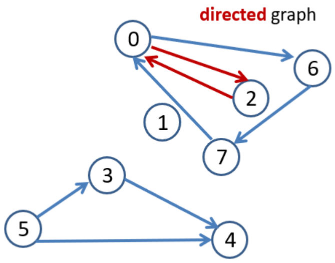


Vertex	0	4	5	1	7
In-degree					
Out-degree					

Give the adjacency matrix and adjacency list representation for the above graph.

Fill in the matrix representation.

Use row as source and column as destination for edges.

Give the Adjacency list representation.

Size __ array of ______________ (array items)

Depth-First Search (DFS) – Adj List

Space complexity: O(____)

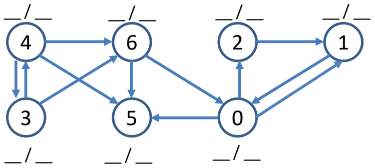
Time complexity:

Representation	DFS	DFS-Visit(G,u)
Adj LIST		
Adj MATRIX		

```
DFS(G)
    1.  For each vertex u of G
        a.  color[u] = WHITE
        b.  pred[u] = NIL
    2.  for (u = 0; u<G.V; u++) // for each vertex u of G
        a.  If color[u] == WHITE
            1.  DFS_visit(G, u, color, pred)

DFS_visit(G,u,color, pred)
    1.  color[u] = GRAY
    2.  ________________________________________
        a.  If color[y]==WHITE
            1.  pred[y] = u
            2.  DFS_visit(G,y, color, pred)
        b.  //if color[y]==GRAY then cycle found
    3.  color[u] = BLACK
```

Visited vertex		Pred

List:



Depth-First Search (DFS) – Adj Matrix

Space complexity: O(____)

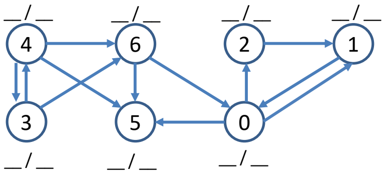
Time complexity:

Representation	DFS	DFS-Visit(G,u)
Adj LIST		
Adj MATRIX		

```
DFS(G)
    1.  For each vertex u of G
        a.  color[u] = WHITE
        b.  pred[u] = NIL
    2.  for (u = 0; u<G.V; u++) // for each vertex u of G
        a.  If color[u] == WHITE
            1.  DFS_visit(G, u, color, pred)

DFS_visit(G,u,color, pred)
    1.  color[u] = GRAY
    2.  ________________________________________
        a.  If color[y]==WHITE
            1.  pred[y] = u
            2.  DFS_visit(G,y, color, pred)
        b.  //if color[y]==GRAY then cycle found
    3.  color[u] = BLACK
```

Visited vertex		Pred

List:



Depth-First Search (DFS) – simple version

Space complexity: O(____)

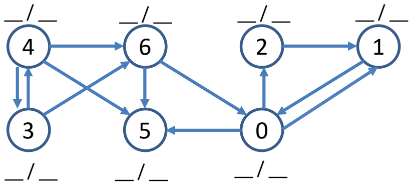
Time complexity:

Representation	DFS	DFS-Visit(G,u)
Adj LIST		
Adj MATRIX		

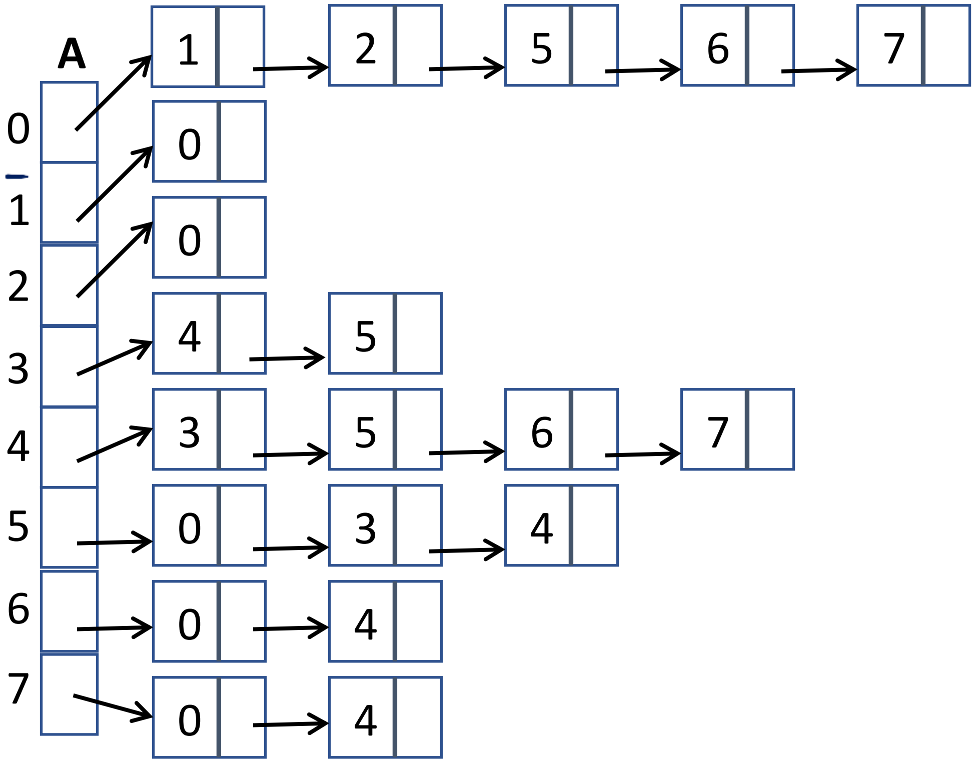
```
DFS(G)
    1. For each vertex u of G
        a. color[u] = WHITE
        b. pred[u] = NIL
    2. for (u = 0; u<G.V; u++) // for each vertex u of G
        a. If color[u] == WHITE
            1. DFS_visit(G, u, color, pred)

DFS_visit(G,u,color, pred)
    1. color[u] = GRAY
    2. For each y adjacent to u    // explore edge (u,y) // use increasing order for neighbors
        a. If color[y]==WHITE
            1. pred[y] = u
            2. DFS_visit(G,y, color, pred)
        b. //if color[y]==GRAY then cycle found
    3. color[u] = BLACK
```

Visited vertex		Pred

List:



	0	1	2	3	4	5	6	7
0	0	**1**	**1**	0	0	**1**	**1**	**1**
1	**1**	0	0	0	0	0	0	0
2	**1**	0	0	0	0	0	0	0
3	0	0	0	0	**1**	**1**	0	0
4	0	0	0	**1**	0	**1**	**1**	**1**
5	**1**	0	0	**1**	**1**	0	0	0
6	**1**	0	0	0	**1**	0	0	0
7	**1**	0	0	0	**1**	0	0	0


A
0
1
2
3
4
5
6
7
1 2 5 6 7
0
0
4 5
3 5 6 7
0 3 4
0 4
0 4