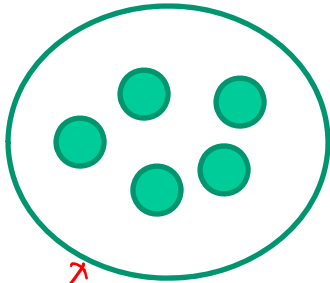


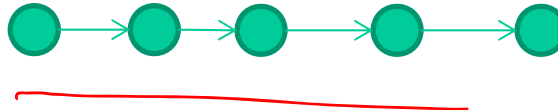
CSE 100

Graph Traversals

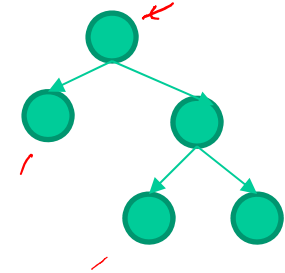
Kinds of Data Structures



Unstructured structures
(sets)



Sequential, linear structures
(arrays, linked lists)

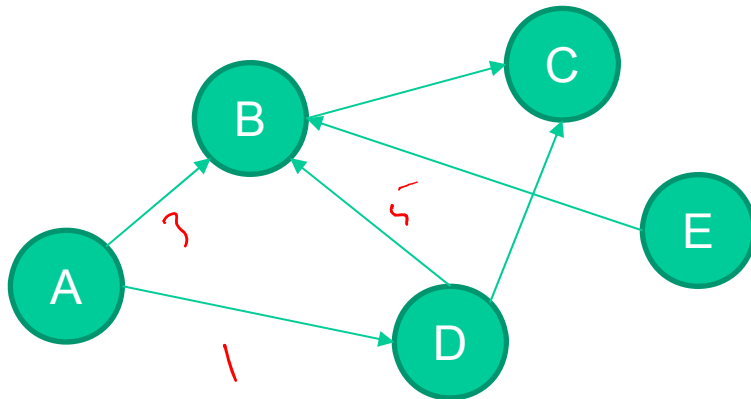


Hierarchical structures
(trees)

Graphs

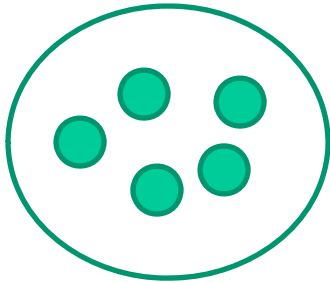
Consist of:

- A collection of elements ("nodes" or "vertices")
- A set of connections ("edges" or "links" or "arcs") between pairs of nodes.
 - Edges may be directed or undirected
 - Edges may have weight associated with them



Graphs are not hierarchical or sequential, no requirements for a "root" or "parent/child" relationships between nodes

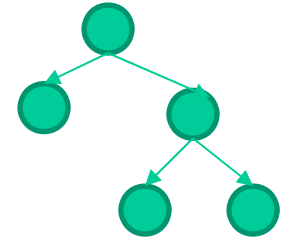
Kinds of Data Structures



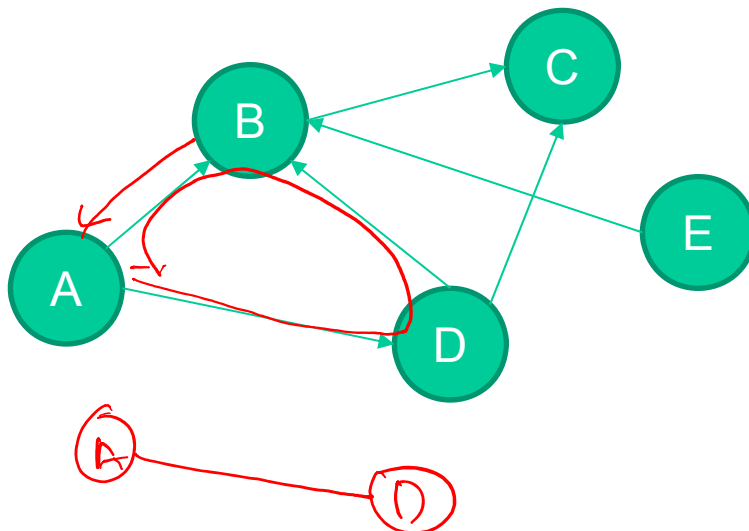
Unstructured structures
(sets)



Sequential, linear structures
(arrays, linked lists)



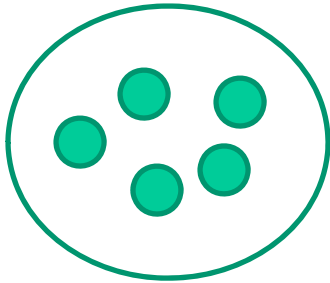
Hierarchical structures
(trees)



Graphs

- A. They consist of both vertices and edges
- B. They do NOT have an inherent order
- C. Edges may be weighed or unweighted
- D. Edges may be directed or undirected
- E. They may contain cycles

