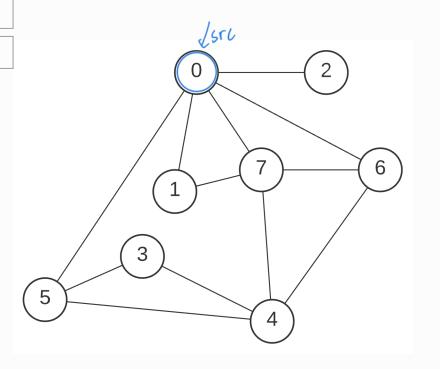


```
bfs(G, src):
         Input: graph G, vertex src
 3
         create visited array, initialised to false
         create predecessor array, initialised to −1
 6
         create queue Q
 8
         mark src as visited
 9
         enqueue src into Q
10
11
         while Q is not empty:
12
             dequeue v from Q
13
14
             for each neighbour w of v:
15
                 if w has not been visited:
16
                     mark w as visited
17
                     set predecessor of w to v
```





5

6

8

9

10 11

12

13 14

15

16

17

18

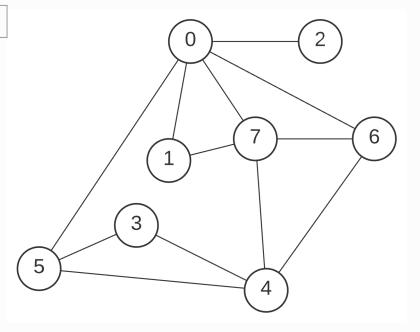
BFS

DFS

Т	F	F	F	F	F	F	F
-1	-1	-1	-1	-1	-1	-1	-1

```
bfs(G, src):
    Input: graph G, vertex src
    create visited array, initialised to false
    create predecessor array, initialised to -1
    create queue Q
   mark src as visited
    enqueue src into Q
    while Q is not empty:
        dequeue v from Q
        for each neighbour w of v:
            if w has not been visited:
                mark w as visited
                set predecessor of w to v
```

$$Q = \{ \underline{0} \}$$

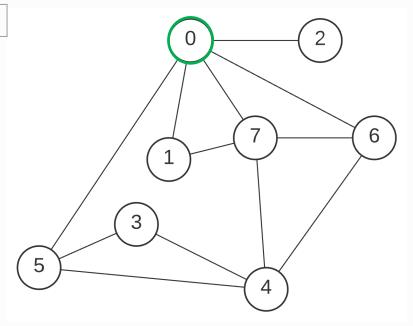




Т	Т	Т	F	F	Т	Т	Т
-1	0	0	-1	-1	0	0	0

```
Q = \{ 1, 2, 5, 6, 7 \}
```

```
bfs(G, src):
         Input: graph G, vertex src
 3
 4
         create visited array, initialised to false
 5
         create predecessor array, initialised to −1
 6
         create queue Q
 8
         mark src as visited
 9
         enqueue src into Q
10
11
         while Q is not empty:
12
             dequeue v from Q
13
14
             for each neighbour w of v:
15
                 if w has not been visited:
16
                     mark w as visited
17
                     set predecessor of w to v
18
                     enqueue w into Q
```





5

6

8

9

10 11

12

13 14

15

16

17

18

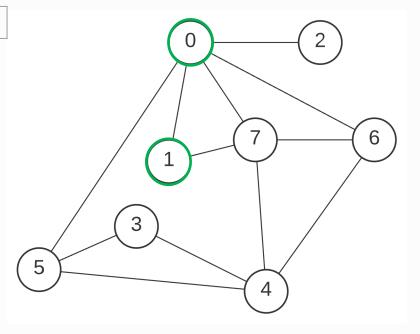
BFS

DFS

Т	Т	Т	F	F	Т	Т	Т
-1	0	0	-1	-1	0	0	0

```
bfs(G, src):
    Input: graph G, vertex src
    create visited array, initialised to false
    create predecessor array, initialised to −1
    create queue Q
   mark src as visited
    enqueue src into Q
    while Q is not empty:
        dequeue v from Q
        for each neighbour w of v:
            if w has not been visited:
                mark w as visited
                set predecessor of w to v
```

