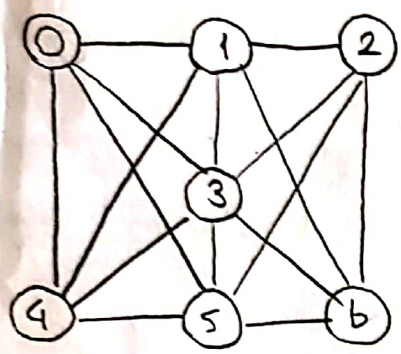
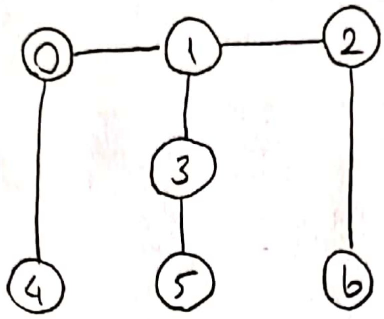
CSE 222 HOMEWORK 8

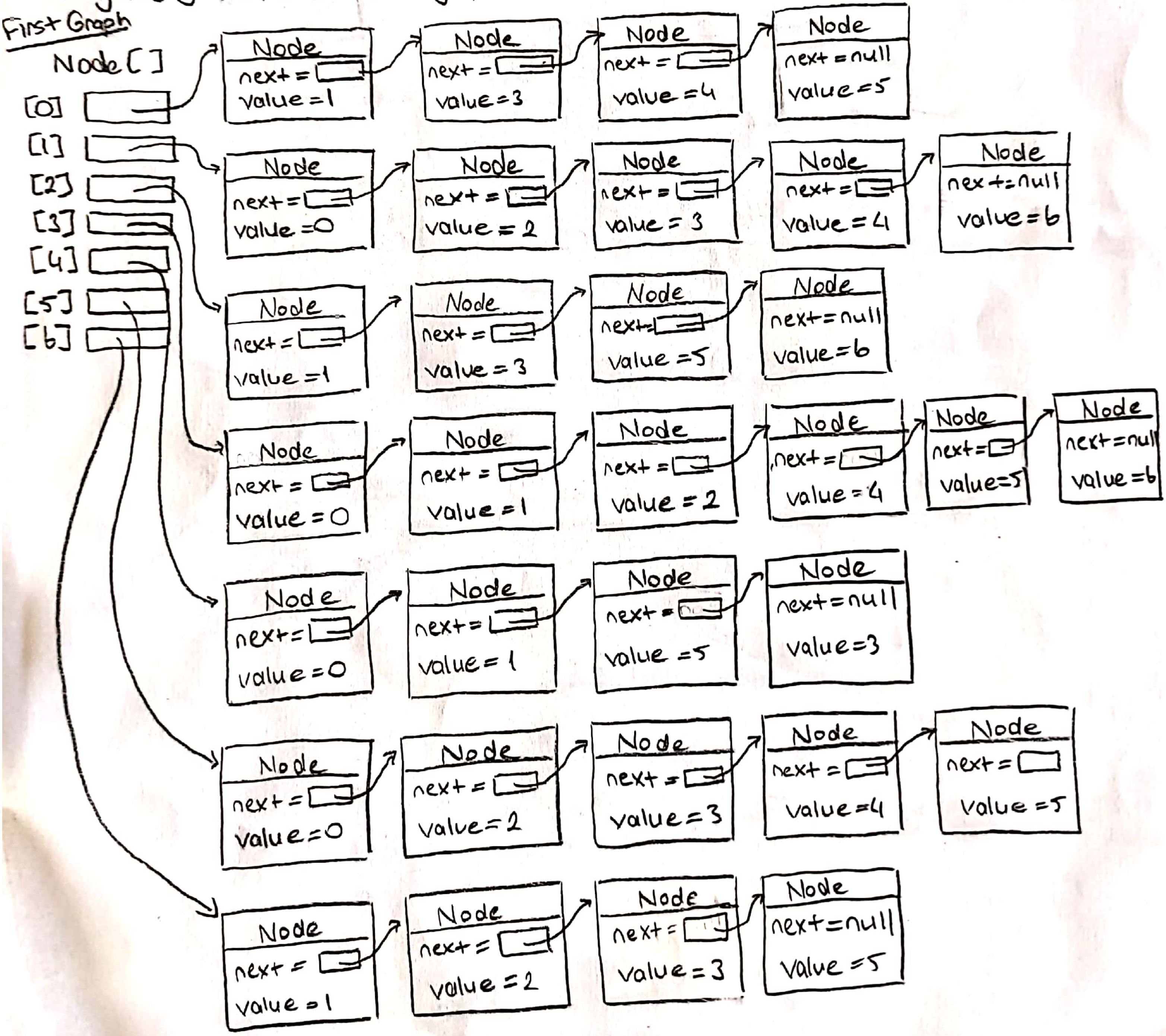
Question 1:



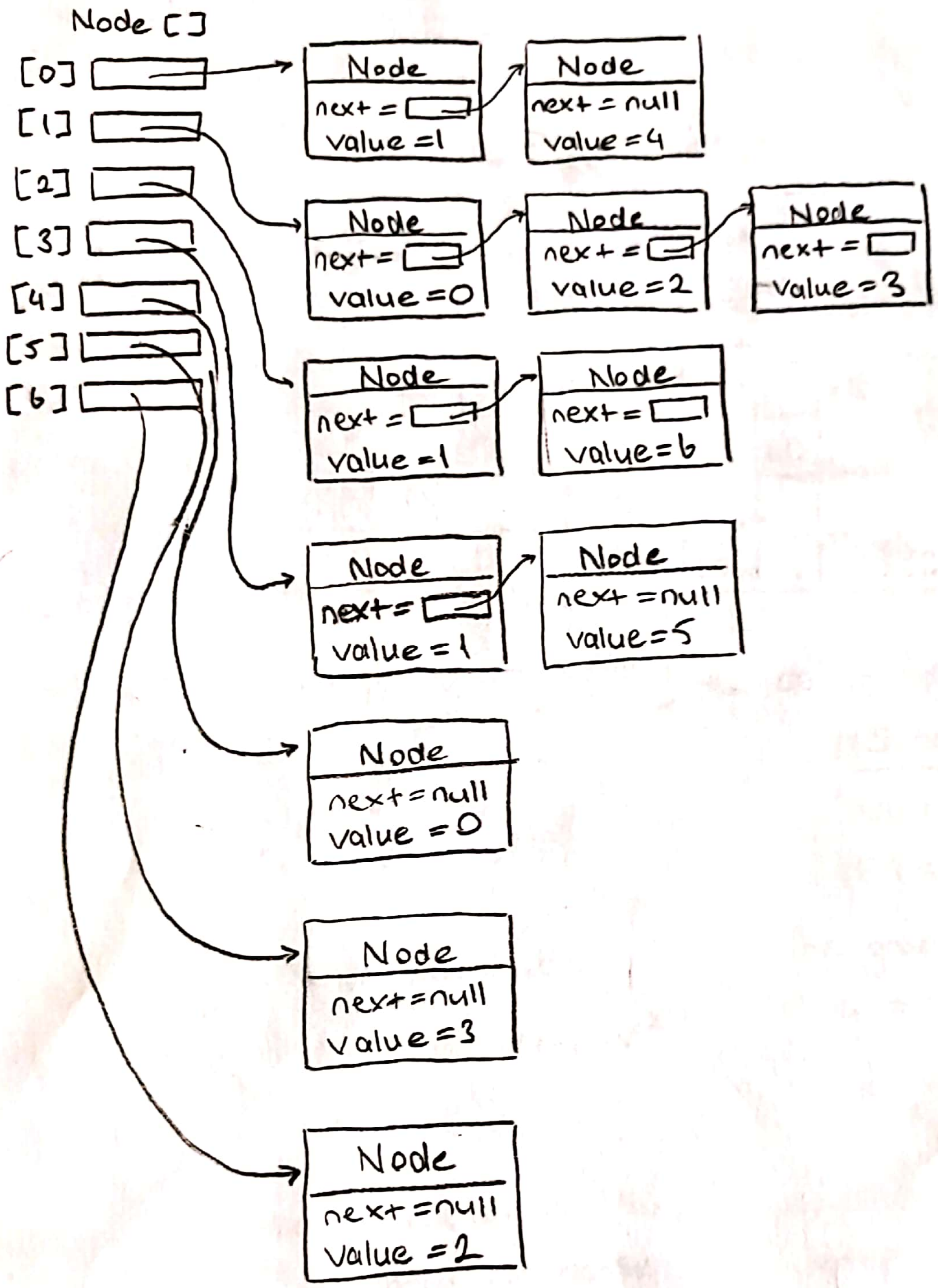


- Adjaceny lists : Both graphs are not directed.