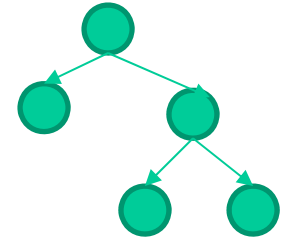
Kinds of Data Structures



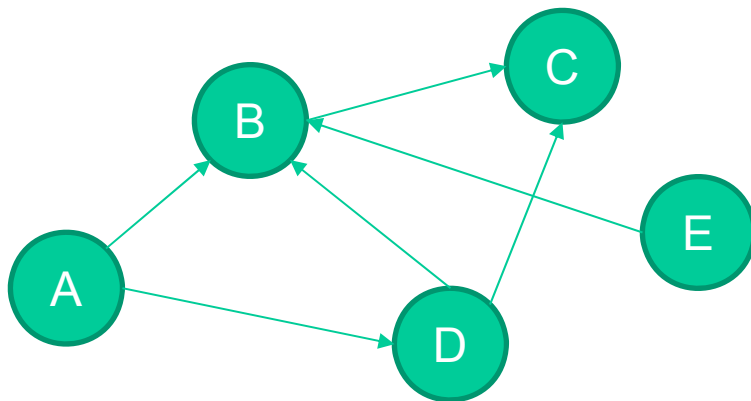
Unstructured structures
(sets)



Sequential, linear structures
(arrays, linked lists)



Hierarchical structures
(trees)

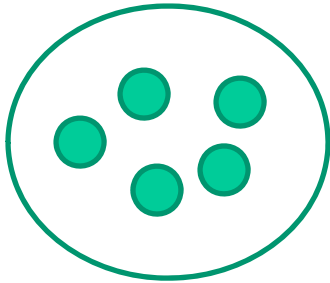


Graphs

Which of the following is true?

- A. A graph can always be represented as a tree
- ☒ B. A tree can always be represented as a graph
- C. Both A and B
- D. Neither A or B

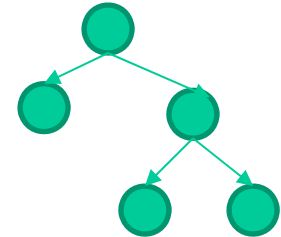
Kinds of Data Structures



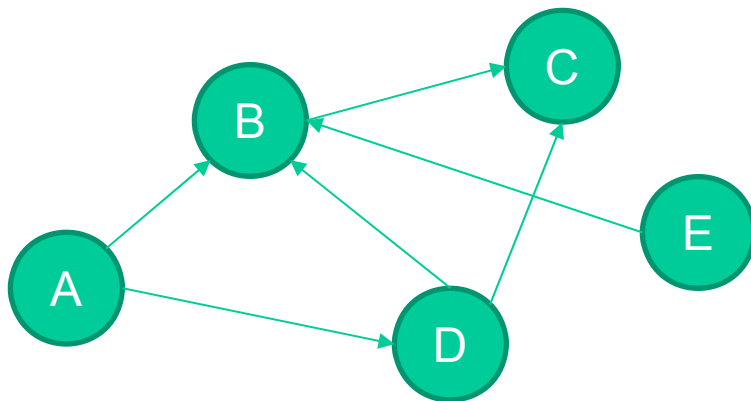
Unstructured structures
(sets)



Sequential, linear structures
(arrays, linked lists)



Hierarchical structures
(trees)



Graphs

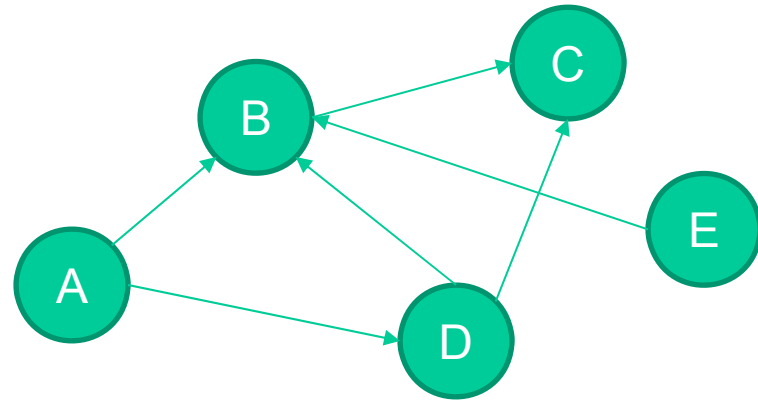
Which of the following is true?