$$Q = \{ 2, 5, 6, 7 \}$$

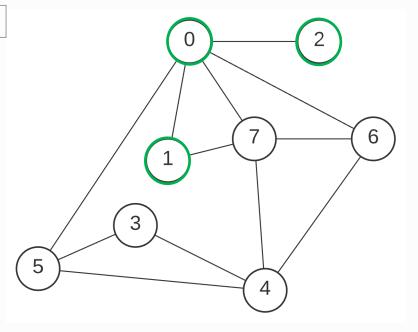




Т	Т	Т	F	F	Т	Т	Т
-1	0	0	-1	-1	0	0	0

```
bfs(G, src):
         Input: graph G, vertex src
 3
 4
         create visited array, initialised to false
 5
         create predecessor array, initialised to −1
 6
         create queue Q
 8
         mark src as visited
 9
         enqueue src into Q
10
11
         while Q is not empty:
12
             dequeue v from Q
13
14
             for each neighbour w of v:
15
                 if w has not been visited:
16
                     mark w as visited
17
                     set predecessor of w to v
18
                     enqueue w into Q
```

$$Q = \{ 5, 6, 7 \}$$





5

6

8

9

10 11

12

13 14

15

16

17

18

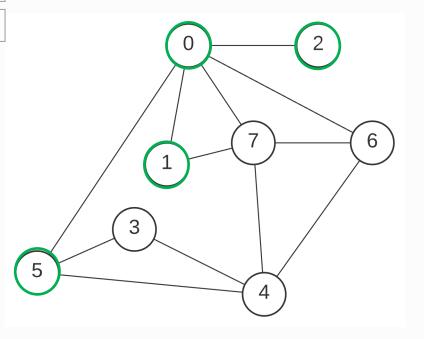
BFS

DFS

Т	Т	Т	Т	Т	Т	Т	Т
-1	0	0	5	5	0	0	0

```
bfs(G, src):
    Input: graph G, vertex src
    create visited array, initialised to false
    create predecessor array, initialised to −1
    create queue Q
   mark src as visited
    enqueue src into Q
    while Q is not empty:
        dequeue v from Q
        for each neighbour w of v:
            if w has not been visited:
                mark w as visited
                set predecessor of w to v
```

$$Q = \{ 6, 7, 3, 4 \}$$

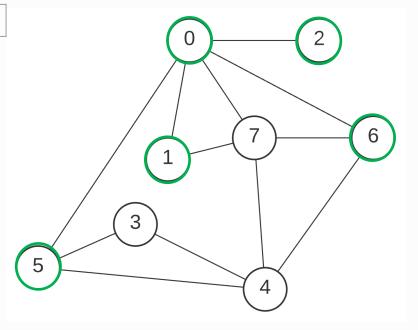




Т	Т	Т	Т	Т	Т	Т	Т
-1	0	0	5	5	0	0	0

```
bfs(G, src):
         Input: graph G, vertex src
 3
         create visited array, initialised to false
 5
         create predecessor array, initialised to −1
 6
         create queue Q
 8
         mark src as visited
 9
         enqueue src into Q
10
11
         while Q is not empty:
12
             dequeue v from Q
13
14
             for each neighbour w of v:
15
                 if w has not been visited:
16
                     mark w as visited
17
                     set predecessor of w to v
18
                     enqueue w into Q
```

