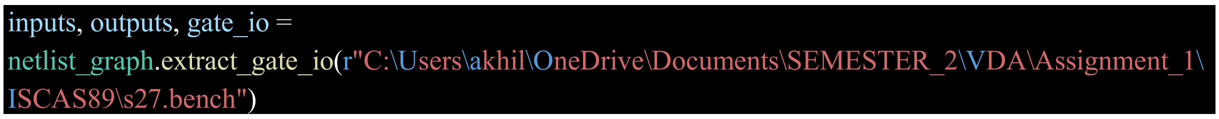
1. Introduction:

A python code is written to extract the gates and information about the interconnects. The gates are considered as nodes and interconnects as edges. Then transversal algorithms called Depth First Search and Breadth First Search

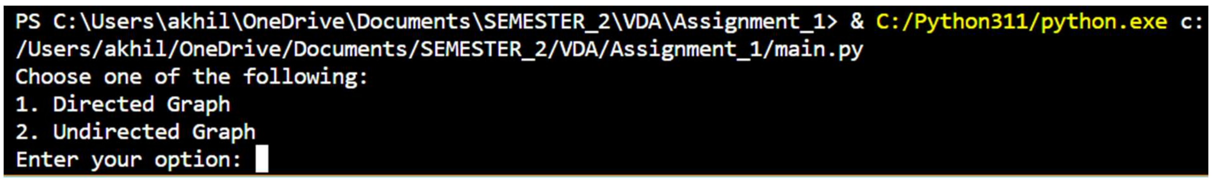
2. Inputs:

The file path of a benchmark file which is a netlist file is given as input. After that an option for directed and undirected graph is asked. Option should be entered by the user and accordingly the graph is plotted.

Eg.



```
inputs, outputs, gate_io =
netlist_graph.extract_gate_io(r"C:\Users\akhil\OneDrive\Documents\SEMESTER_2\VDA\Assignment_1\
ISCAS89\s27.bench")
```



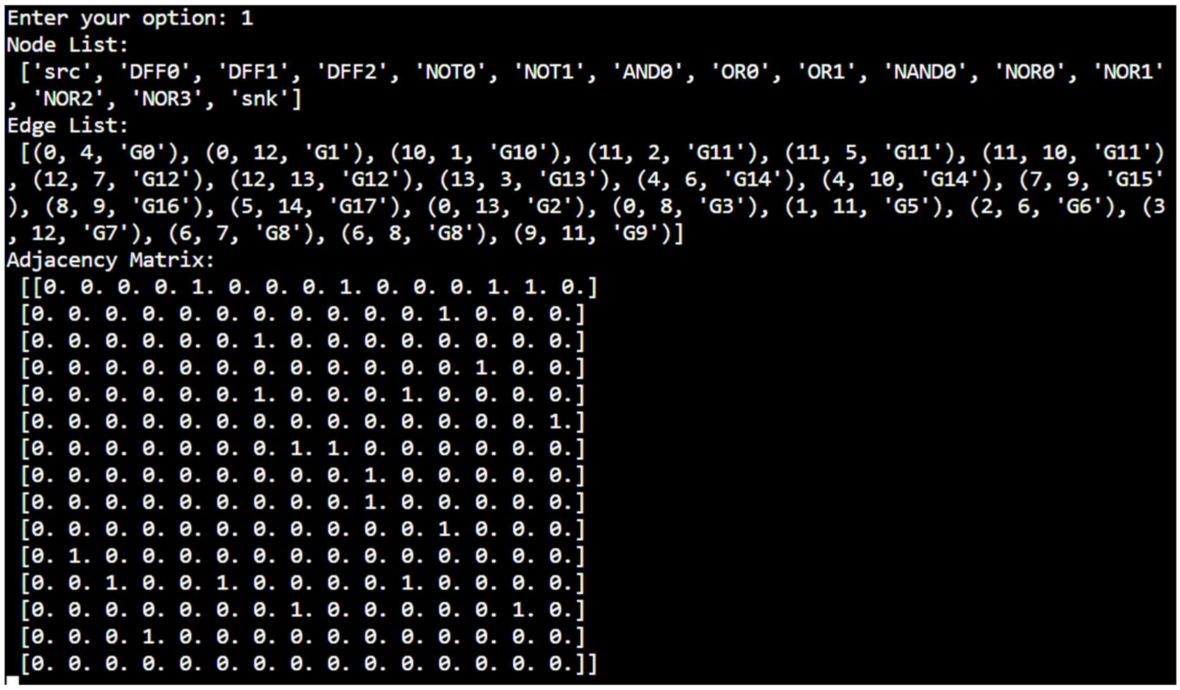
```
PS C:\Users\akhil\OneDrive\Documents\SEMESTER_2\VDA\Assignment_1> & C:/Python311/python.exe c:
/Users/akhil/OneDrive/Documents/SEMESTER_2/VDA/Assignment_1/main.py
Choose one of the following:
1. Directed Graph
2. Undirected Graph
Enter your option: 
```

3. Outputs:

Once the required option is chosen, the list of nodes, edges and the adjacency matrix is displayed. In the edge list, the nodes are displayed as numbers which corresponds to the index value in node list.

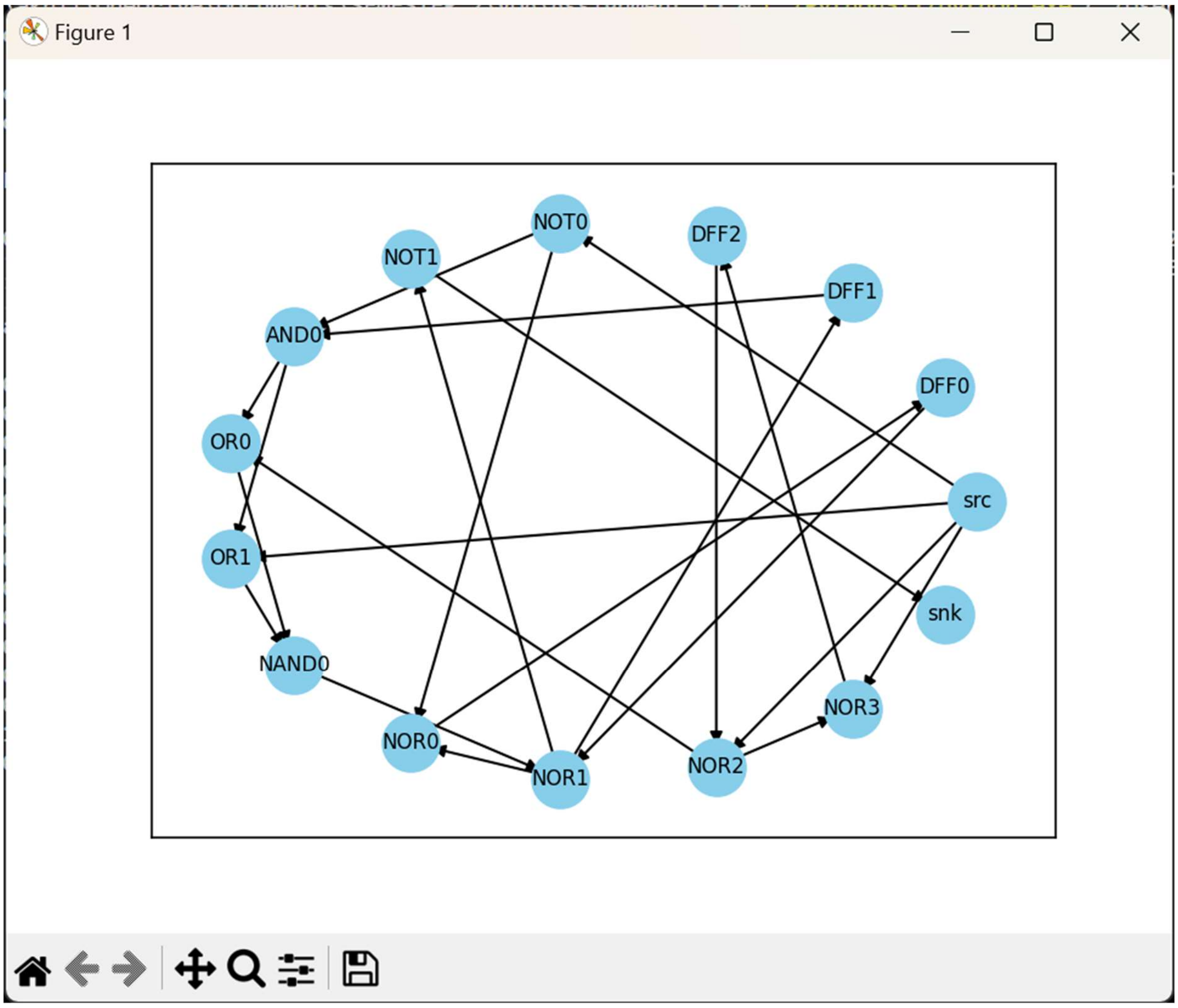
Below is the output when chosen option is Directed Graph.

