CSCI-SHU 210 Data Structures

Recitation13 Worksheet Graph, DFS, BFS

Part 1: Visualizing graphs

For the following diagram, draw the 2 Graph ADTs to represent the diagram.

- Adjacency map
- Adjacency matrix

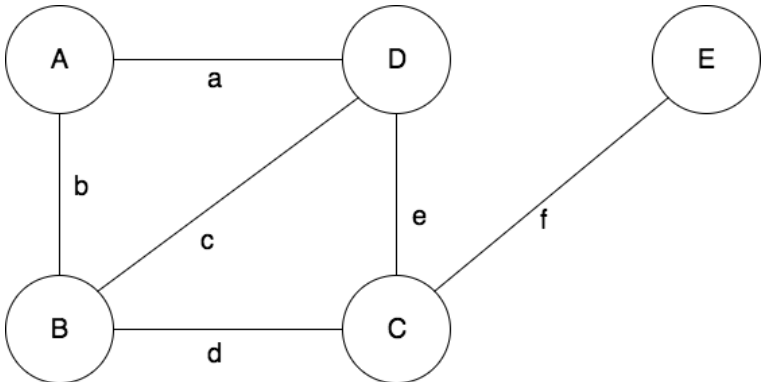


Figure 1: Unweighted, undirected graph

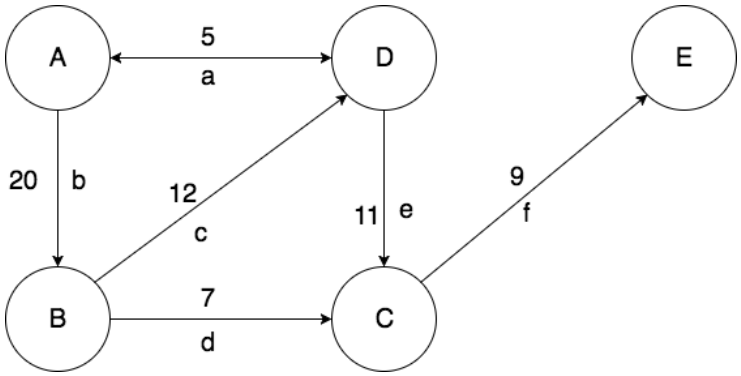


Figure 2: Weighted, directed graph

Part 2: Understand textbook graph implementation

graph.py (Adjacency map implementation)

Part 3: Using the textbook graph implementation

Your task 1: Learn how to use graph.py. Try to build the following graphs within graph.py:

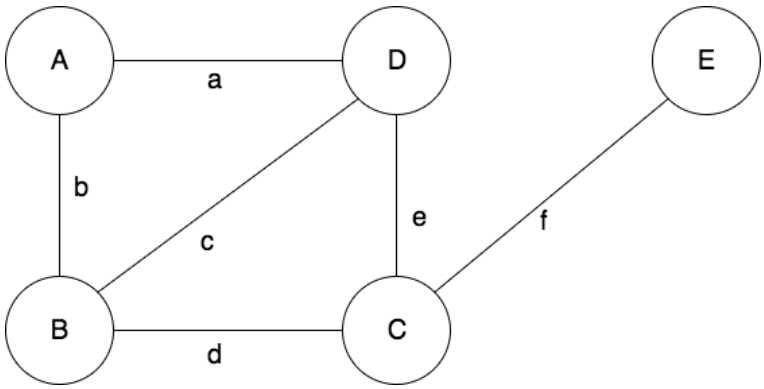


Figure 1: Unweighted, undirected graph

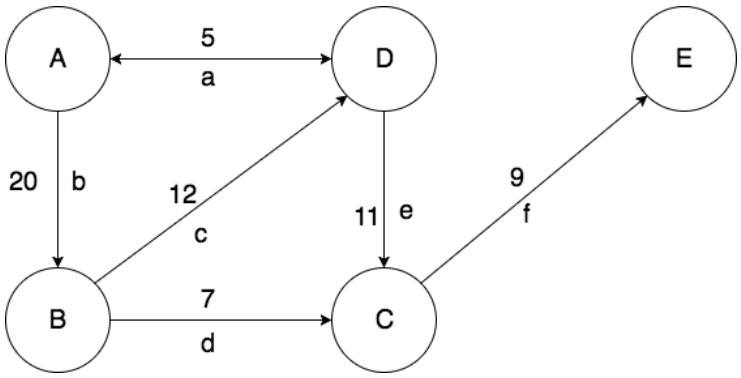


Figure 2: Weighted, directed graph

Remember, this is an adjacency map representation of Graph ADT!

Your task 2: Try to call Graph.degree(v, outgoing=True) function.

What is the degree of vertex “C” if not directed (Figure 1 graph)?

