CS2302 Data Structures

Spring 2020

Exercises - Graphs

1. Write the function *num_vertices(G)* that receives a graph *G* represented as an adjacency matrix and returns the number of vertices in *G*.

2. Write the function *count_edges(G)* that receives a graph *G* represented as an adjacency matrix and returns the number of edges in *G*. Make sure your function works for directed and undirected graphs.

3. Write the function *highest_weight_edge(G)* that receives a weighted directed graph *G* represented as an adjacency matrix and returns a list of length 3 representing the highest-weight edge in *G* using the format [source, destination, weight] (break ties arbitrarily).

4. The out-degree of a vertex *v* in a directed graph *G=(V,E)* is the number of edges going out from *v* in *G*. Write the function *out_degrees(G)* that receives a graph *G* represented as an adjacency matrix and returns a list of length |*V*| containing the out-degrees of the vertices in *V*.

5. The in-degree if a vertex *v* in a directed graph *G=(V,E)* is the number of edges going into *v*. Write the function *in_degrees(G)* that receives a graph *G* represented as an adjacency matrix and returns a list of length |*V*| containing the in-degrees of the vertices in *V*.

6. Write a function *reverse_edges(G)* that receives a graph G represented as an adjacency matrix and reverses the direction of the edges in G.

7. Write a function *al_to_am(G)* that receives a graph G represented as an adjacency list and returns the adjacency matrix representation of G.

8. Write a function *am_to_al(G)* that receives a graph G represented as an adjacency matrix and returns the adjacency list representation of G.