First Graph



Second Graph



- Adjacency Matrix :

First Graph :

Column

Row

	[0]	[1]	[2]	[3]	[4]	[5]	[6]
[0]		1.0		1.0	1.0	1.0	
[1]	1.0		1.0	1.0	1.0		1.0
[2]		1.0		1.0		1.0	1.0
[3]	1.0	1.0	1.0		1.0	1.0	1.0
[4]	1.0	1.0		1.0		1.0	
[5]	1.0		1.0	1.0	1.0		1.0
[6]		1.0	1.0	1.0		1.0	

Second Graph :

Column

	[0]	[1]	[2]	[3]	[4]	[5]	[6]
[0]		1.0			1.0		
[1]	1.0		1.0	1.0			
[2]		1.0					1.0
[3]		1.0				1.0	
[4]	1.0						
[5]				1.0			
[6]			1.0				

- What are the $|V|=n$, the $|E|=m$, and the density? Which representation is better?

First graph

$|V| = 7$

$|E| = 32$

The graph is dense because $|E|$ is close to, but less than $|V|^2$

$|V|^2 = 49$ $|V| < |E| < |V|^2$

Adjacency matrix representation is better because the graph is dense

Second Graph

$|V| = 7$

$|E| = 12$

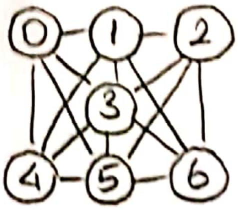
The graph is sparse because $|E|$ is much less than $|V|^2$

$|E| < |V|^2$

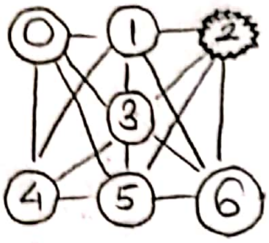
Adjacency List representation is better because the graph is sparse

— Draw DFS tree starting from vertex 2 In descending order;

○ unvisited ✹ being visited ⊘ visited



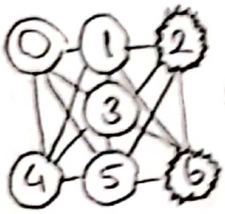
1- Start with vertex2 and



Discovery order : 2

Finish Order:

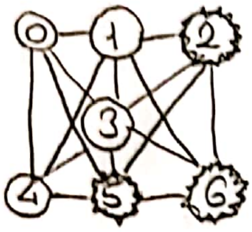
2- Choose a vertex that is not being visited (Decreasing order) 6



Discovery Order : 2,6

Finish Order:

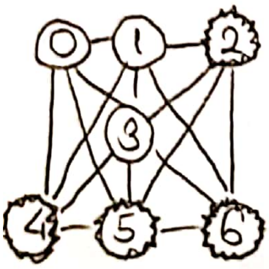
3- Choose an adjacent vertex



Discovery Order: 2,6,5

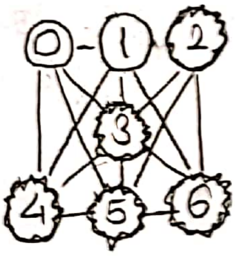
Finish Order:

4- Choose an adjecent vertex



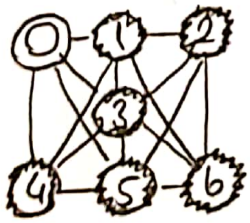
Discovery Order: 2,6,5,4

5- Choose an adjecent vertex



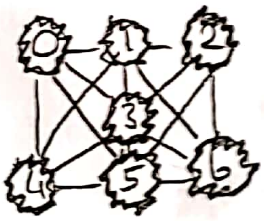
Discovery Order : 2,6,5,4,3

6- Choose an. adjacent vertex



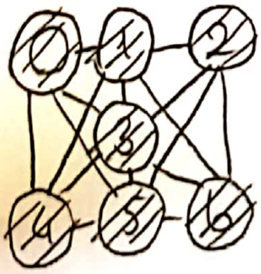
Discovery Order : 2,6,5,4,3,1

7- Choose an adjacent vertex



Discovery Order : 2,6,5,4,3,1,0

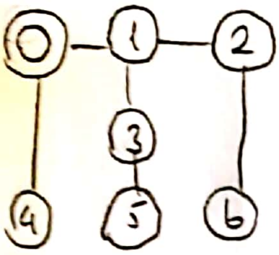
8- Mark all nodes as visited use discovery order as stack