- A. A graph can always be represented as a tree
- B. A tree can always be represented as a graph**
- C. Both A and B
- D. Neither A or B

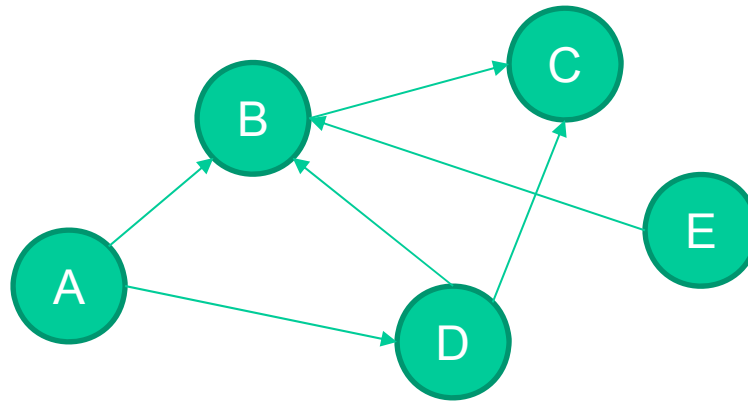
Note that trees are special cases of graphs; lists are special cases of trees.

Why Graphs?

- Organism
- Money
- Social network
- Cure cancer
-



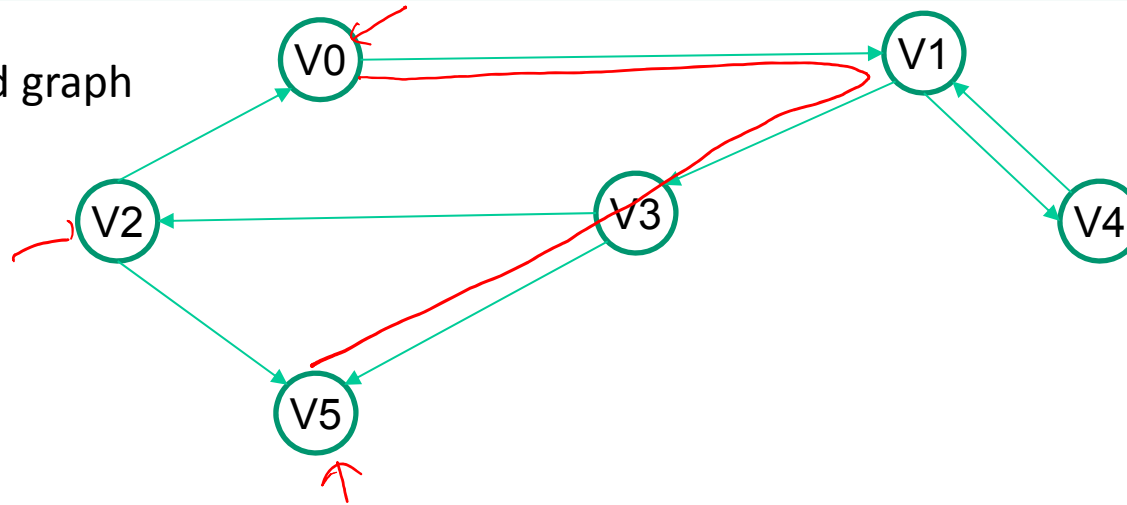
Why Graphs?



Remember: If your problem maps to a well-known graph problem, it usually means you can solve it blazingly fast!

Graphs: Example

A directed graph



$V = \{ v_0, v_1, \dots, v_5 \}$

$|V| = 6$

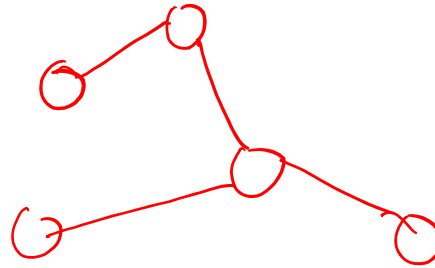
$E = \{ (0,1), (1,3), (1,4), (4,1), (3,2), (3,5), (2,5), (2,0) \}$

$|E| = 8$

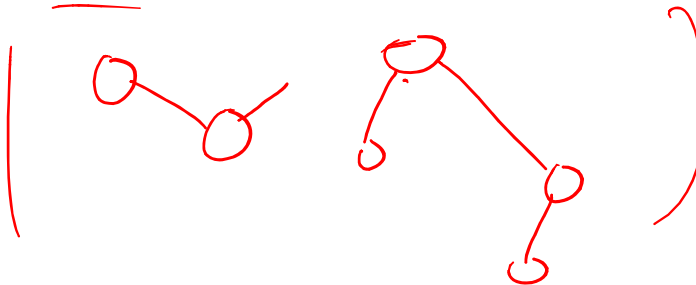
Path: $v_0 - v_1 - v_3 - v_5$

Connected, disconnected and fully connected graphs

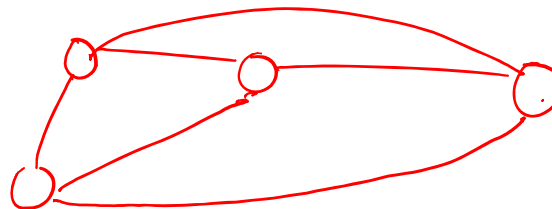
- Connected graphs:



- Disconnected graphs:



- Fully connected (complete graphs):