What is the degree of vertex “C” if directed (Figure 2 graph)?

Your task 3: Try to call Graph.incident_edges(v, outgoing=True) function.

What are incident_edges of vertex “C” if not directed (Figure 1 graph)?

What are incident_edges of vertex “C” if directed (Figure 2 graph)?

Part 4: Implementing graph algorithms

1. BFS
2. DFS

Your task 4: Implement BFS(g, s, discovered) function.

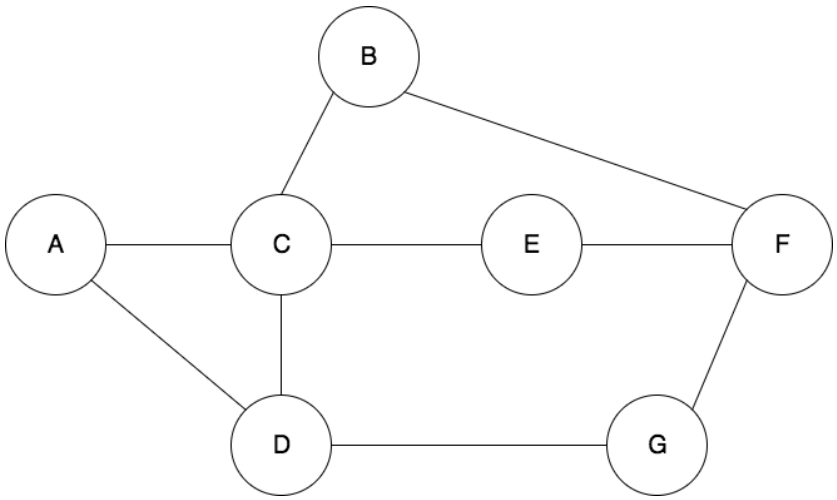
Parameter g is the graph.

Parameter s is the starting vertex.

Parameter discovered is a dictionary, you can use it to track visited vertices.

The following graph have been built.

You can use it to test your BFS result.



Suppose the BFS start from vertex A. Here's a trace for the BFS algorithm:

phase	Discovered list (Output)	current_level	next_level_list
init	A	[A]	[]
1	A C	[**A**]	[C]
2	A C D	[**A**]	[C D]
3	A C D	[C D]	[]
4	A C D B	[**C** D]	[B]
5	A C D B E	[**C** D]	[B E]
6	A C D B E G	[C **D**]	[B E G]
7	A C D B E G	[B E G]	[]
8	A C D B E G F	[**B** E G]	[F]
9	A C D B E G F	[B **E** G]	[F]
10	A C D B E G F	[B E **G**]	[F]

11	A C D B E G F	[F]	[]
12	A C D B E G F	[F]	[]
13	A C D B E G F	[]	[]

Your task 5: Implement DFS(g, s, discovered) function.

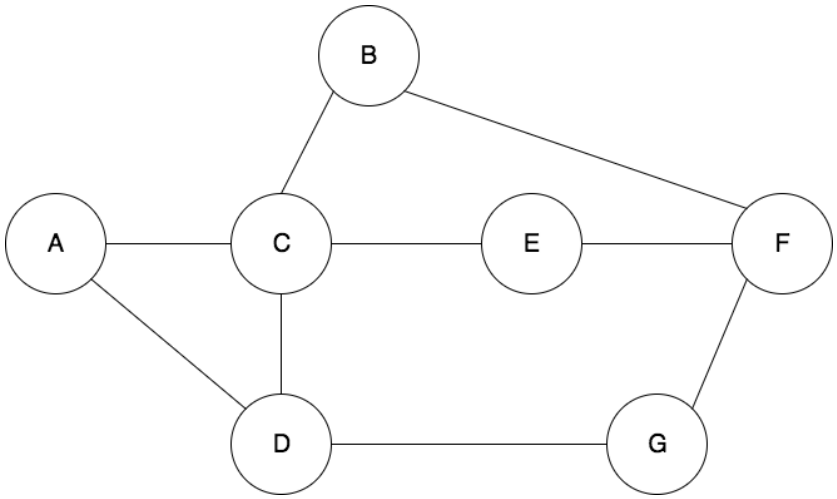
Parameter g is the graph.

Parameter s is the starting vertex.

Parameter discovered is a dictionary, you can use it to track visited vertices.

The following graph have been built.

You can use it to test your DFS result.



Suppose the DFS start from vertex A. Here's a trace for the DFS algorithm:

phase	discovered_list (Output list)	function call stack (top of stack on the left)
init	A	A
1	A C	C A
2	A C B	B C A
3	A C B F	F B C A
4	A C B F E	E F B C A
5	A C B F E G	G F B C A
6	A C B F E G D	D G F B C A