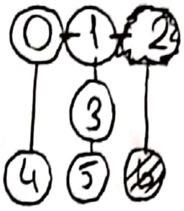
Discovery Order : 2,6,5,4,3,1,0

Finish Order: 0,1,3,4,5,6,2

Second Graph



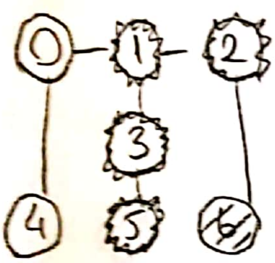
1- Start from index 2
mark it as being visited
go to it most largest adjacent
index and mark it as well



Discovery Order : 2,6

Finish Order : 6

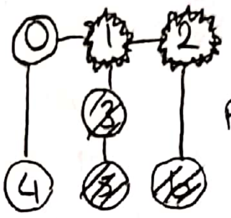
2- Mark 1 is being visited
Mark 3 is being visited
Mark 5 is being visited



Discovery Order: 2,6,1,3,5

Finish Order : 6

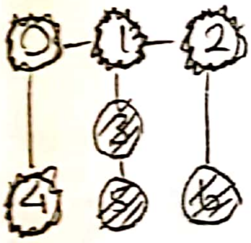
3- Mark 5 and then 3 as visited



Discovery Order: 2,6,1,3,5

Finish Order : 6,5,3

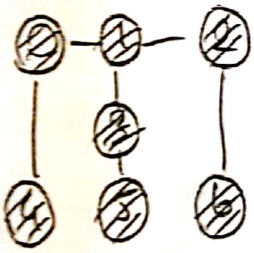
4- Mark 0 then 4 as being visited



Discovery Order : 2,6,1,3,5,0,4

Finish Order : 6,5,3

5- Mark all remaining elements from stack as vistide

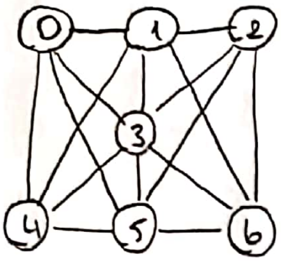


Discovery Order : 2,6,1,3,5,0,4

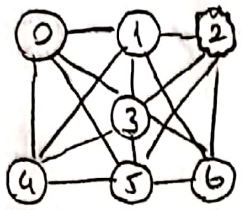
Finish Order : 6,5,3, 4,0,1,2

- Draw BFS tree starting from vertex 2 in descending order.

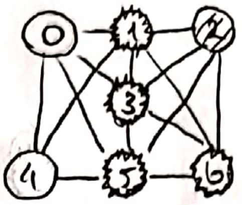
First Graph   ○ unvisited   ✺ identified   ⊘ visited



1- identify the first node



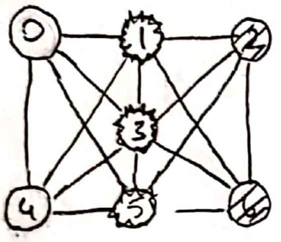
2- while visiting identify its adjacent nodes



Queue : 6,5,3,1

Visiting sequence : 2

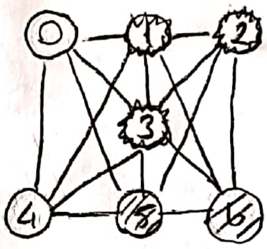
3- Visit the first node in queue, 6



Queue : 5,3,1

Visiting Sequence : 2,6

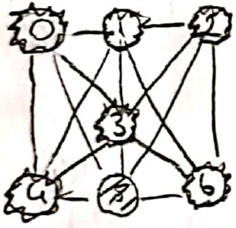
4- There is no adjacent nodes to 6 that is not identified or visited. Visit first node, 5



Queue 3,1

Visiting Sequence : 2,6,5

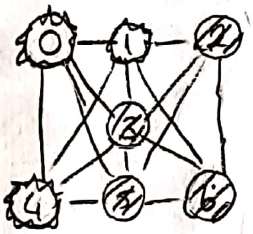
5- Identify the nodes that is adjacent and not visited or identified