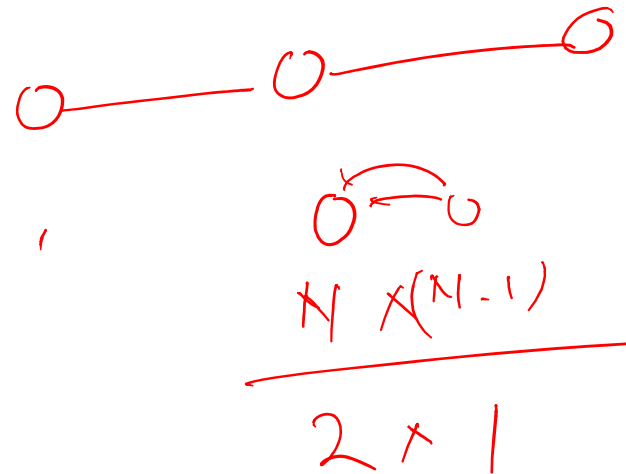


Q: What are the minimum and maximum number of edges in a undirected connected graph $G(V,E)$ with no self loops, where $N=|V|$?

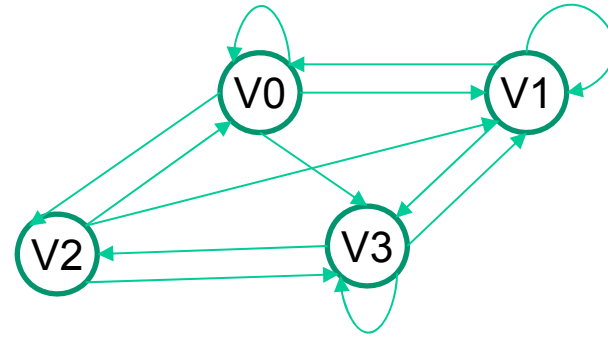
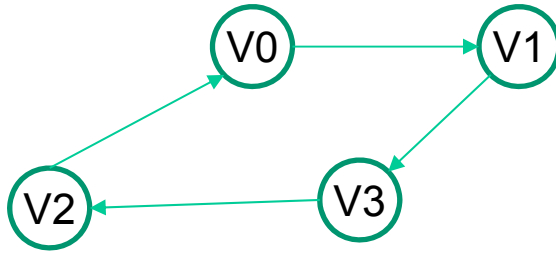
A. 0, N^2

B. N , N^2

C. $N-1$, $N(N-1)/2$

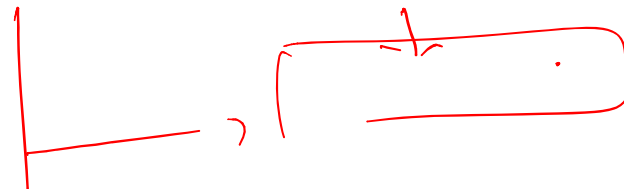


Sparse vs. Dense Graphs

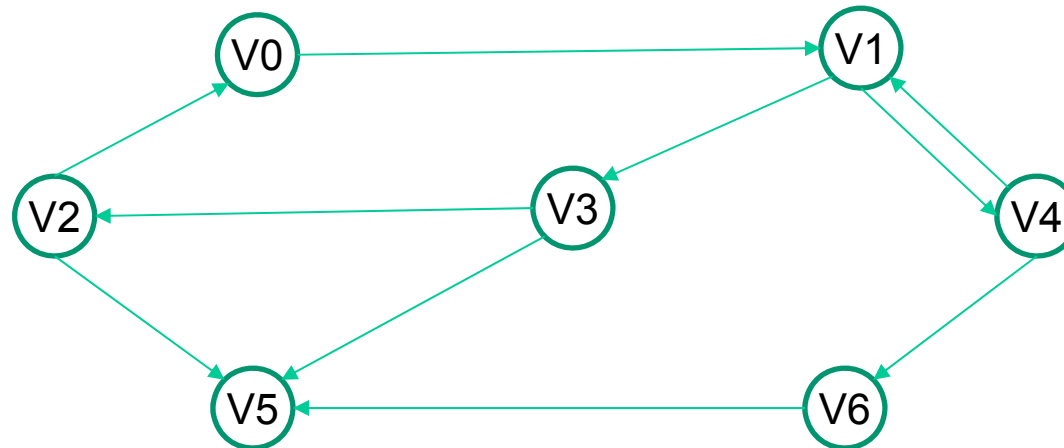


A dense graph is one where $|E|$ is “close to” $|V|^2$.
A sparse graph is one where $|E|$ is “closer to” $|V|$.