Node list, Edge list and Adjacency matrix



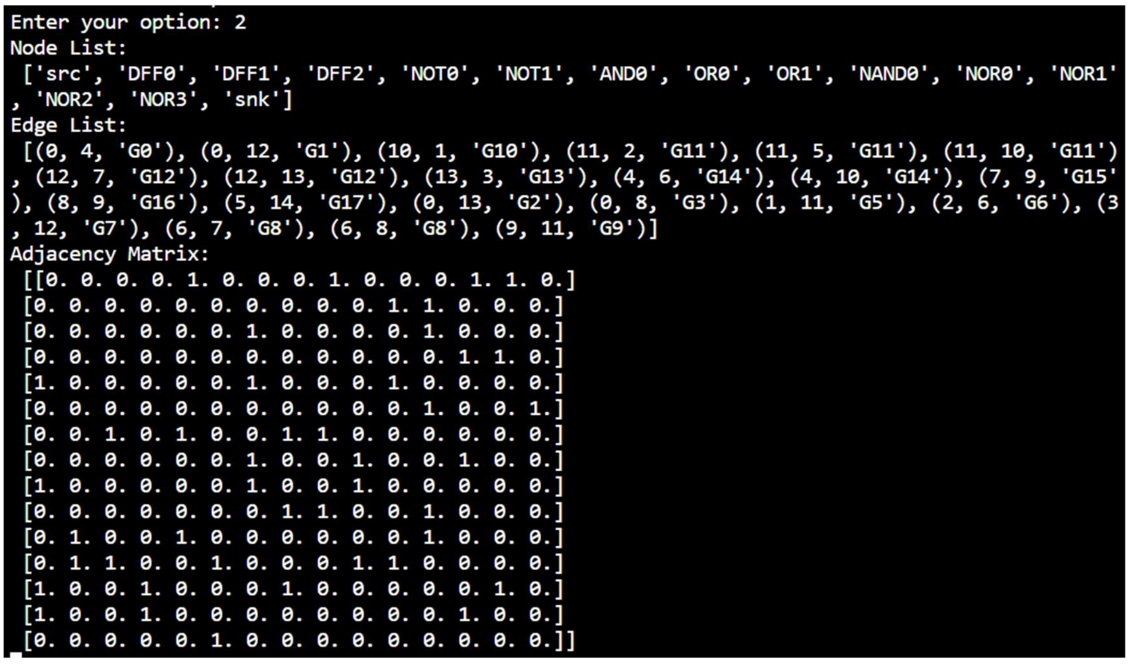
```
Enter your option: 1
Node List:
 ['src', 'DFF0', 'DFF1', 'DFF2', 'NOT0', 'NOT1', 'AND0', 'OR0', 'OR1', 'NAND0', 'NOR0', 'NOR1'
, 'NOR2', 'NOR3', 'snk']
Edge List:
 [(0, 4, 'G0'), (0, 12, 'G1'), (10, 1, 'G10'), (11, 2, 'G11'), (11, 5, 'G11'), (11, 10, 'G11')
, (12, 7, 'G12'), (12, 13, 'G12'), (13, 3, 'G13'), (4, 6, 'G14'), (4, 10, 'G14'), (7, 9, 'G15'
), (8, 9, 'G16'), (5, 14, 'G17'), (0, 13, 'G2'), (0, 8, 'G3'), (1, 11, 'G5'), (2, 6, 'G6'), (3
, 12, 'G7'), (6, 7, 'G8'), (6, 8, 'G8'), (9, 11, 'G9')]
Adjacency Matrix:
 [[0. 0. 0. 0. 1. 0. 0. 0. 1. 0. 0. 0. 1. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
 [0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 0. 0. 1. 1. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 1. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
```

Directed Graph:



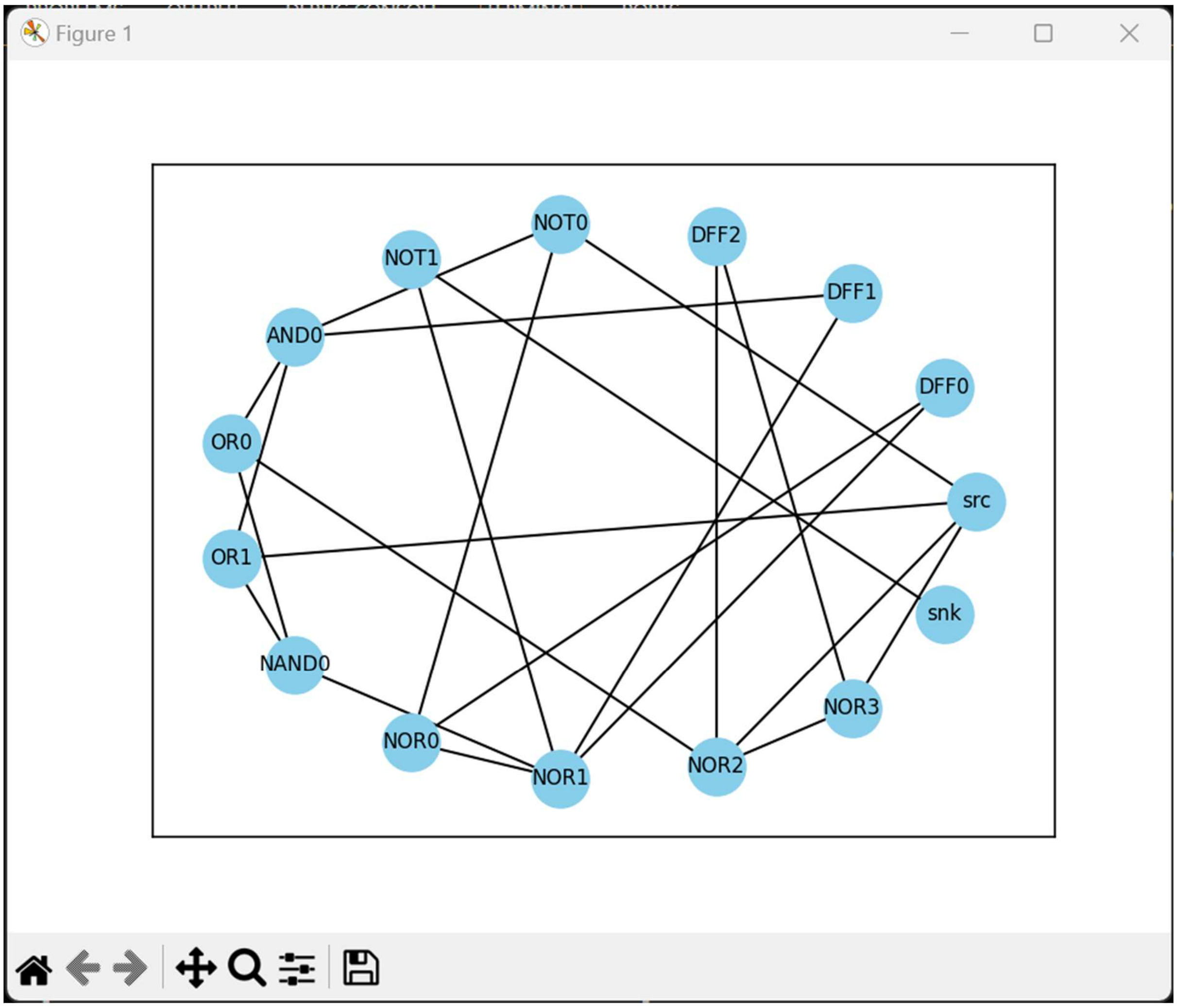
Similarly, below is the output when chosen option is Directed Graph.

Node list, Edge list and Adjacency matrix:



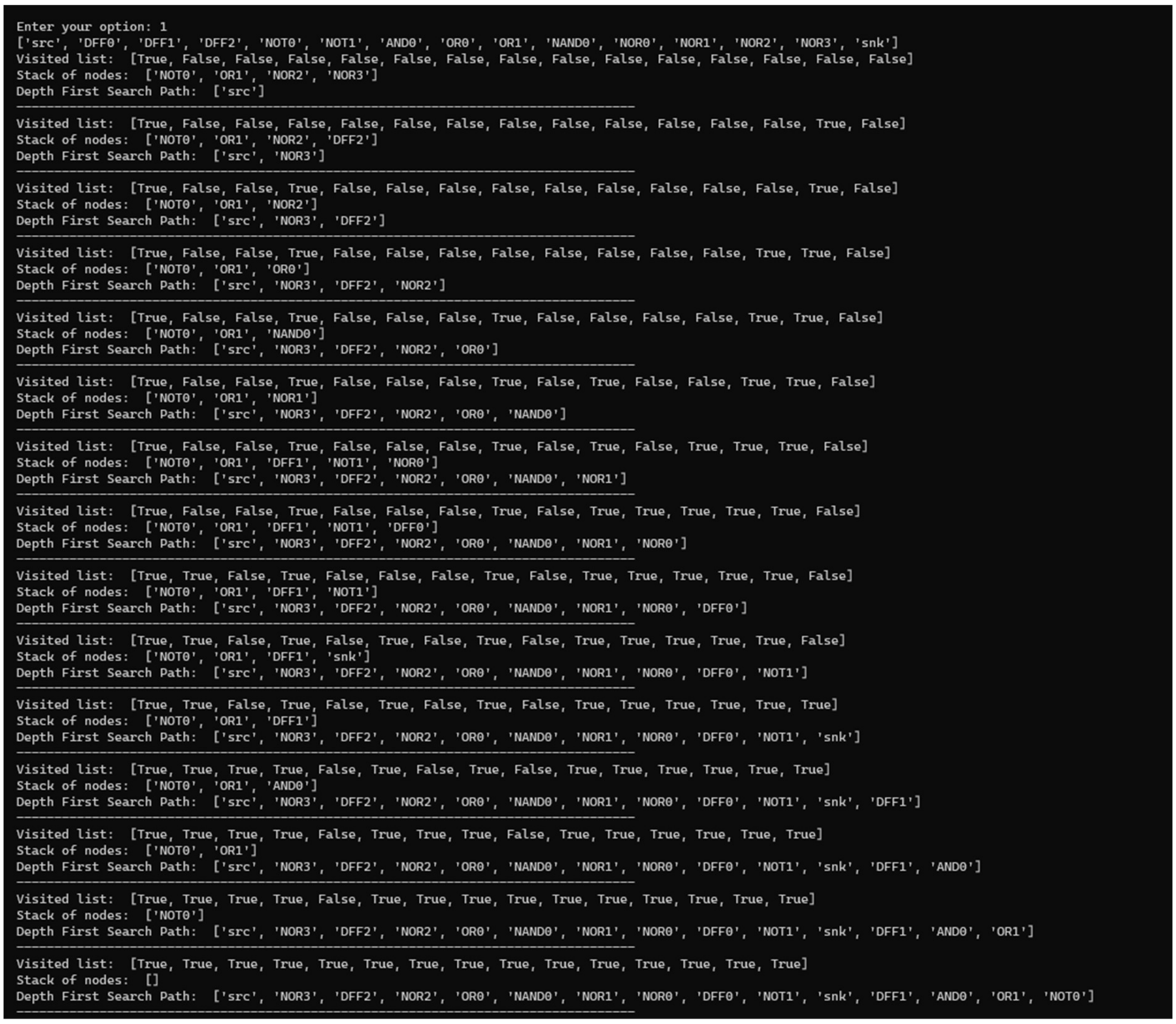
```
Enter your option: 2
Node List:
 ['src', 'DFF0', 'DFF1', 'DFF2', 'NOT0', 'NOT1', 'AND0', 'OR0', 'OR1', 'NAND0', 'NOR0', 'NOR1'
, 'NOR2', 'NOR3', 'snk']
Edge List:
 [(0, 4, 'G0'), (0, 12, 'G1'), (10, 1, 'G10'), (11, 2, 'G11'), (11, 5, 'G11'), (11, 10, 'G11')
, (12, 7, 'G12'), (12, 13, 'G12'), (13, 3, 'G13'), (4, 6, 'G14'), (4, 10, 'G14'), (7, 9, 'G15'
), (8, 9, 'G16'), (5, 14, 'G17'), (0, 13, 'G2'), (0, 8, 'G3'), (1, 11, 'G5'), (2, 6, 'G6'), (3
, 12, 'G7'), (6, 7, 'G8'), (6, 8, 'G8'), (9, 11, 'G9')]
Adjacency Matrix:
 [[0. 0. 0. 0. 1. 0. 0. 0. 1. 0. 0. 0. 1. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 0.]
 [1. 0. 0. 0. 0. 0. 1. 0. 0. 0. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 1.]
 [0. 0. 1. 0. 1. 0. 0. 1. 1. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 1. 0. 0. 1. 0. 0. 1. 0. 0.]
 [1. 0. 0. 0. 0. 0. 1. 0. 0. 1. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 1. 1. 0. 0. 1. 0. 0. 0.]
 [0. 1. 0. 0. 1. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]
 [0. 1. 1. 0. 0. 1. 0. 0. 0. 1. 1. 0. 0. 0. 0.]
 [1. 0. 0. 1. 0. 0. 0. 1. 0. 0. 0. 0. 0. 1. 0.]
 [1. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
 [0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
```