Queue : 3,1,4,0

Visiting Sequence 2,6,5

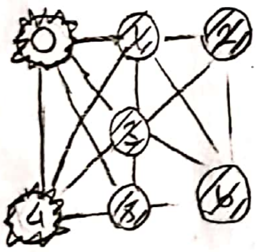
6- Visit the first node in queue, 3



Queue : 1,4,0

Visiting Sequence : 2,6,5,3

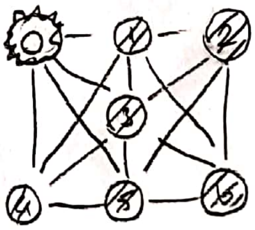
7- Visit the first node in queue, 1



Queue : 4,0

Visiting Sequence : 2,6,5,3,1

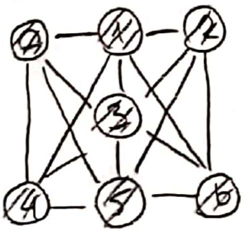
8 - Visit the first node in queue, 4



Queue : 0

Visiting Sequence : 2,6,5,3,1,4

9- Visit the first node in queue, 0

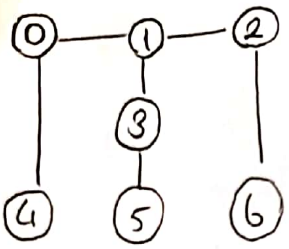


Queue : Empty

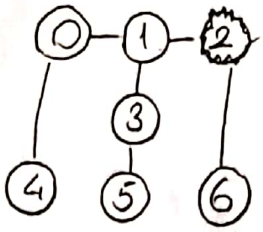
Visiting Sequence : 2,6,5,3,1,4,0

int[] parent
[0] = 5
[1] = 2
[2] = -1
[3] = 2
[4] = 5
[5] = 2
[6] = 2

Second graph : ○ unidentified ▨ visited ✹ Identified



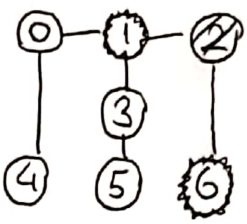
1- Identify 2



Queue : 2

Visit Sequence :

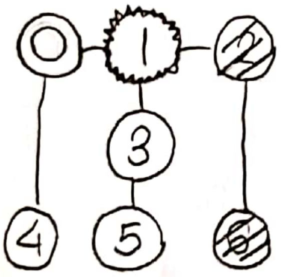
2- Visit first node, 2 and identify adjacency nodes



Queue : 6, 1

Visit Sequence : 2

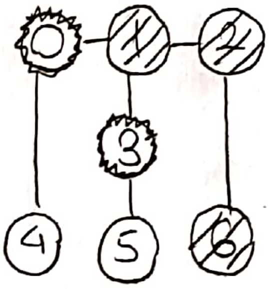
3- Visit first node, 6 and identify adjacency nodes that is not already visited or identified



Queue : 1

Visit Sequence : 2, 6

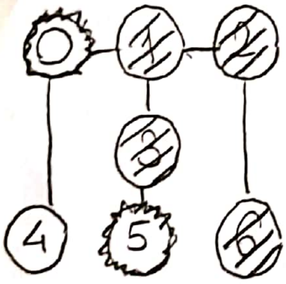
4- Visit first node, 1 and identify adjacens nodes



Queue : 3, 0

Visit Sequence : 2, 6, 1

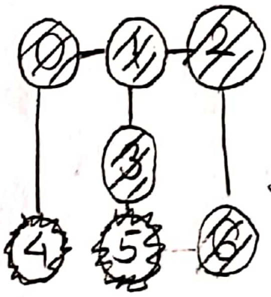
5- Visit first node, 3, and identify adjacency nodes



Queue : 0, 5

Visit Sequence : 2, 6, 1, 3

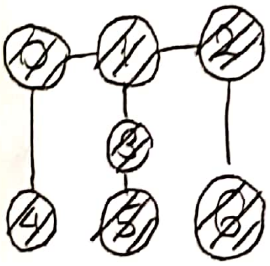
6- Visit first node, 0 identify adjacency nodes



Queue : 5, 4

Visit Sequence : 2, 6, 1, 3, 0

6- Visit remaing nodes



Queue : Empty

Visit Sequence : 2, 6, 1, 3, 0, 5, 4

parent [ ]

[0] = 1

[1] = 2

[2] = -1

[3] = 1

[4] = 0

[5] = 3

[6] = 2

CamScanner ile tarandı