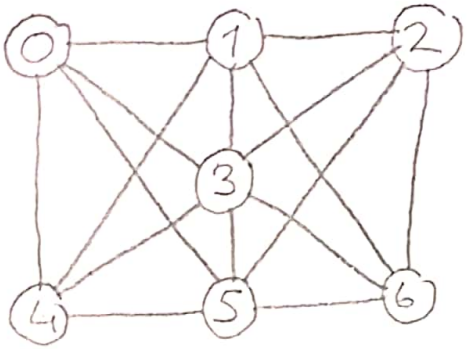
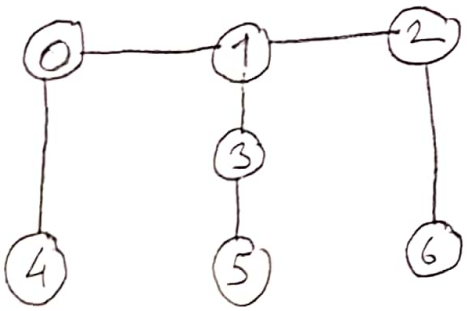
Homework #8 - part 1

Muhammed Yasir Fidan
161044056



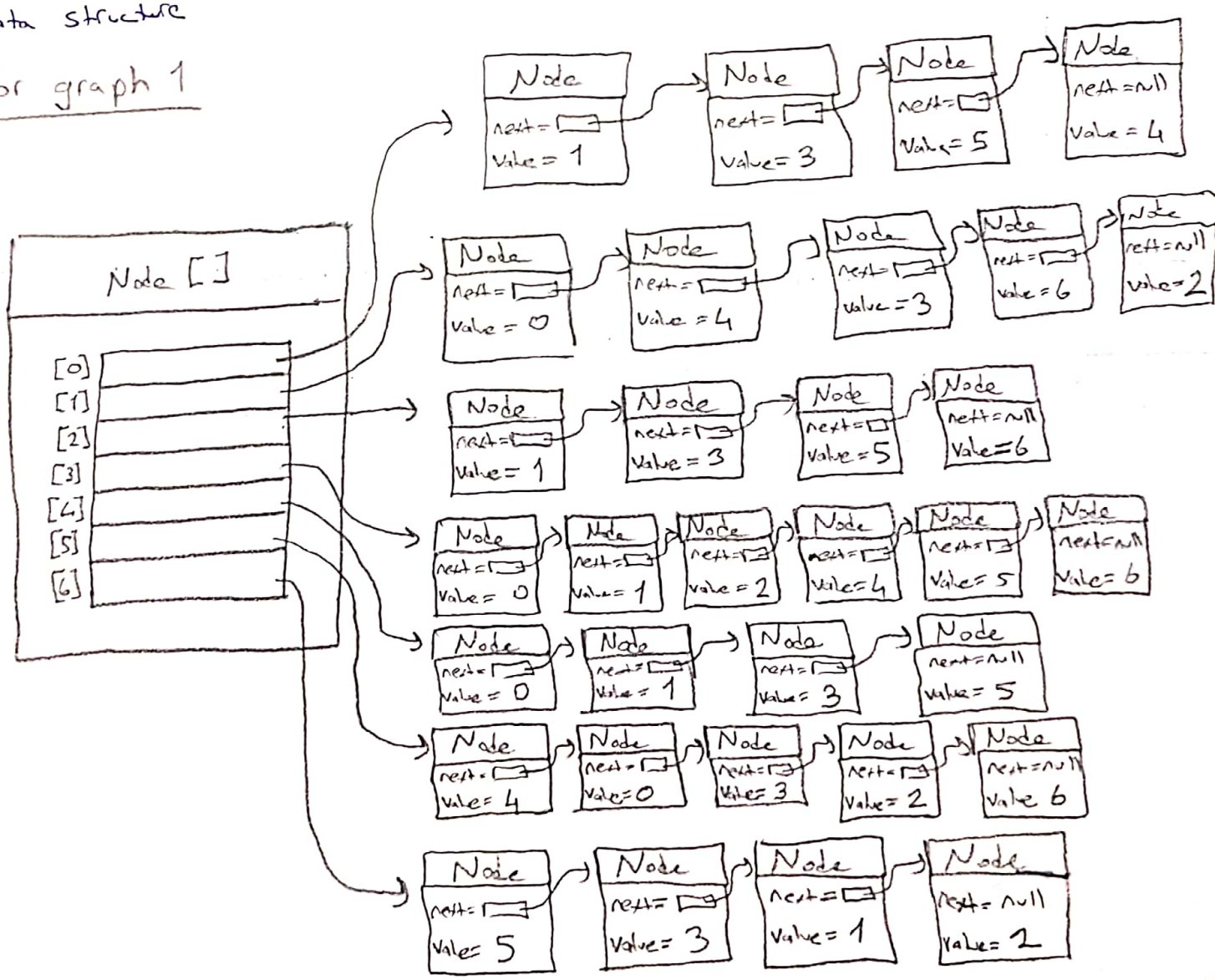
graph 1



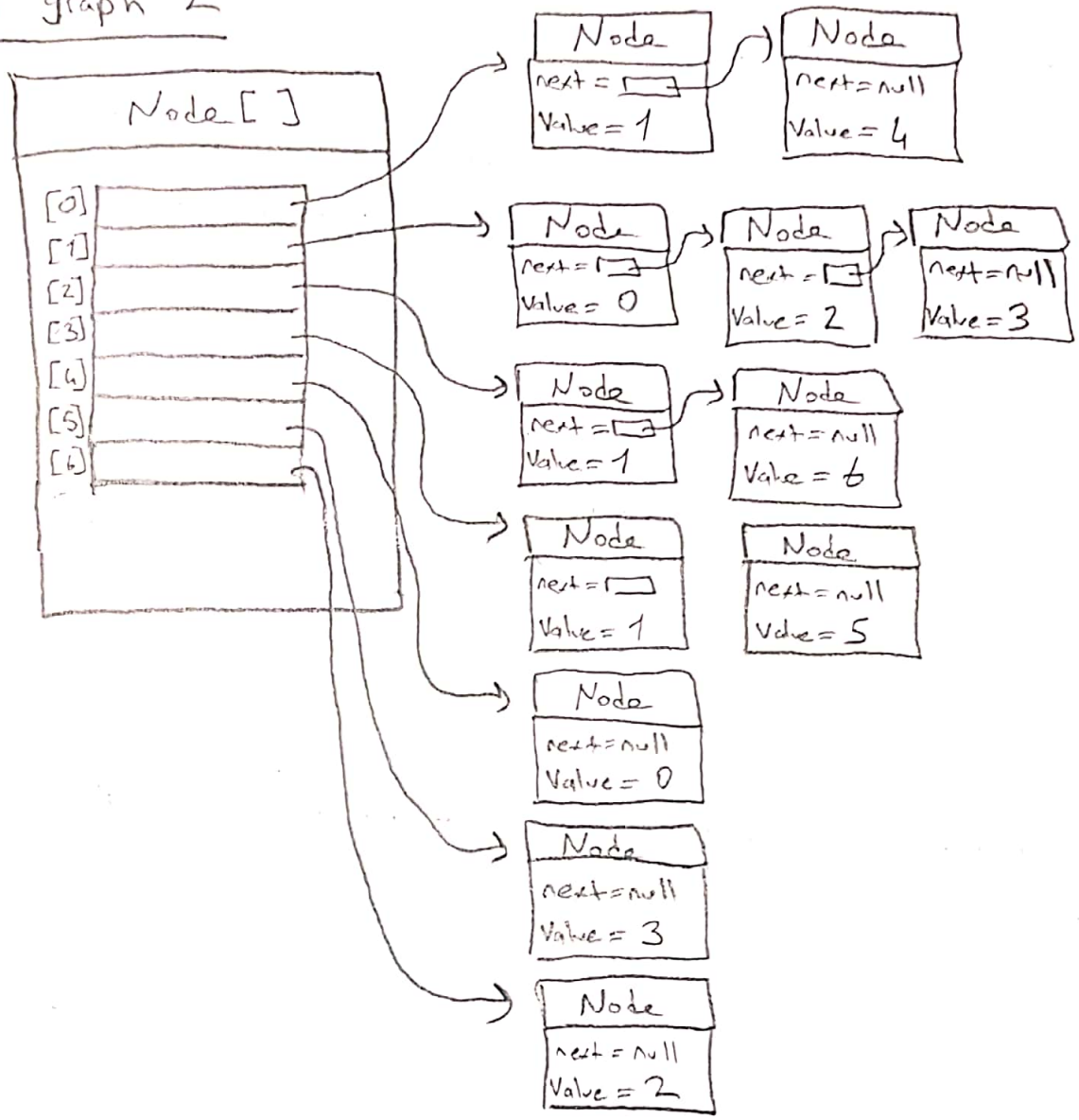
graph 2

- Represent the graphs above using adjacency lists. Draw the corresponding data structure

for graph 1



for graph 2



- Represent the graphs using an adjacency matrix. Draw the corresponding data structure

for graph 1

	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	

for graph 2

	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				

- For each graph, what are the $|V|=n$, the $|E|=m$, and the density? Which representation is better for each graph? Explain your answer.

for graph 1

$|V|=n=7$ $|E|=m=32$

This graph is a dense graph because $|E|$ close to $|V|^2$ For dense graphs, the adjacency matrix gives better performance, because of that use matrix representation for graph 1.