S



Representing Graphs: Adjacency Matrix

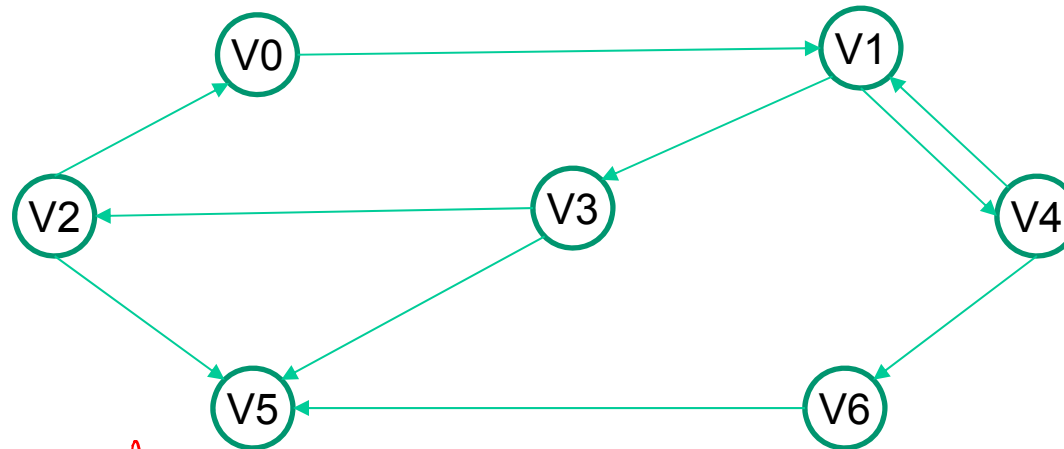


	j i						
	0	1	2	3	4	5	6
0		1					
1	1						
2							
3							
4							
5							
6							

A 2D array where each entry $[i][j]$ encodes connectivity information between i and j

- For an unweighted graph, the entry is 1 if there is an edge from i to j , 0 otherwise
- For a weighted graph, the entry is the weight of the edge from i to j , or "infinity" if there is no edge
- Note an undirected graph's adjacency matrix will be symmetrical

Representing Graphs: Adjacency Matrix



$N = |V|$

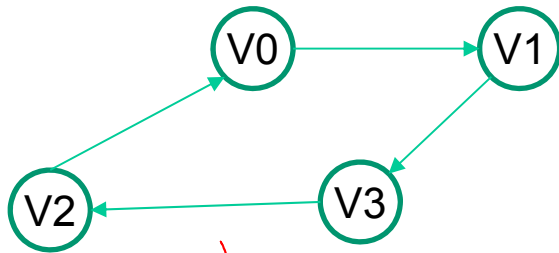
	0	1	2	3	4	5	6
0	$\checkmark$	1	$\checkmark$				
1	$\checkmark$	$\checkmark$		1	1		
2	1					1	
3			1			1	
4		1					1
5							
6	$\checkmark$	$\checkmark$				1	

How big is an adjacency matrix in terms of the number of nodes and edges (BigO, tightest bound)?

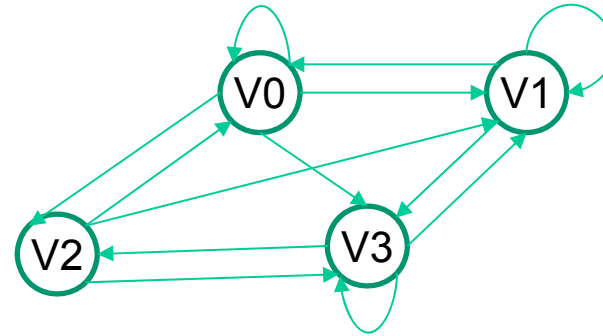
- A. $|V|$
- B. $|V| + |E|$
- C. $|V|^2$**
- D. $|E|^2$
- E. Other

When is that OK? When is it a problem?

Space efficiency of Adjacency Matrix



	0	1	2	3
0	0	1	0	0
1	0	0	0	1
2	1	0	0	0
3	0	0	1	0



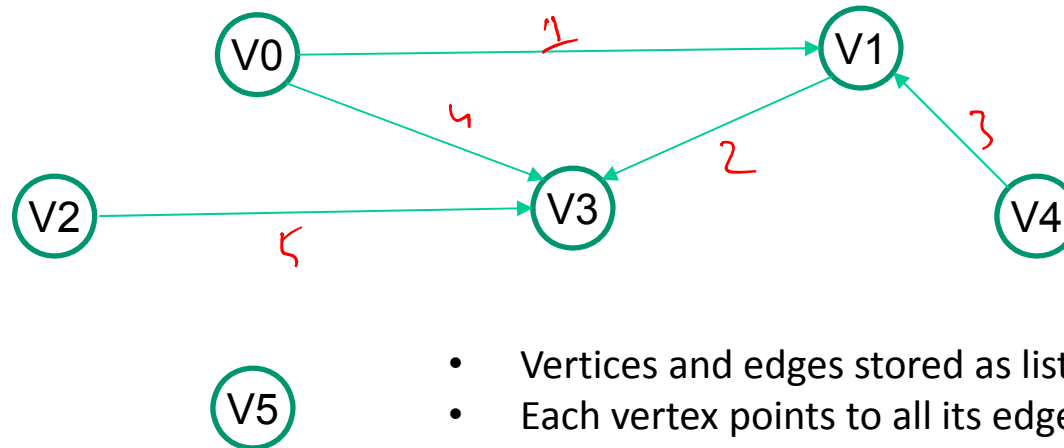
	0	1	2	3
0	1	1	1	1
1	1	1	0	1
2	1	1	0	1
3	0	1	1	1

A dense graph is one where $|E|$ is “close to” $|V|^2$.