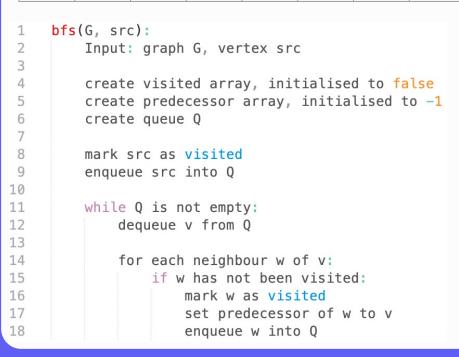
$$Q = \{ 7, 3, 4 \}$$

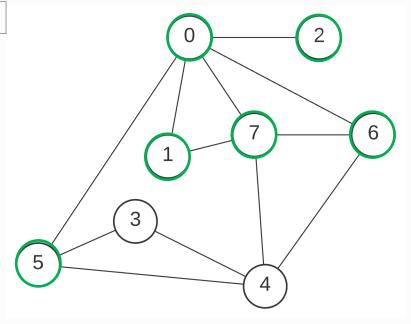




Т	Т	Т	Т	Т	Т	Т	Т
-1	0	0	5	5	0	0	0

```
Q = \{ 3, 4 \}
```



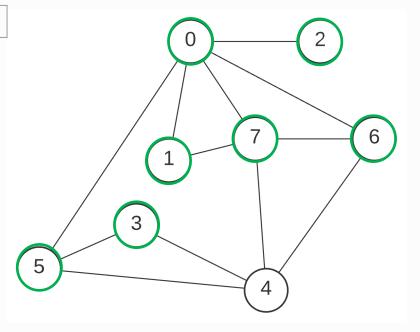




Т	Т	Т	Т	Т	Т	Т	Т
-1	0	0	5	5	0	0	0

```
Q = \{ \underline{4} \}
```

```
bfs(G, src):
         Input: graph G, vertex src
 3
 4
         create visited array, initialised to false
 5
         create predecessor array, initialised to -1
 6
         create queue Q
 8
         mark src as visited
 9
         enqueue src into Q
10
11
         while Q is not empty:
12
             dequeue v from Q
13
14
             for each neighbour w of v:
15
                 if w has not been visited:
16
                     mark w as visited
17
                     set predecessor of w to v
18
                     enqueue w into Q
```

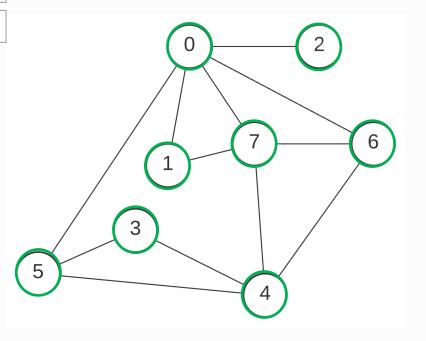




Т	Т	Т	Т	Т	Т	Т	Т
-1	0	0	5	5	0	0	0

```
bfs(G, src):
         Input: graph G, vertex src
 3
 4
         create visited array, initialised to false
 5
         create predecessor array, initialised to −1
 6
         create queue Q
 8
         mark src as visited
 9
         enqueue src into Q
10
11
         while Q is not empty:
12
             dequeue v from Q
13
14
             for each neighbour w of v:
15
                 if w has not been visited:
16
                     mark w as visited
17
                     set predecessor of w to v
18
                     enqueue w into Q
```

$$Q = \{ \}$$

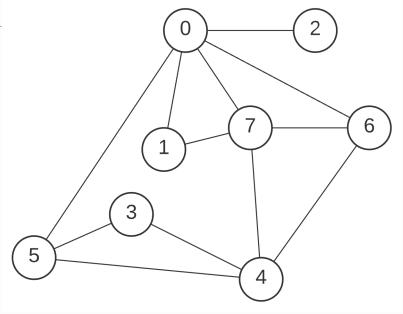




F	F	F	F	F	F	F	F
-1	-1	-1	-1	-1	-1	-1	-1

```
dfs(G, src):
         Input: graph G, vertex src
 4
         create visited array, initialised to false
         create predecessor array, initialised to −1
 6
         create stack S
 8
         push src onto S
 9
10
         while S is not empty:
             pop v from S
11
             if v has been visited:
12
13
                 continue
14
15
             mark v as visited
16
             for each neighbour w of v:
17
18
                 if w has not been visited:
19
                     set predecessor of w to v
20
                     push w onto S
```

$$S = \{ \underline{0} \}$$

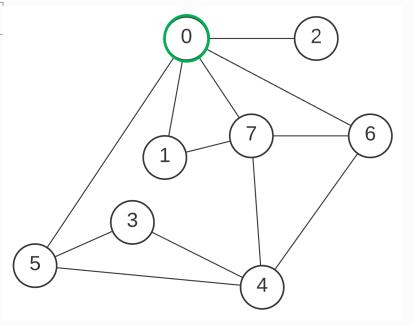




Т	F	F	F	F	F	F	F
-1	0	0	-1	-1	0	0	0

```
dfs(G, src):
         Input: graph G, vertex src
 4
         create visited array, initialised to false
         create predecessor array, initialised to −1
 6
         create stack S
 8
         push src onto S
 9
10
         while S is not empty:
             pop v from S
11
             if v has been visited:
12
13
                 continue
14
15
             mark v as visited
16
             for each neighbour w of v:
17
                 if w has not been visited:
18
19
                     set predecessor of w to v
20
                     push w onto S
```