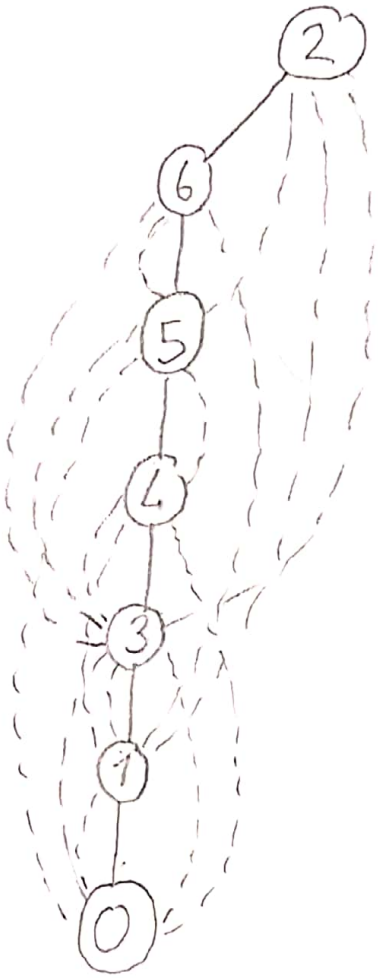
for graph 2

$|V|=n=7$ $|E|=m=6$

This graph is a sprse graph because $|E|$ is much less than $|V|^2$. For a sparse graph, the adjacency list representation gives better performce, because of that use adjacency list representation for graph 2.

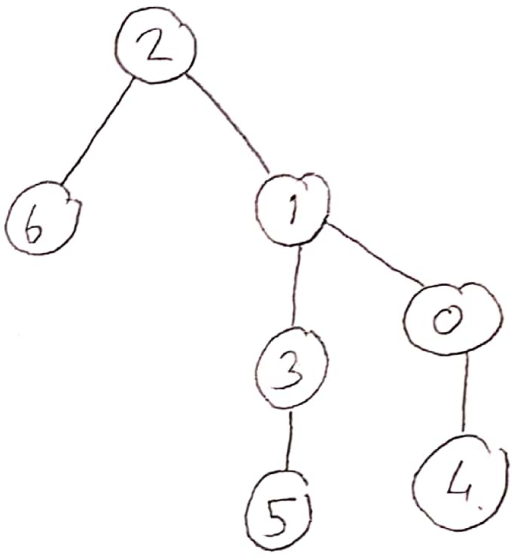
- Draw the DFS tree starting from vertex 2 and traversing the vertices adjacent to a vertex in descending order.

for graph 1



- Start from 2
- Visit 2's bigger adjacent (6)
- Visit 6's bigger adjacent (5)
- Visit 5's bigger adjacent (4)
- Visit 4's bigger adjacent (3)
- Visit 3's bigger adjacent (1)
- Visit 1's bigger adjacent (0)
- Return back but all nodes visited in graph
- So just draw back edges and finish

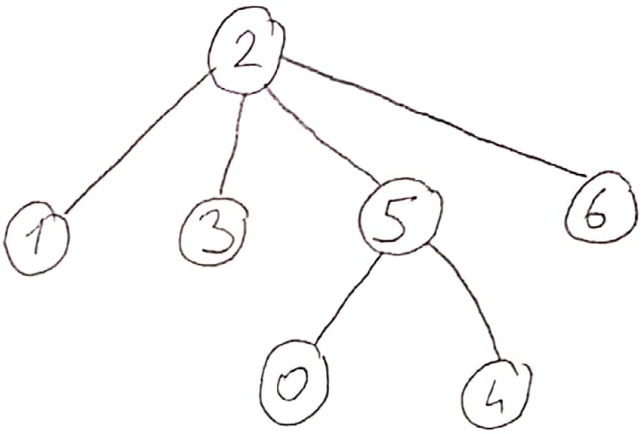
for graph 2



- Start from 2
- Visit 2's bigger adjacent (6)
- There is no unvisited adjacent of 6 so return 2.
- Visit 2's unvisited bigger adjacent (1)
- Visit 1's bigger unvisited adjacent (3)
- Visit 3's bigger unvisited adjacent (5)
- There is no unvisited adjacent of 6 so return back to 1.
- Visit 1's bigger unvisited adjacent (0)
- Visit 0's bigger unvisited adjacent (4)
- All nodes visited. Just draw back edges and finish.

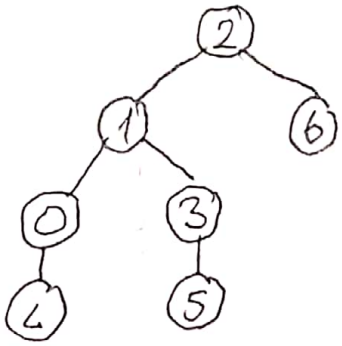
- Draw BFS tree starting from vertex 2 and traversing the vertices adjacent to a vertex in descending order.

for graph 1



- Visit the start node first (2)
- then all nodes that are adjacent to it (1, 3, 5, 6)
- then all nodes that can be reached by a path from the start node containing two edges (0, 4)
- All nodes visited, search completed.

for graph 2



- Visit the start node first (2)
- then all nodes that are adjacent to it (1, 6)
- then all nodes that can be reached by a path from the start node containing two edges (0, 3)
- then all nodes that can be reached by a path from the start node containing three edges (4, 5)
- All nodes visited, search completed.