A sparse graph is one where $|E|$ is “closer to” $|V|$.

Adjacency matrices are space inefficient for sparse graphs

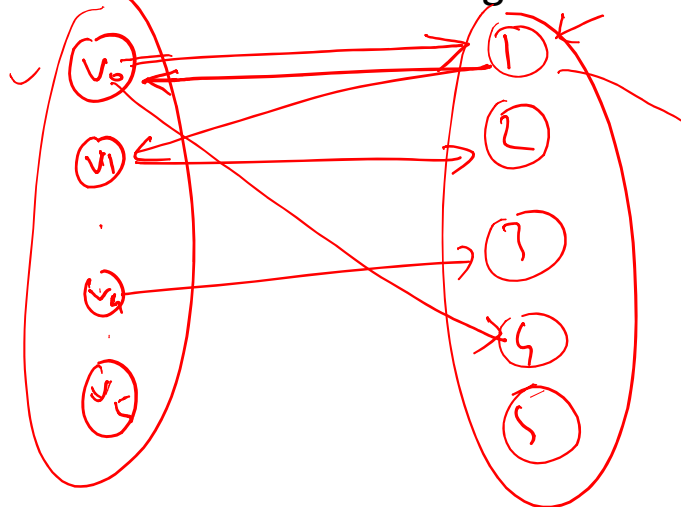
Representing Graphs: Adjacency Lists



- Vertices and edges stored as lists
- Each vertex points to all its edges
- Each edge points to the two vertices that it connects
- If the graph is directed: edge nodes differentiate between the head and tail of the connection
- If the graph is weighted edge nodes also contain weights

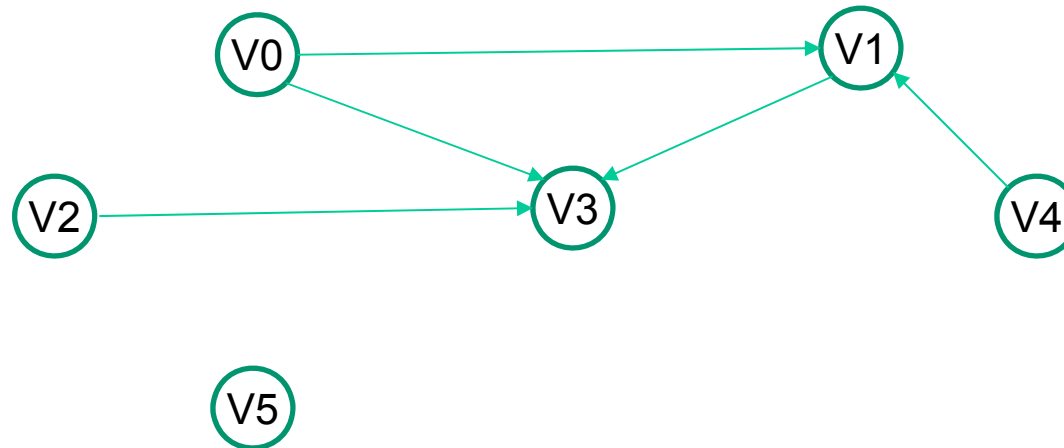
Vertex List

Edge List



Head X
Tail X -

Representing Graphs: Adjacency Lists



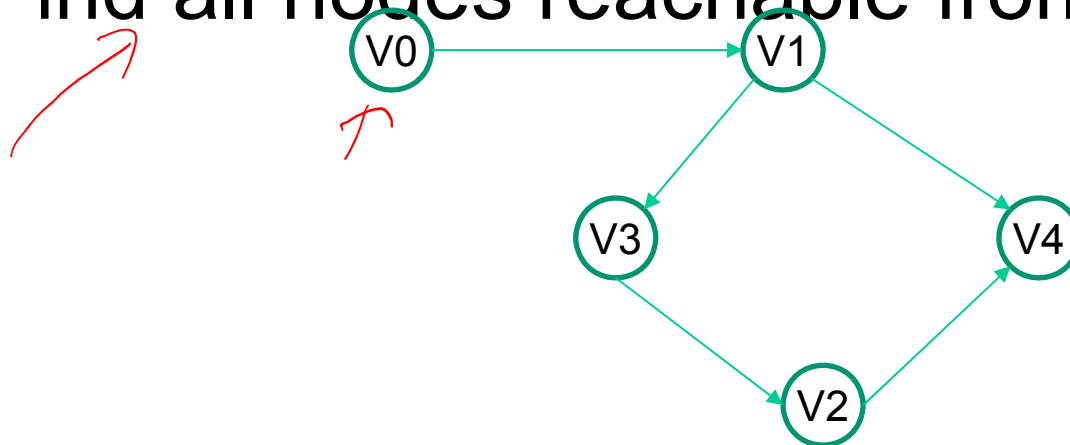
Each vertex has a list with the vertices adjacent to it.
In a weighted graph this list will include weights.