Undirected Graph:



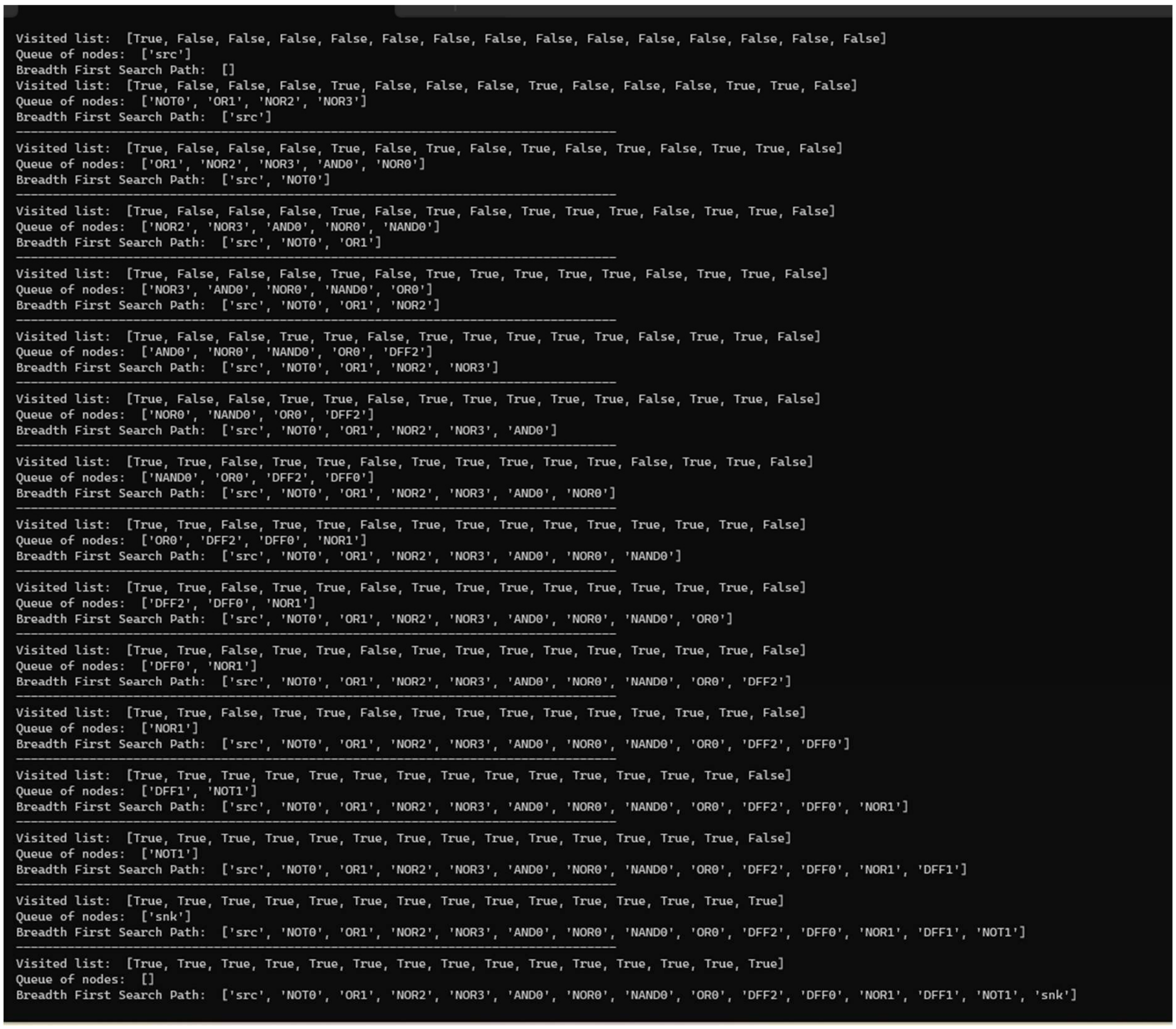
After the corresponding graph is plotted, DFS and BFS algorithms are started and the corresponding path is printed.

For Depth first search algorithm, visited node list, stack used for search and the DFS path are printed for iteration.



```
Enter your option: 1
['src', 'DFF0', 'DFF1', 'DFF2', 'NOT0', 'NOT1', 'AND0', 'OR0', 'OR1', 'NAND0', 'NOR0', 'NOR1', 'NOR2', 'NOR3', 'snk']
Visited list:  [True, False, False, False, False, False, False, False, False, False, False, False, False, False, False]
Stack of nodes:  ['NOT0', 'OR1', 'NOR2', 'NOR3']
Depth First Search Path:  ['src']
----------------------------------------------------------------------------------
Visited list:  [True, False, False, False, False, False, False, False, False, False, False, False, False, True, False]
Stack of nodes:  ['NOT0', 'OR1', 'NOR2', 'DFF2']
Depth First Search Path:  ['src', 'NOR3']
----------------------------------------------------------------------------------
Visited list:  [True, False, False, True, False, False, False, False, False, False, False, False, False, True, False]
Stack of nodes:  ['NOT0', 'OR1', 'NOR2']
Depth First Search Path:  ['src', 'NOR3', 'DFF2']
----------------------------------------------------------------------------------
Visited list:  [True, False, False, True, False, False, False, False, False, False, False, False, True, True, False]
Stack of nodes:  ['NOT0', 'OR1', 'OR0']
Depth First Search Path:  ['src', 'NOR3', 'DFF2', 'NOR2']
----------------------------------------------------------------------------------
Visited list:  [True, False, False, True, False, False, False, True, False, False, False, False, True, True, False]
Stack of nodes:  ['NOT0', 'OR1', 'NAND0']
Depth First Search Path:  ['src', 'NOR3', 'DFF2', 'NOR2', 'OR0']
----------------------------------------------------------------------------------
Visited list:  [True, False, False, True, False, False, False, True, False, True, False, False, True, True, False]
Stack of nodes:  ['NOT0', 'OR1', 'NOR1']
Depth First Search Path:  ['src', 'NOR3', 'DFF2', 'NOR2', 'OR0', 'NAND0']
----------------------------------------------------------------------------------
Visited list:  [True, False, False, True, False, False, False, True, False, True, False, True, True, True, False]
Stack of nodes:  ['NOT0', 'OR1', 'DFF1', 'NOT1', 'NOR0']
Depth First Search Path:  ['src', 'NOR3', 'DFF2', 'NOR2', 'OR0', 'NAND0', 'NOR1']
----------------------------------------------------------------------------------
Visited list:  [True, False, False, True, False, False, False, True, False, True, True, True, True, True, False]
Stack of nodes:  ['NOT0', 'OR1', 'DFF1', 'NOT1', 'DFF0']
Depth First Search Path:  ['src', 'NOR3', 'DFF2', 'NOR2', 'OR0', 'NAND0', 'NOR1', 'NOR0']
----------------------------------------------------------------------------------
Visited list:  [True, True, False, True, False, False, False, True, False, True, True, True, True, True, False]
Stack of nodes:  ['NOT0', 'OR1', 'DFF1', 'NOT1']
Depth First Search Path:  ['src', 'NOR3', 'DFF2', 'NOR2', 'OR0', 'NAND0', 'NOR1', 'NOR0', 'DFF0']
----------------------------------------------------------------------------------
Visited list:  [True, True, False, True, False, True, False, True, False, True, True, True, True, True, False]
Stack of nodes:  ['NOT0', 'OR1', 'DFF1', 'snk']
Depth First Search Path:  ['src', 'NOR3', 'DFF2', 'NOR2', 'OR0', 'NAND0', 'NOR1', 'NOR0', 'DFF0', 'NOT1']
----------------------------------------------------------------------------------
Visited list:  [True, True, False, True, False, True, False, True, False, True, True, True, True, True, True]
Stack of nodes:  ['NOT0', 'OR1', 'DFF1']
Depth First Search Path:  ['src', 'NOR3', 'DFF2', 'NOR2', 'OR0', 'NAND0', 'NOR1', 'NOR0', 'DFF0', 'NOT1', 'snk']
----------------------------------------------------------------------------------
Visited list:  [True, True, True, True, False, True, False, True, False, True, True, True, True, True, True]
Stack of nodes:  ['NOT0', 'OR1', 'AND0']
Depth First Search Path:  ['src', 'NOR3', 'DFF2', 'NOR2', 'OR0', 'NAND0', 'NOR1', 'NOR0', 'DFF0', 'NOT1', 'snk', 'DFF1']
----------------------------------------------------------------------------------
Visited list:  [True, True, True, True, False, True, True, True, False, True, True, True, True, True, True]
Stack of nodes:  ['NOT0', 'OR1']
Depth First Search Path:  ['src', 'NOR3', 'DFF2', 'NOR2', 'OR0', 'NAND0', 'NOR1', 'NOR0', 'DFF0', 'NOT1', 'snk', 'DFF1', 'AND0']
----------------------------------------------------------------------------------
Visited list:  [True, True, True, True, False, True, True, True, True, True, True, True, True, True, True]
Stack of nodes:  ['NOT0']
Depth First Search Path:  ['src', 'NOR3', 'DFF2', 'NOR2', 'OR0', 'NAND0', 'NOR1', 'NOR0', 'DFF0', 'NOT1', 'snk', 'DFF1', 'AND0', 'OR1']
----------------------------------------------------------------------------------
Visited list:  [True, True, True, True, True, True, True, True, True, True, True, True, True, True, True]
Stack of nodes:  []
Depth First Search Path:  ['src', 'NOR3', 'DFF2', 'NOR2', 'OR0', 'NAND0', 'NOR1', 'NOR0', 'DFF0', 'NOT1', 'snk', 'DFF1', 'AND0', 'OR1', 'NOT0']
----------------------------------------------------------------------------------
```

