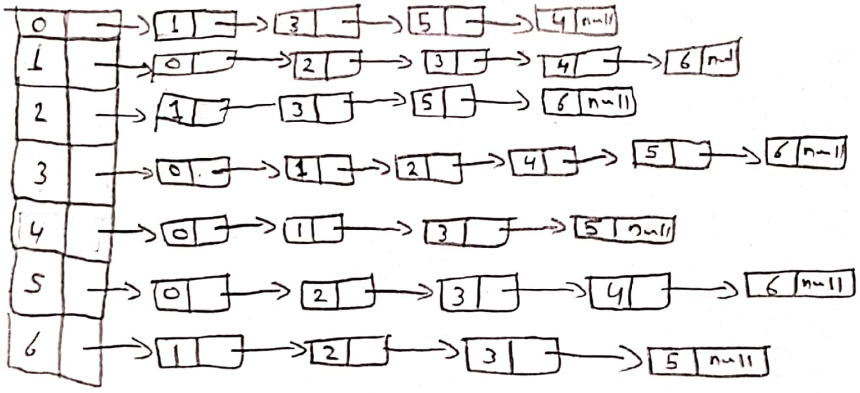
Muhammed Eyüp
18010426 79

Q1: graph a: V: {0, 1, 2, 3, 4, 5, 6}
E: {(0,1), (0,3), (0,5), (0,4), (1,0), (1,2), (1,3), (1,4),
(1,6), (2,1)(2,3), (2,5), (2,6), (3,0), (3,1), (3,2), (3,4),
(3,5), (3,6), (4,0), (4,1), (4,3), (4,5), (5,0), (5,2), (5,3),
(5,4), (5,6), (6,1), (6,2), (6,3), (6,5)}

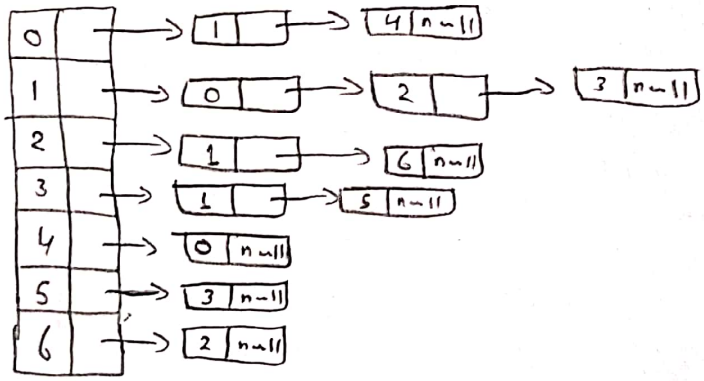
graph b: V: {0, 1, 2, 3, 4, 5, 6}
E: {(0,1), (0,4), (1,0), (1,2), (1,3), (2,1), (2,6),
(3,1), (3,5), (4,0), (5,3), (6,2)}

a) Adjacency list representation:

graph a:



graph b:



Muhammed Eyüp
18010426 79

b) Adjacency matrix representation:

graph a:

	0	1	2	3	4	5	6
0		1		1	1	1	
1	1		1	1	1		1
2		1		1		1	1
3	1	1	1		1	1	1
4	1	1		1		1	
5	1		1	1	1		1
6		1	1	1		1	

graph b:

	0	1	2	3	4	5	6
0		1			1		
1	1		1	1			
2		1					1
3		1				1	
4	1						
5				1			
6			1				

c) graph a: $|V| = 7 \quad |E| = 16 \quad \text{Density} = \frac{|E|}{|V|^2} = \frac{16}{49}$

for this graph a matrix representation is better since it's a somewhat dense graph and almost complete so it's better to take advantage of the efficency of a matrix on dense graphs

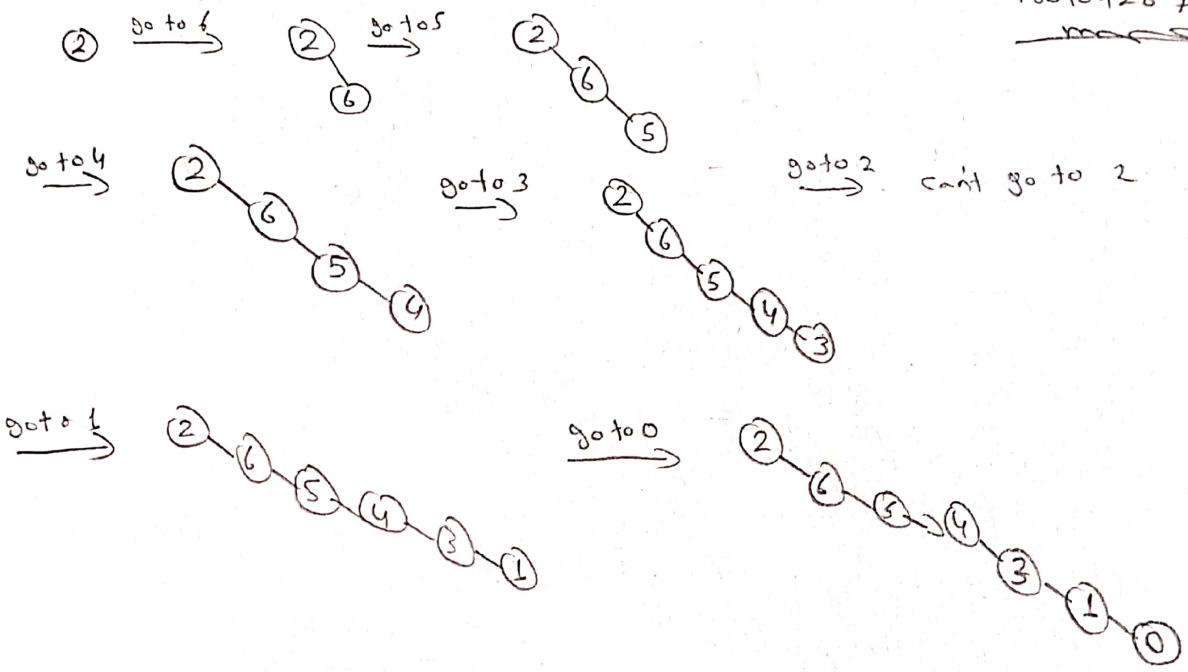
graph b: $|V| = 7 \quad |E| = 6 \Rightarrow \text{Density} = \frac{|E|}{|V|^2} = \frac{6}{49}$

