TUTORIAL 7

GRAPH TRAVERSALS

Problem: How can we visit all of the vertices in a graph?

Solution: Traversals!

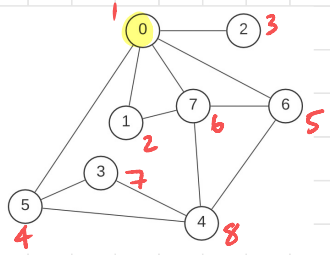
1. **Depth-first**: Pick a path and follow it until you hit a dead end. Then backtrack to find a path you haven't yet followed. Repeat until all vertices are visited.

2. **Breadth-first**: Visit all neighbouring vertices first, Then visit neighbours-of-neighbours etc. until all vertices are visited. In effect, you fan out and follow all paths one edge at a time.

Implementation: Actually incredibly similar! A DFS uses a stack to visit deeper nodes first, but a BFS uses a queue to visit close nodes first. Because DFS uses a stack, it is possible to implement it recursively instead.

So: You should use a DFS as your default traversal unless a BFS is required, because recursive DFS is easier to implement.

1.



BFS starting at 0:

Ite#	Printed	Queue
0	–	0
1	0	1 2 5 6 7
2	1	2 5 6 7
3	2	5 6 7 3 4
4	5	
5	6	
6	7	
7	3	
8	4	

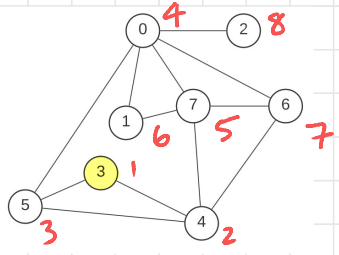
we've seen everything now, so the algo is just going to dequeue + print

Visited?

0	1	2	3	4	5	6	7
✓							
✓	✓	✓			✓	✓	✓
✓	✓	✓			✓	✓	✓
✓	✓	✓	✓	✓	✓	✓	✓

Predecessors

0	1	2	3	4	5	6	7
	0	0			0	0	0
	0	0			0	0	0
	0	0	5	5	0	0	0



DFS starting at 3:

It#	Printed	Stack (bottom → top)
0	–	3
1	3	5 4
2	4	5 7 6 5
3	5	5 7 6 0
4	0	5 7 6 7 6 2 1
5	7	5 7 6 7 6 2 7
6	1	5 7 6 7 6 2 6
7	6	5 7 6 7 6 2
8	2	5 7 6 7 6

Visited?

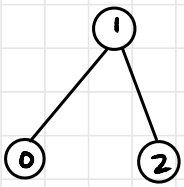
	0	1	2	3	4	5	6	7
				✓				
				✓				
				✓	✓			
				✓	✓	✓		
	✓			✓	✓	✓		
	✓	✓		✓	✓	✓		
	✓	✓		✓	✓	✓		✓
	✓	✓		✓	✓	✓	✓	✓
	✓	✓	✓	✓	✓	✓	✓	✓

Predecessors

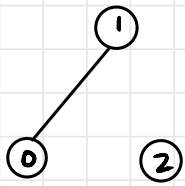
	0	1	2	3	4	5	6	7
					3	3		
					3	4	4	4
	5				3	4	4	4
	5	0	0		3	4	0	0
	5	0	0		3	4	0	1
	5	0	0		3	4	7	1
	5	0	0		3	4	7	1
	5	0	0		3	4	7	1

INTERESTING GRAPH SUBSTRUCTURES

Connected: There is a path from every vertex to every other vertex.



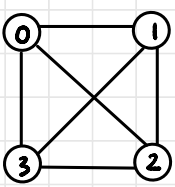
connected



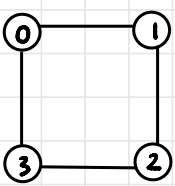
not connected

If a graph is not connected, it can be broken into multiple connected components (i.e. connected subgraphs).

Complete: If there is an edge b/w every pair of vertices.



Complete
(K_4)



not complete
(C_4)

Hamiltonian path/cycle: A path/cycle that visits each vertex in the graph exactly once.

Euler path/cycle: Like a Hamiltonian path/cycle but visiting each edge exactly once.

It is quite easy to check if a graph has an Euler path/cycle (see MATH1081), but extremely hard to check for Hamiltonian "—"!

3.

n variables $x_1, x_2, \ldots, x_n$

m equalities $x_i = x_j$ for some $1 \leq i, j \leq n$

k inequalities $x_i \neq x_j$ " "

Problem: Is this system consistent?

Solution: Think about turning the equations into a graph and then the connected components in that graph.