For Breadth First Search algorithm, visited node list, Queue used for search and the BFS path for each iteration are printed.



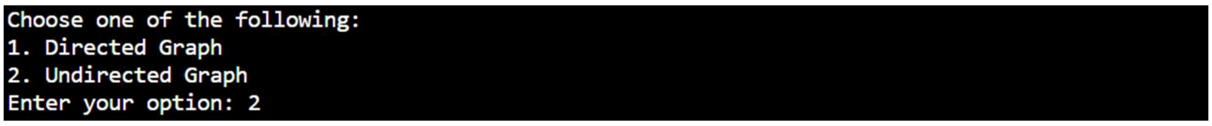
```
Visited list:  [True, False, False, False, False, False, False, False, False, False, False, False, False, False, False]
Queue of nodes:  ['src']
Breadth First Search Path:  []
Visited list:  [True, False, False, False, True, False, False, False, True, False, False, False, True, True, False]
Queue of nodes:  ['NOT0', 'OR1', 'NOR2', 'NOR3']
Breadth First Search Path:  ['src']
--------------------------------------------------------------------------------
Visited list:  [True, False, False, False, True, False, True, False, True, False, True, False, True, True, False]
Queue of nodes:  ['OR1', 'NOR2', 'NOR3', 'AND0', 'NOR0']
Breadth First Search Path:  ['src', 'NOT0']
--------------------------------------------------------------------------------
Visited list:  [True, False, False, False, True, False, True, False, True, True, True, False, True, True, False]
Queue of nodes:  ['NOR2', 'NOR3', 'AND0', 'NOR0', 'NAND0']
Breadth First Search Path:  ['src', 'NOT0', 'OR1']
--------------------------------------------------------------------------------
Visited list:  [True, False, False, False, True, False, True, True, True, True, True, False, True, True, False]
Queue of nodes:  ['NOR3', 'AND0', 'NOR0', 'NAND0', 'OR0']
Breadth First Search Path:  ['src', 'NOT0', 'OR1', 'NOR2']
--------------------------------------------------------------------------------
Visited list:  [True, False, False, True, True, False, True, True, True, True, True, False, True, True, False]
Queue of nodes:  ['AND0', 'NOR0', 'NAND0', 'OR0', 'DFF2']
Breadth First Search Path:  ['src', 'NOT0', 'OR1', 'NOR2', 'NOR3']
--------------------------------------------------------------------------------
Visited list:  [True, False, False, True, True, False, True, True, True, True, True, False, True, True, False]
Queue of nodes:  ['NOR0', 'NAND0', 'OR0', 'DFF2']
Breadth First Search Path:  ['src', 'NOT0', 'OR1', 'NOR2', 'NOR3', 'AND0']
--------------------------------------------------------------------------------
Visited list:  [True, True, False, True, True, False, True, True, True, True, True, False, True, True, False]
Queue of nodes:  ['NAND0', 'OR0', 'DFF2', 'DFF0']
Breadth First Search Path:  ['src', 'NOT0', 'OR1', 'NOR2', 'NOR3', 'AND0', 'NOR0']
--------------------------------------------------------------------------------
Visited list:  [True, True, False, True, True, False, True, True, True, True, True, True, True, True, False]
Queue of nodes:  ['OR0', 'DFF2', 'DFF0', 'NOR1']
Breadth First Search Path:  ['src', 'NOT0', 'OR1', 'NOR2', 'NOR3', 'AND0', 'NOR0', 'NAND0']
--------------------------------------------------------------------------------
Visited list:  [True, True, False, True, True, False, True, True, True, True, True, True, True, True, False]
Queue of nodes:  ['DFF2', 'DFF0', 'NOR1']
Breadth First Search Path:  ['src', 'NOT0', 'OR1', 'NOR2', 'NOR3', 'AND0', 'NOR0', 'NAND0', 'OR0']
--------------------------------------------------------------------------------
Visited list:  [True, True, False, True, True, False, True, True, True, True, True, True, True, True, False]
Queue of nodes:  ['DFF0', 'NOR1']
Breadth First Search Path:  ['src', 'NOT0', 'OR1', 'NOR2', 'NOR3', 'AND0', 'NOR0', 'NAND0', 'OR0', 'DFF2']
--------------------------------------------------------------------------------
Visited list:  [True, True, False, True, True, False, True, True, True, True, True, True, True, True, False]
Queue of nodes:  ['NOR1']
Breadth First Search Path:  ['src', 'NOT0', 'OR1', 'NOR2', 'NOR3', 'AND0', 'NOR0', 'NAND0', 'OR0', 'DFF2', 'DFF0']
--------------------------------------------------------------------------------
Visited list:  [True, True, True, True, True, True, True, True, True, True, True, True, True, True, False]
Queue of nodes:  ['DFF1', 'NOT1']
Breadth First Search Path:  ['src', 'NOT0', 'OR1', 'NOR2', 'NOR3', 'AND0', 'NOR0', 'NAND0', 'OR0', 'DFF2', 'DFF0', 'NOR1']
--------------------------------------------------------------------------------
Visited list:  [True, True, True, True, True, True, True, True, True, True, True, True, True, True, False]
Queue of nodes:  ['NOT1']
Breadth First Search Path:  ['src', 'NOT0', 'OR1', 'NOR2', 'NOR3', 'AND0', 'NOR0', 'NAND0', 'OR0', 'DFF2', 'DFF0', 'NOR1', 'DFF1']
--------------------------------------------------------------------------------
Visited list:  [True, True, True, True, True, True, True, True, True, True, True, True, True, True, True]
Queue of nodes:  ['snk']
Breadth First Search Path:  ['src', 'NOT0', 'OR1', 'NOR2', 'NOR3', 'AND0', 'NOR0', 'NAND0', 'OR0', 'DFF2', 'DFF0', 'NOR1', 'DFF1', 'NOT1']
--------------------------------------------------------------------------------
Visited list:  [True, True, True, True, True, True, True, True, True, True, True, True, True, True, True]
Queue of nodes:  []
Breadth First Search Path:  ['src', 'NOT0', 'OR1', 'NOR2', 'NOR3', 'AND0', 'NOR0', 'NAND0', 'OR0', 'DFF2', 'DFF0', 'NOR1', 'DFF1', 'NOT1', 'snk']
```