for this graph it's defenetly better to use a List representation because the graph is sparse and adjacency lists are much more efficient on sparce graphs

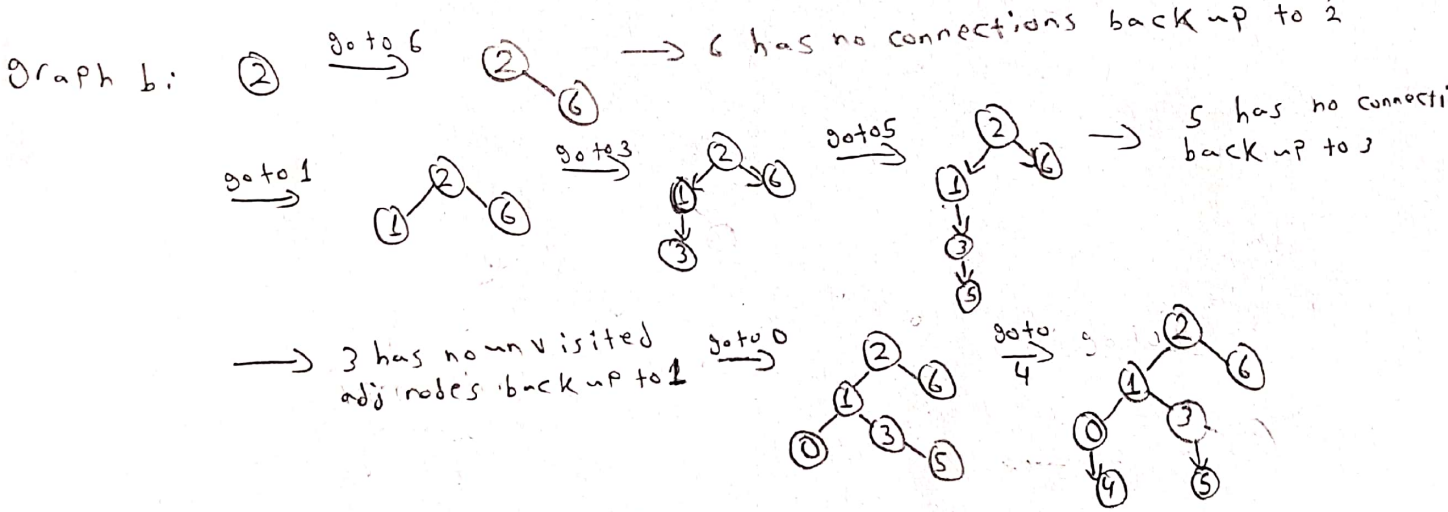
Muhammed Eyüp
18010426 79

d) drawing DFS tree

graph a:



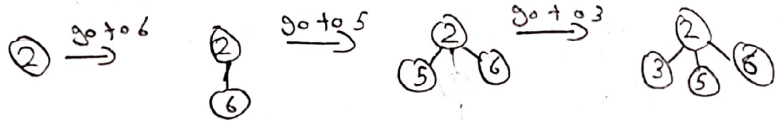
graph b:

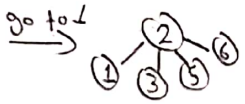


Muhammed Eyup
1801042679

e) Drawing BFS tree

graph a:





→ 2 has no more adj nodes go to adj nodes of 6

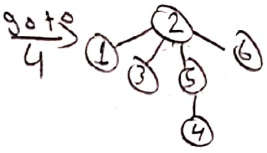
go to 5 → 5 is already visited

go to 3 → 3 is already visited

go to 1 → 1 is already visited

→ visit adj nodes of 5

go to 6 → 6 is already visted



go to 3 → 3 is already visited

go to 2 → already visited

→ visit adj nodes of 3

go to 6 → Visited

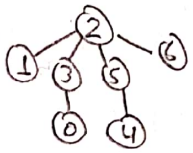
go to 5 → visited

go to 4 → visited

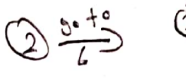
go to 2 → visited

go to 1 → visited

go to 0



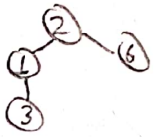
graph b:



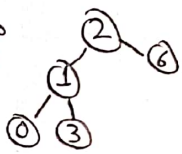
→ visit adj nodes of 6 → no nodes

→ visit adj nodes of 1

go to 3

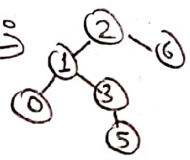


go to 0



→ visit adj nodes of 3

go to 5



→ visit adj nodes of 5 → no nodes

→ visit adj nodes of 0

go to 4