$$S = \{7, 6, 5, 2, \underline{1}\}$$

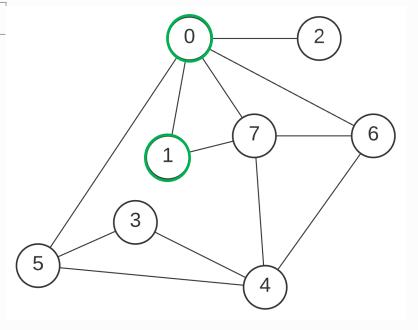




Т	Т	F	F	F	F	F	F
-1	0	0	-1	-1	0	0	1

```
dfs(G, src):
         Input: graph G, vertex src
         create visited array, initialised to false
         create predecessor array, initialised to −1
 6
         create stack S
 8
         push src onto S
 9
10
         while S is not empty:
             pop v from S
11
             if v has been visited:
12
13
                 continue
14
15
             mark v as visited
16
             for each neighbour w of v:
17
18
                 if w has not been visited:
19
                     set predecessor of w to v
20
                     push w onto S
```

$$S = \{ 7, 6, 5, 2, \underline{7} \}$$

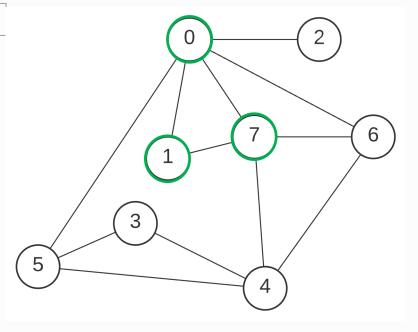




Т	Т	F	F	F	F	F	Т
-1	0	0	-1	7	0	7	1

```
dfs(G, src):
         Input: graph G, vertex src
         create visited array, initialised to false
         create predecessor array, initialised to −1
 6
         create stack S
 8
         push src onto S
 9
10
         while S is not empty:
             pop v from S
11
             if v has been visited:
12
13
                 continue
14
15
             mark v as visited
16
             for each neighbour w of v:
17
18
                 if w has not been visited:
19
                     set predecessor of w to v
20
                     push w onto S
```

$$S = \{ 7, 6, 5, 2, 6, 4 \}$$

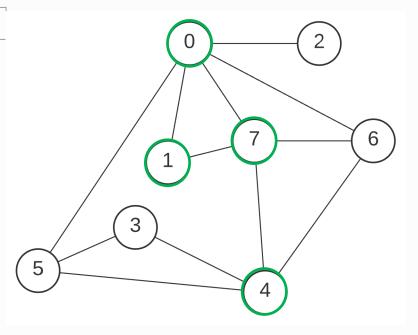




Т	Т	F	F	Т	F	F	Т
-1	0	0	4	7	4	4	1

```
dfs(G, src):
         Input: graph G, vertex src
         create visited array, initialised to false
         create predecessor array, initialised to −1
 6
         create stack S
 8
         push src onto S
 9
10
         while S is not empty:
             pop v from S
11
             if v has been visited:
12
13
                 continue
14
15
             mark v as visited
16
             for each neighbour w of v:
17
18
                 if w has not been visited:
19
                     set predecessor of w to v
20
                     push w onto S
```

```
S = \{ 7, 6, 5, 2, 6, 6, 5, 3 \}
```

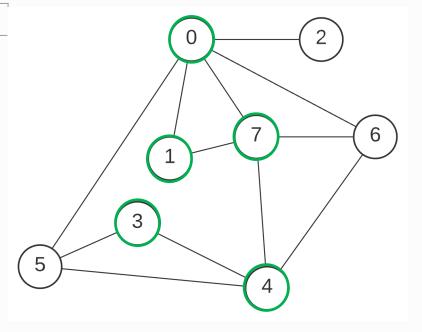




Т	Т	F	Т	Т	F	F	Т
-1	0	0	4	7	3	4	1

```
dfs(G, src):
         Input: graph G, vertex src
         create visited array, initialised to false
         create predecessor array, initialised to −1
 6
         create stack S
 8
         push src onto S
 9
10
         while S is not empty:
             pop v from S
11
             if v has been visited:
12
13
                 continue
14
15
             mark v as visited
16
             for each neighbour w of v:
17
18
                 if w has not been visited:
19
                     set predecessor of w to v
20
                     push w onto S
```