4. Other outputs:

Inputs given:

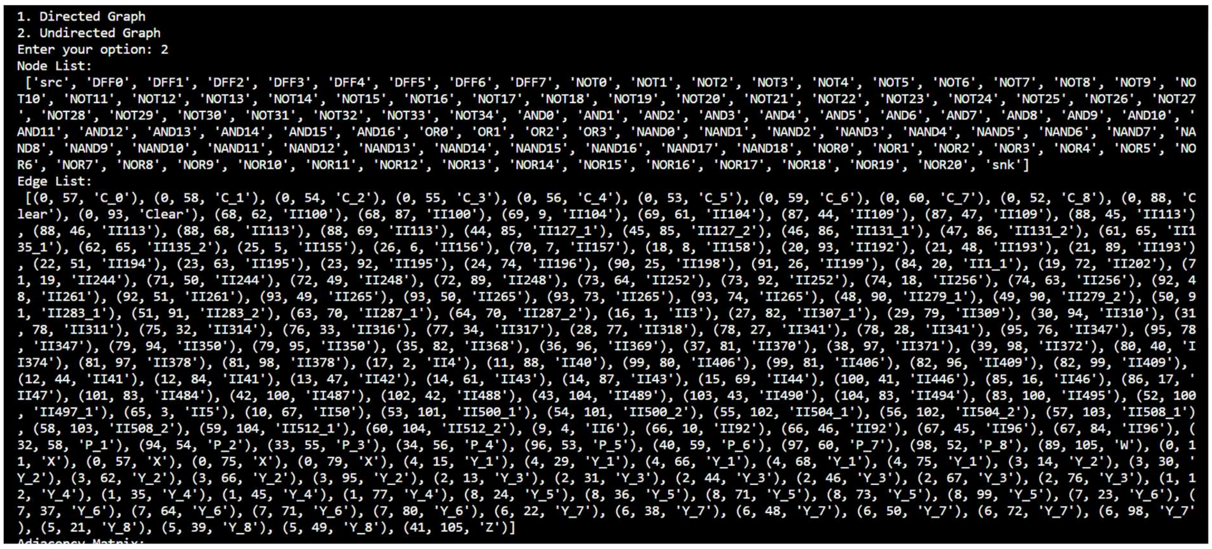
```
inputs, outputs, gate_io =
netlist_graph.extract_gate_io(r"C:\Users\akhil\OneDrive\Documents\SEMESTER_2\VDA\Assignment_1\
ISCAS89\s208.bench")
```

From the options, option 2 – Undirected Graphs was chosen.



```
Choose one of the following:
1. Directed Graph
2. Undirected Graph
Enter your option: 2
```

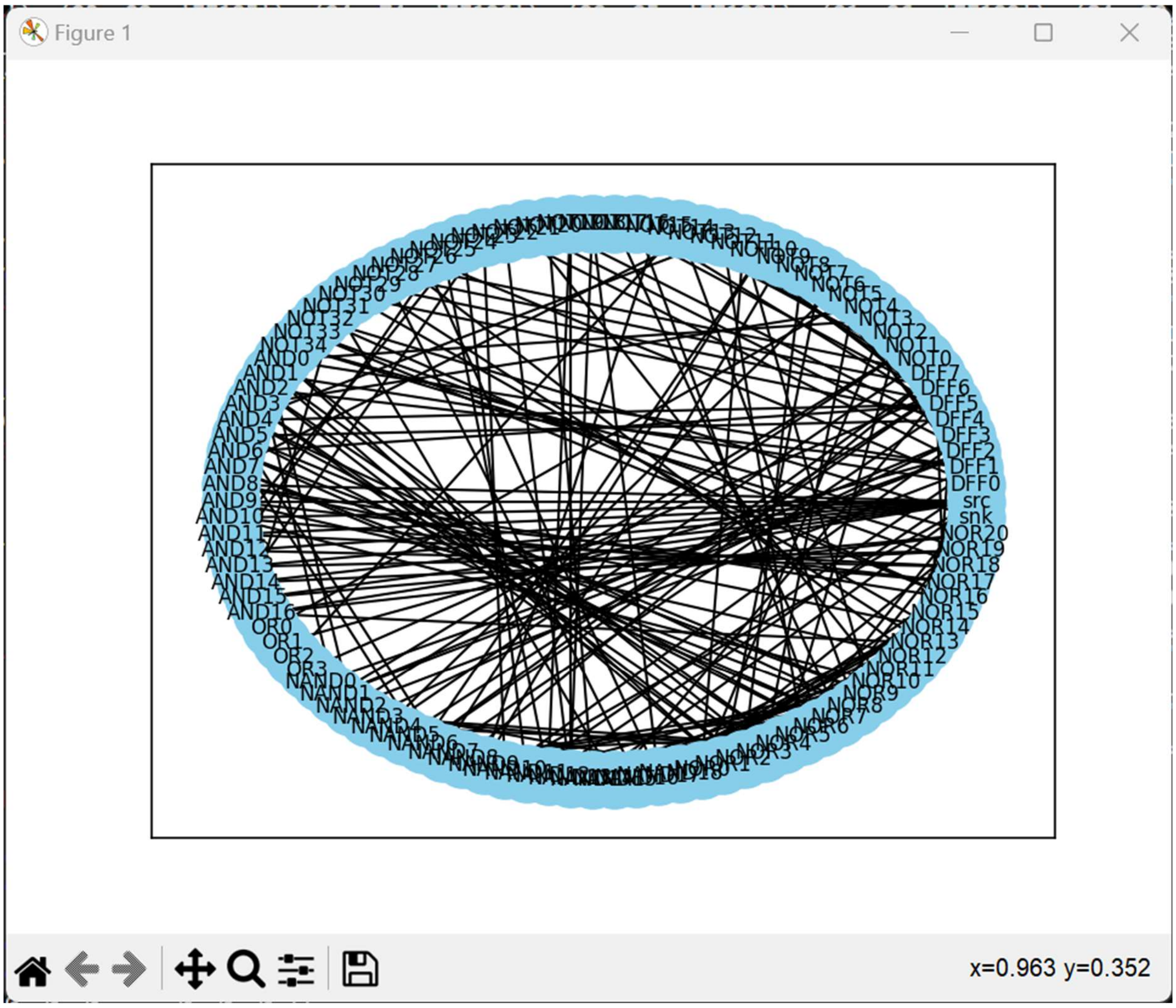
Node list and Edge list:



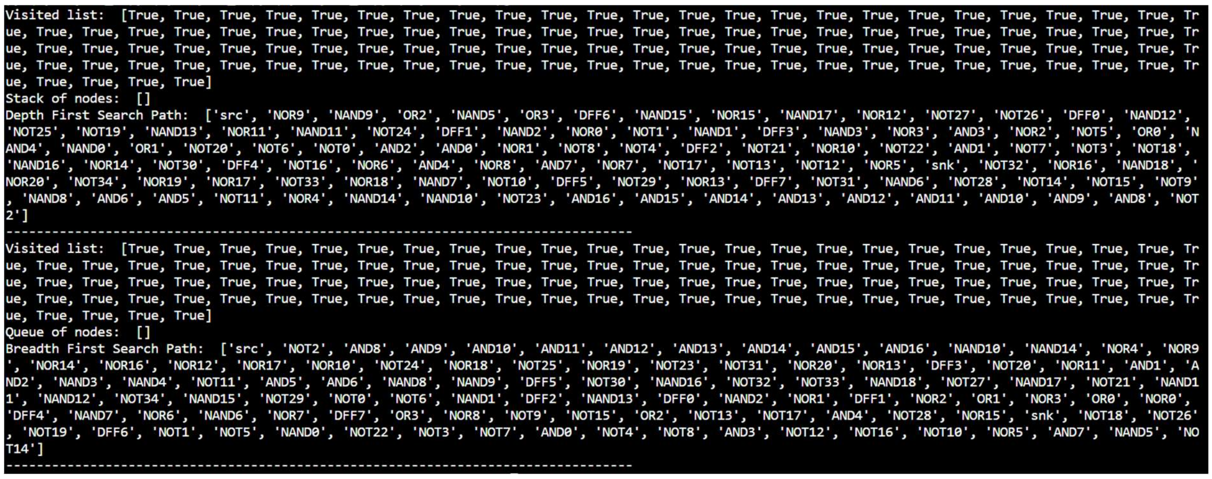
```
1. Directed Graph
2. Undirected Graph
Enter your option: 2
Node List:
 ['src', 'DFF0', 'DFF1', 'DFF2', 'DFF3', 'DFF4', 'DFF5', 'DFF6', 'DFF7', 'NOT0', 'NOT1', 'NOT2', 'NOT3', 'NOT4', 'NOT5', 'NOT6', 'NOT7', 'NOT8', 'NOT9', 'NO
T10', 'NOT11', 'NOT12', 'NOT13', 'NOT14', 'NOT15', 'NOT16', 'NOT17', 'NOT18', 'NOT19', 'NOT20', 'NOT21', 'NOT22', 'NOT23', 'NOT24', 'NOT25', 'NOT26', 'NOT27
', 'NOT28', 'NOT29', 'NOT30', 'NOT31', 'NOT32', 'NOT33', 'NOT34', 'AND0', 'AND1', 'AND2', 'AND3', 'AND4', 'AND5', 'AND6', 'AND7', 'AND8', 'AND9', 'AND10', '
AND11', 'AND12', 'AND13', 'AND14', 'AND15', 'AND16', 'OR0', 'OR1', 'OR2', 'OR3', 'NAND0', 'NAND1', 'NAND2', 'NAND3', 'NAND4', 'NAND5', 'NAND6', 'NAND7', 'NA
ND8', 'NAND9', 'NAND10', 'NAND11', 'NAND12', 'NAND13', 'NAND14', 'NAND15', 'NAND16', 'NAND17', 'NAND18', 'NOR0', 'NOR1', 'NOR2', 'NOR3', 'NOR4', 'NOR5', 'NO
R6', 'NOR7', 'NOR8', 'NOR9', 'NOR10', 'NOR11', 'NOR12', 'NOR13', 'NOR14', 'NOR15', 'NOR16', 'NOR17', 'NOR18', 'NOR19', 'NOR20', 'snk']
Edge List:
 [(0, 57, 'C_0'), (0, 58, 'C_1'), (0, 54, 'C_2'), (0, 55, 'C_3'), (0, 56, 'C_4'), (0, 53, 'C_5'), (0, 59, 'C_6'), (0, 60, 'C_7'), (0, 52, 'C_8'), (0, 88, 'C
lear'), (0, 93, 'Clear'), (68, 62, 'II100'), (68, 87, 'II100'), (69, 9, 'II104'), (69, 61, 'II104'), (87, 44, 'II109'), (87, 47, 'II109'), (88, 45, 'II113')
, (88, 46, 'II113'), (88, 68, 'II113'), (88, 69, 'II113'), (44, 85, 'II127_1'), (45, 85, 'II127_2'), (46, 86, 'II131_1'), (47, 86, 'II131_2'), (61, 65, 'II1
35_1'), (62, 65, 'II135_2'), (25, 5, 'II155'), (26, 6, 'II156'), (70, 7, 'II157'), (18, 8, 'II158'), (20, 93, 'II192'), (21, 48, 'II193'), (21, 89, 'II193')
, (22, 51, 'II194'), (23, 63, 'II195'), (23, 92, 'II195'), (24, 74, 'II196'), (90, 25, 'II198'), (91, 26, 'II199'), (84, 20, 'II1_1'), (19, 72, 'II202'), (7
1, 19, 'II244'), (71, 50, 'II244'), (72, 49, 'II248'), (72, 89, 'II248'), (73, 64, 'II252'), (73, 92, 'II252'), (74, 18, 'II256'), (74, 63, 'II256'), (92, 4
8, 'II261'), (92, 51, 'II261'), (93, 49, 'II265'), (93, 50, 'II265'), (93, 73, 'II265'), (93, 74, 'II265'), (48, 90, 'II279_1'), (49, 90, 'II279_2'), (50, 9
1, 'II283_1'), (51, 91, 'II283_2'), (63, 70, 'II287_1'), (64, 70, 'II287_2'), (16, 1, 'II3'), (27, 82, 'II307_1'), (29, 79, 'II309'), (30, 94, 'II310'), (31
, 78, 'II311'), (75, 32, 'II314'), (76, 33, 'II316'), (77, 34, 'II317'), (28, 77, 'II318'), (78, 27, 'II341'), (78, 28, 'II341'), (95, 76, 'II347'), (95, 78
, 'II347'), (79, 94, 'II350'), (79, 95, 'II350'), (35, 82, 'II368'), (36, 96, 'II369'), (37, 81, 'II370'), (38, 97, 'II371'), (39, 98, 'II372'), (80, 40, 'I
I374'), (81, 97, 'II378'), (81, 98, 'II378'), (17, 2, 'II4'), (11, 88, 'II40'), (99, 80, 'II406'), (99, 81, 'II406'), (82, 96, 'II409'), (82, 99, 'II409'),
(12, 44, 'II41'), (12, 84, 'II41'), (13, 47, 'II42'), (14, 61, 'II43'), (14, 87, 'II43'), (15, 69, 'II44'), (100, 41, 'II446'), (85, 16, 'II46'), (86, 17, '
II47'), (101, 83, 'II484'), (42, 100, 'II487'), (102, 42, 'II488'), (43, 104, 'II489'), (103, 43, 'II490'), (104, 83, 'II494'), (83, 100, 'II495'), (52, 100
, 'II497_1'), (65, 3, 'II5'), (10, 67, 'II50'), (53, 101, 'II500_1'), (54, 101, 'II500_2'), (55, 102, 'II504_1'), (56, 102, 'II504_2'), (57, 103, 'II508_1')
, (58, 103, 'II508_2'), (59, 104, 'II512_1'), (60, 104, 'II512_2'), (9, 4, 'II6'), (66, 10, 'II92'), (66, 46, 'II92'), (67, 45, 'II96'), (67, 84, 'II96'), (
32, 58, 'P_1'), (94, 54, 'P_2'), (33, 55, 'P_3'), (34, 56, 'P_4'), (96, 53, 'P_5'), (40, 59, 'P_6'), (97, 60, 'P_7'), (98, 52, 'P_8'), (89, 105, 'W'), (0, 1
1, 'X'), (0, 57, 'X'), (0, 75, 'X'), (0, 79, 'X'), (4, 15, 'Y_1'), (4, 29, 'Y_1'), (4, 66, 'Y_1'), (4, 68, 'Y_1'), (4, 75, 'Y_1'), (3, 14, 'Y_2'), (3, 30, '
Y_2'), (3, 62, 'Y_2'), (3, 66, 'Y_2'), (3, 95, 'Y_2'), (2, 13, 'Y_3'), (2, 31, 'Y_3'), (2, 44, 'Y_3'), (2, 46, 'Y_3'), (2, 67, 'Y_3'), (2, 76, 'Y_3'), (1, 1
2, 'Y_4'), (1, 35, 'Y_4'), (1, 45, 'Y_4'), (1, 77, 'Y_4'), (8, 24, 'Y_5'), (8, 36, 'Y_5'), (8, 71, 'Y_5'), (8, 73, 'Y_5'), (8, 99, 'Y_5'), (7, 23, 'Y_6'), (
7, 37, 'Y_6'), (7, 64, 'Y_6'), (7, 71, 'Y_6'), (7, 80, 'Y_6'), (6, 22, 'Y_7'), (6, 38, 'Y_7'), (6, 48, 'Y_7'), (6, 50, 'Y_7'), (6, 72, 'Y_7'), (6, 98, 'Y_7'
), (5, 21, 'Y_8'), (5, 39, 'Y_8'), (5, 49, 'Y_8'), (41, 105, 'Z')]
```

