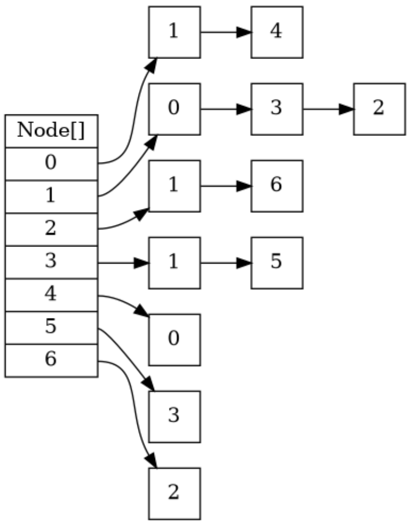
GIT Department Of Computer Engineering
CSE 222/505 - Spring 2020
Homework 8 Part 1

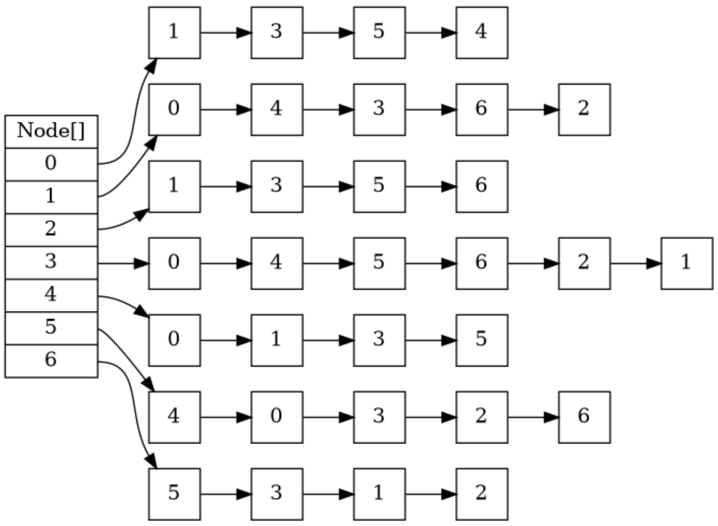
Fatih Kaan Salgır
171044009

1 Adjacency List Representation

1.1 First Graph



1.2 Second Graph



2 Adjacency Matrix Representation

2.1 First Graph

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

2.2 Second Graph

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

3

Density is the ratio of the number of edges with respect to the maximum possible edges. Since both graphs are undirected, densitiy is;

$$D = \frac{2|E|}{|V|(|V|-1)}$$

3.1 First Graph

- $|V| = 7$
- $|E| = 16$
- $D = \frac{2\times16}{7\times6} = 0.76$

3.2 Second Graph

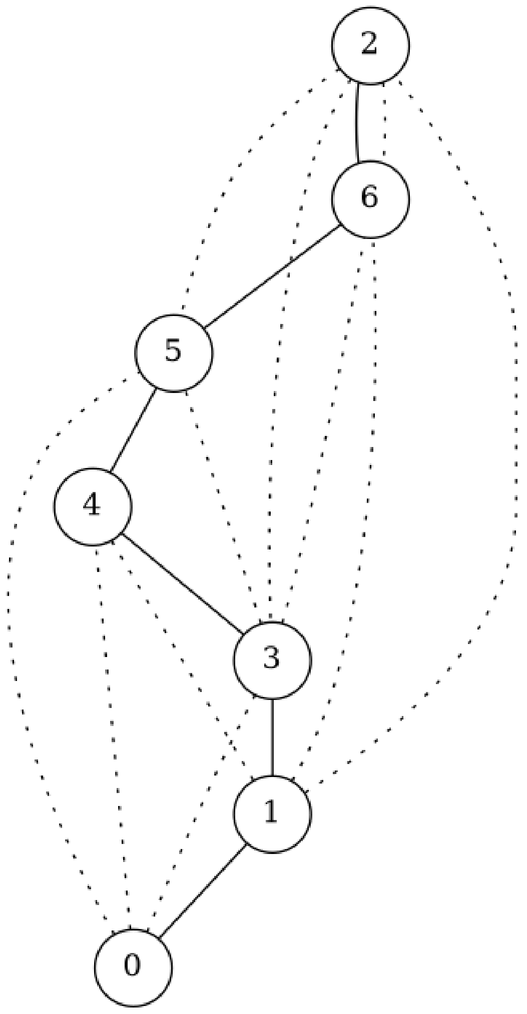
- $|V| = 7$
- $|E| = 6$
- $D = \frac{2\times6}{7\times6} = 0.29$

To compare the representations we can consider **time efficiency** and **storage efficiency**. With respect to time efficiency, first graph is relatively dense, therefore adjacency matrix representation is better. Because even though examining all edges requires $O(V^2)$, checking if (u, v) is an edge requires constant time. Similarly, since second graph is sparse, list representation is better. Although checking if (u, v) is an edge requires $O(E_u)$, complexitiy of examining all nodes is $O(E)$.

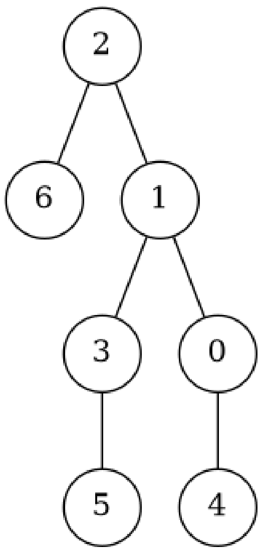
If we consider storage efficiency; in list representation we need to store the next cell address beside it's value. In matrix representation regardless of the number of edges, we create matrix for all possible edges. Therefore using matrix representation for dense graphs, similarly using list representation for sparse graphs is more efficient in terms of storage.

4 DFS Tree

4.1 First Graph

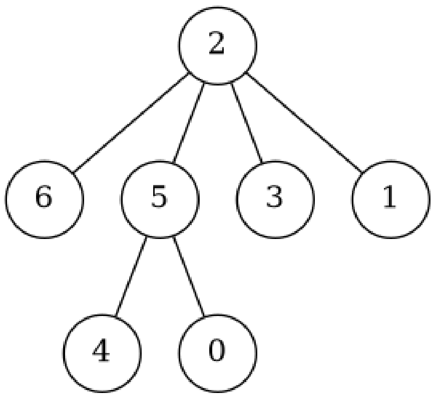


4.2 Second Graph



5 BFS Tree

5.1 First Graph



5.2 Second Graph