$$S = \{ 7, 6, 5, 2, 6, 6, \underline{5} \}$$

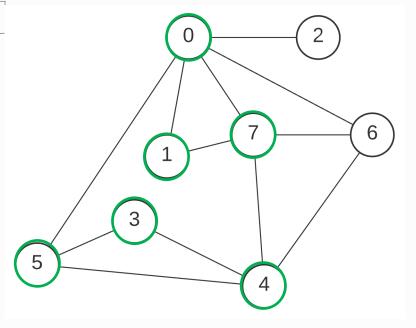




Т	Т	F	Т	Т	Т	F	Т
-1	0	0	4	7	3	4	1

```
dfs(G, src):
         Input: graph G, vertex src
         create visited array, initialised to false
         create predecessor array, initialised to −1
 6
         create stack S
 8
         push src onto S
 9
10
         while S is not empty:
             pop v from S
11
             if v has been visited:
12
13
                 continue
14
15
             mark v as visited
16
             for each neighbour w of v:
17
18
                 if w has not been visited:
19
                     set predecessor of w to v
20
                     push w onto S
```

```
S = \{ 7, 6, 5, 2, 6, \underline{6} \}
```

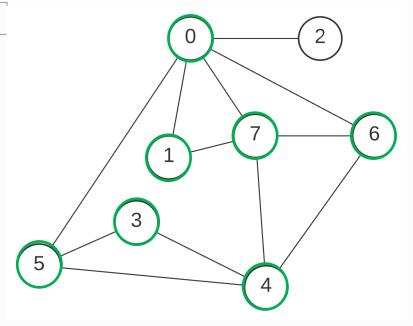




Т	Т	F	Т	Т	Т	Т	Т
-1	0	0	4	7	3	4	1

```
dfs(G, src):
         Input: graph G, vertex src
         create visited array, initialised to false
         create predecessor array, initialised to −1
 6
         create stack S
 8
         push src onto S
 9
10
         while S is not empty:
             pop v from S
11
             if v has been visited:
12
13
                 continue
14
15
             mark v as visited
16
             for each neighbour w of v:
17
18
                 if w has not been visited:
19
                     set predecessor of w to v
20
                     push w onto S
```

$$S = \{ 7, 6, 5, 2, \underline{6} \}$$

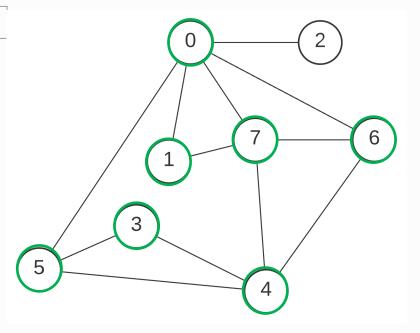




Т	Т	F	Т	Т	Т	Т	Т
-1	0	0	4	7	3	4	1

```
dfs(G, src):
         Input: graph G, vertex src
         create visited array, initialised to false
         create predecessor array, initialised to −1
 6
         create stack S
 8
         push src onto S
 9
10
         while S is not empty:
             pop v from S
11
             if v has been visited:
12
13
                 continue
14
15
             mark v as visited
16
             for each neighbour w of v:
17
18
                 if w has not been visited:
19
                     set predecessor of w to v
20
                     push w onto S
```

$$S = \{ 7, 6, 5, \underline{2} \}$$

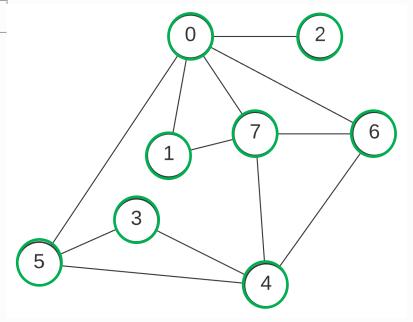




Т	Т	Т	Т	Т	Т	Т	Т
-1	0	0	4	7	3	4	1

```
dfs(G, src):
         Input: graph G, vertex src
         create visited array, initialised to false
         create predecessor array, initialised to −1
 6
         create stack S
 8
         push src onto S
 9
10
         while S is not empty:
             pop v from S
11
             if v has been visited:
12
13
                 continue
14
15
             mark v as visited
16
             for each neighbour w of v:
17
                 if w has not been visited:
18
19
                     set predecessor of w to v
20
                     push w onto S
```

$$S = \{ 7, 6, \underline{5} \}$$





Т	Т	Т	Т	Т	Т	Т	T
-1	0	0	4	7	3	4	1

```
dfs(G, src):
         Input: graph G, vertex src
         create visited array, initialised to false
         create predecessor array, initialised to −1
 6
         create stack S
 8
         push src onto S
 9
10
         while S is not empty:
             pop v from S
11
             if v has been visited:
12
13
                 continue
14
15
             mark v as visited
16
             for each neighbour w of v:
17
                 if w has not been visited:
18
19
                     set predecessor of w to v
20
                     push w onto S
```

