects										
E1	Path Found		X							
E2	Path not found			X	X	X	X	X	X	X

Reduced table for DFS

Causes		Value	1	2-4	5-8
C1	All edges have been added	Y, N	Y	Y	N
C2	Source node valid	Y, N	Y	Y	Y
C3	Destination node valid	Y, N	Y	N	Y
Effects					
E1	Path Found		X		
E2	Path not found			X	X

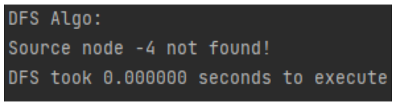
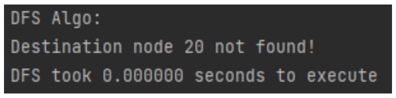
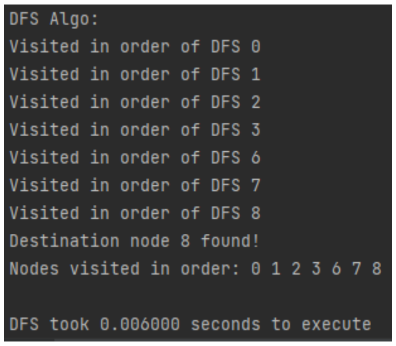
Test scenario for DFS algorithm

Test scenario	Test case	Pre-condition	Test steps	Test data	Expected result	Actual result	Pass/Fail
Check functionality of DFS algorithm (C1: All edges have been added C2: Source node not valid - E2: Path not found)	Check whether path is obtained when source node is invalid.	graph { 0 -- 1; 0 -- 3; 1 -- 0; 1 -- 2; 2 -- 1; 3 -- 0; 3 -- 6; 4 -- 5; 5 -- 4; 5 -- 8; 6 -- 3; 6 -- 7; 7 -- 6; 7 -- 8; 8 -- 5; 8 -- 7; }	1. Run the codes 2. Check if the output is as expected under DFS algorithm	Source node: -4 Destination node: 6	“Source node -4 not found” message should be printed	“Source node -4 not found” message is printed	Pass
Check functionality of DFS algorithm (C1: All edges have been added C3: Destination node not valid - E2: Path not found)	Check whether path is obtained when source node is valid but destination node is invalid.	graph { 0 -- 1; 0 -- 3; 1 -- 0; 1 -- 2; 2 -- 1; 3 -- 0; 3 -- 6; 4 -- 5; 5 -- 4; 5 -- 8; 6 -- 3; 6 -- 7; 7 -- 6; 7 -- 8; 8 -- 5; 8 -- 7; }	1. Run the codes 2. Check if the output is as expected under DFS algorithm	Source node: 0 Destination node: 20	“Destination node 20 not found” message should be printed	“Destination node 20 not found” message is printed	Pass
Check functionality of DFS algorithm (C2: Source node valid,	Check whether path is obtained when both	graph { 0 -- 1; 0 -- 3; 1 -- 0; 1 -- 2; 2 -- 1;	1. Run the codes 2. Check if the output is	Source node: 0 Destination node: 8	“Nodes visited in order: 0 1 2 3 6 7 8” should be	“Nodes visited in order: 0 1 2 3 6 7 8” is printed	Pass

C3: Destination node valid - E1: path is found)	source and destinati on nodes are valid.	3 -- 0;
		3 -- 6;
		4 -- 5;
		5 -- 4;
		5 -- 8;
		6 -- 3;
		6 -- 7;
		7 -- 6;
		7 -- 8;
		8 -- 5;
		8 -- 7;
		}
			as expected under DFS algorithm		printed		

Testing results for DFS

Source node invalid	Destination node invalid	Source and destination nodes valid


`DFS Algo:`
`Source node -4 not found!`
`DFS took 0.000000 seconds to execute`
	
	`DFS Algo:`
	`Destination node 20 not found!`
	`DFS took 0.000000 seconds to execute`
		
		`DFS Algo:`
		`Visited in order of DFS 0`
		`Visited in order of DFS 1`
		`Visited in order of DFS 2`
		`Visited in order of DFS 3`
		`Visited in order of DFS 6`
		`Visited in order of DFS 7`
		`Visited in order of DFS 8`
		`Destination node 8 found!`
		`Nodes visited in order: 0 1 2 3 6 7 8`
		
		`DFS took 0.006000 seconds to execute`