Note: Adjacency matrix is not displayed as it is very huge.

Undirected Graph:



DFS and BFS paths:



```
Visited list:  [True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, Tr
ue, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, Tr
ue, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, Tr
ue, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, Tr
ue, True, True, True, True]
Stack of nodes:  []
Depth First Search Path:  ['src', 'NOR9', 'NAND9', 'OR2', 'NAND5', 'OR3', 'DFF6', 'NAND15', 'NOR15', 'NAND17', 'NOR12', 'NOT27', 'NOT26', 'DFF0', 'NAND12',
'NOT25', 'NOT19', 'NAND13', 'NOR11', 'NAND11', 'NOT24', 'DFF1', 'NAND2', 'NOR0', 'NOT1', 'NAND1', 'DFF3', 'NAND3', 'NOR3', 'AND3', 'NOR2', 'NOT5', 'OR0', 'N
AND4', 'NAND0', 'OR1', 'NOT20', 'NOT6', 'NOT0', 'AND2', 'AND0', 'NOR1', 'NOT8', 'NOT4', 'DFF2', 'NOT21', 'NOR10', 'NOT22', 'AND1', 'NOT7', 'NOT3', 'NOT18',
'NAND16', 'NOR14', 'NOT30', 'DFF4', 'NOT16', 'NOR6', 'AND4', 'NOR8', 'AND7', 'NOR7', 'NOT17', 'NOT13', 'NOT12', 'NOR5', 'snk', 'NOT32', 'NOR16', 'NAND18', '
NOR20', 'NOT34', 'NOR19', 'NOR17', 'NOT33', 'NOR18', 'NAND7', 'NOT10', 'DFF5', 'NOT29', 'NOR13', 'DFF7', 'NOT31', 'NAND6', 'NOT28', 'NOT14', 'NOT15', 'NOT9'
, 'NAND8', 'AND6', 'AND5', 'NOT11', 'NOR4', 'NAND14', 'NAND10', 'NOT23', 'AND16', 'AND15', 'AND14', 'AND13', 'AND12', 'AND11', 'AND10', 'AND9', 'AND8', 'NOT
2']
--------------------------------------------------------------------------------
Visited list:  [True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, Tr
ue, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, Tr
ue, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, Tr
ue, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, True, Tr
ue, True, True, True, True]
Queue of nodes:  []
Breadth First Search Path:  ['src', 'NOT2', 'AND8', 'AND9', 'AND10', 'AND11', 'AND12', 'AND13', 'AND14', 'AND15', 'AND16', 'NAND10', 'NAND14', 'NOR4', 'NOR9
', 'NOR14', 'NOR16', 'NOR12', 'NOR17', 'NOR10', 'NOT24', 'NOR18', 'NOT25', 'NOR19', 'NOT23', 'NOT31', 'NOR20', 'NOR13', 'DFF3', 'NOT20', 'NOR11', 'AND1', 'A
ND2', 'NAND3', 'NAND4', 'NOT11', 'AND5', 'AND6', 'NAND8', 'NAND9', 'DFF5', 'NOT30', 'NAND16', 'NOT32', 'NOT33', 'NAND18', 'NOT27', 'NAND17', 'NOT21', 'NAND1
1', 'NAND12', 'NOT34', 'NAND15', 'NOT29', 'NOT0', 'NOT6', 'NAND1', 'DFF2', 'NAND13', 'DFF0', 'NAND2', 'NOR1', 'DFF1', 'NOR2', 'OR1', 'NOR3', 'OR0', 'NOR0',
'DFF4', 'NAND7', 'NOR6', 'NAND6', 'NOR7', 'DFF7', 'OR3', 'NOR8', 'NOT9', 'NOT15', 'OR2', 'NOT13', 'NOT17', 'AND4', 'NOT28', 'NOR15', 'snk', 'NOT18', 'NOT26'
, 'NOT19', 'DFF6', 'NOT1', 'NOT5', 'NAND0', 'NOT22', 'NOT3', 'NOT7', 'AND0', 'NOT4', 'NOT8', 'AND3', 'NOT12', 'NOT16', 'NOT10', 'NOR5', 'AND7', 'NAND5', 'NO
T14']
--------------------------------------------------------------------------------
```