How much storage does this representation need?
(BigO, tightest bound)

- A. $|V|$
- B. $|E|$
- C. $|V| + |E|$
- D. $|V|^2$
- E. $|E|^2$

Searching a graph

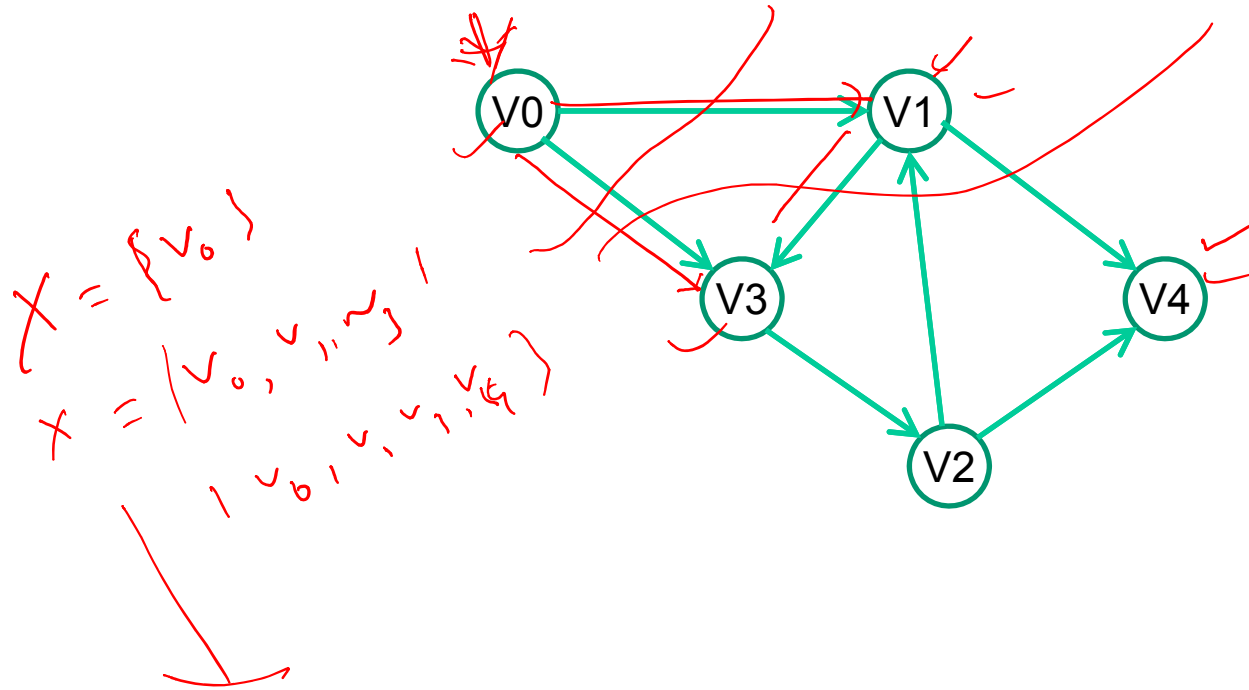
- Find if a path exists between any two nodes
- Find the shortest path between any two nodes
- Find all nodes reachable from a given node



Generic Goals:

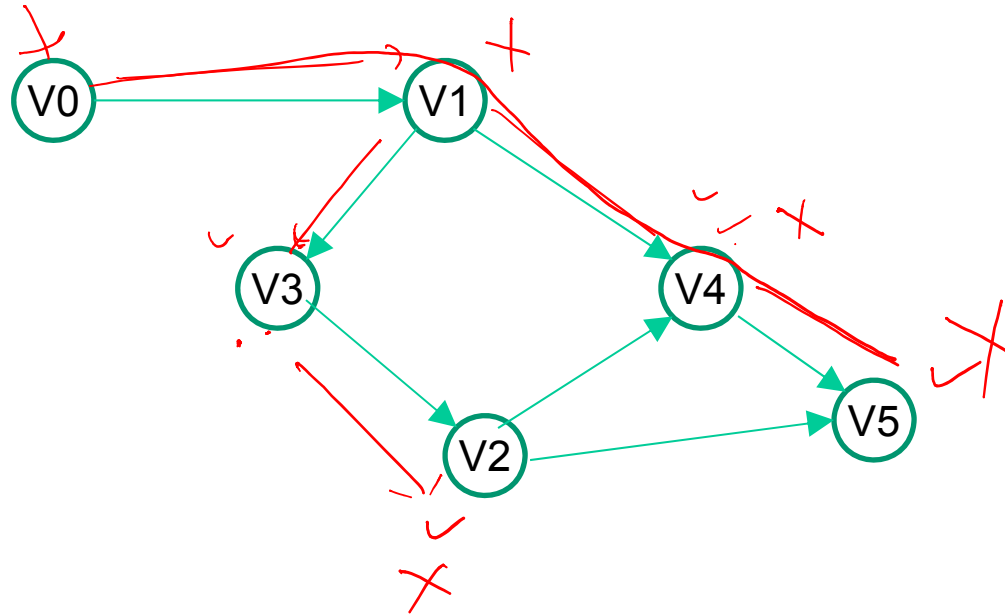
- Find everything that can be explored
- Don't explore anything twice

Generic approach to graph search



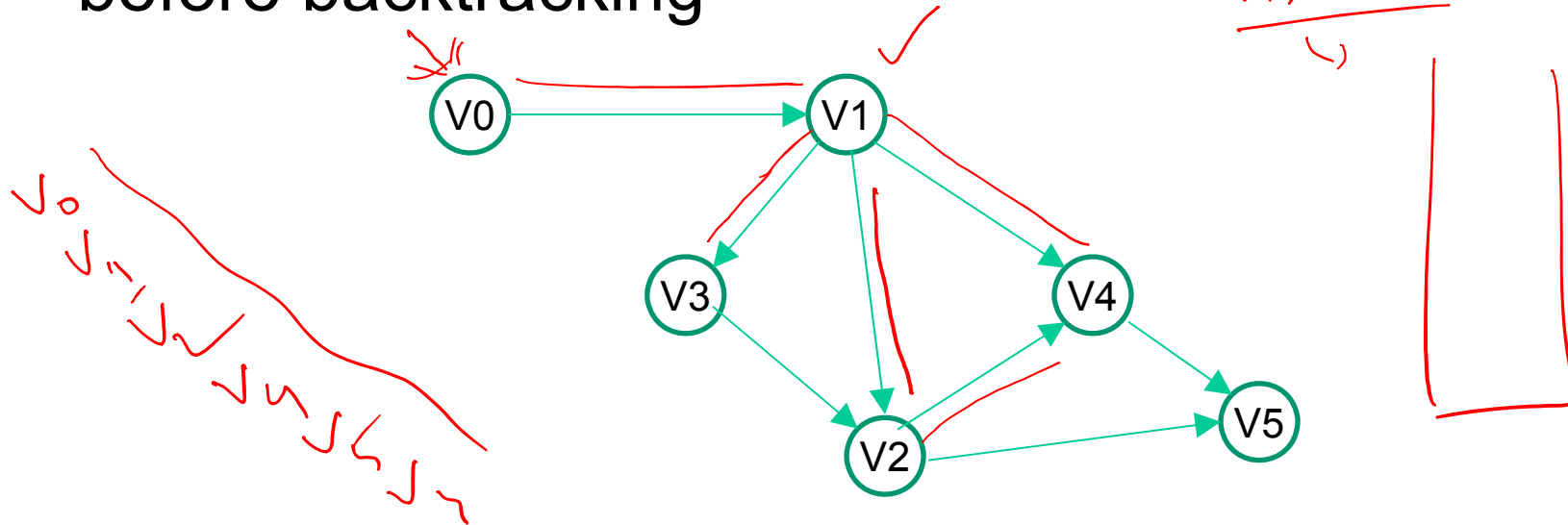
Depth First Search for Graph Traversal

- Search as far down a single path as possible before backtracking



Depth First Search for Graph Traversal

- Search as far down a single path as possible before backtracking

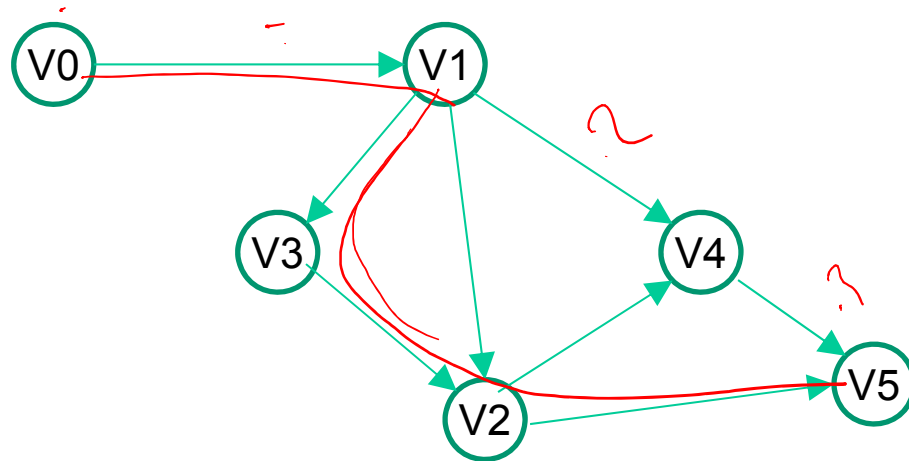


Assuming DFS chooses the lower number node to explore first, in what order does DFS visit the nodes in this graph?

- A. V0, V1, V2, V3, V4, V5
- B. V0, V1, V3, V4, V2, V5
- C. V0, V1, V3, V2, V4, V5
- ☒ D. V0, V1, V2, V4, V5, V3

Depth First Search for Graph Traversal

- Search as far down a single path as possible before backtracking



Does DFS always find the shortest path between nodes?

A. Yes

☒ B. No