CSE 250 Recitation

10/23-10/24 : Graph Representations and Traversals, PA2

Graph Representations

- Given the following edge list, what is the best algorithm to create the equivalent adjacency list?
- What is the runtime of the algorithm you came up with?

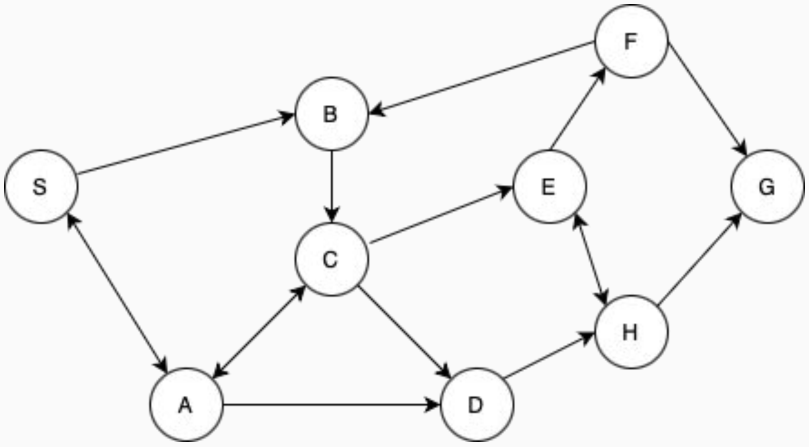
DtoH	EtoF
CtoE	BtoC
AtoD	FtoB
HtoE	StoB
StoA	CtoA
CtoD	FtoG
AtoS	AtoC
HtoG	EtoH

Graph Representations

- Note that both lists represent the same graph and both are capable of performing a graph traversal like DFS or BFS
- However one representation is far more efficient at performing a graph traversal

S	StoA, StoB
A	AtoS, AtoC, AtoD
B	BtoC
C	CtoA, CtoD, CtoE
D	DtoH
E	EtoF, EtoH
F	FtoB, FtoG
G	
H	HtoE, HtoG

Graph Traversal



1. Insert an arbitrary starting node into the [DATASTRUCTURE]
2. While the [DATASTRUCTURE] is not empty:
   a. Remove a node from the [DATASTRUCTURE]
   b. Mark the node as visited
   c. Insert all of the node’s unvisited neighbors into the [DATASTRUCTURE]

1. [DATASTRUCTURE] ⟵ Stack
2. [DATASTRUCTURE] ⟵ Queue

PA2: Implementation

- Now that testing has been completed, we can turn our focus to implementing PA2
- There are three major functions to implement in PA2:
  - computeOutgoingEdges()
  - pathWithFewestIntersections() (BFS)
  - pathWithShortestDistance() (Dijkstra's)
- If you haven't noticed already, we talked about how to implement computeOutgoingEdges() on the first slide

PA2: Implementation

- The other two functions will require you to coordinate three different data structures to perform a graph traversal and return the correct path
    - Node Scheduling Sequence
        - Queue or priority queue depending on type of graph traversal
    - Visited Set
    - Path Builder Hashmap
- We just saw how to use the scheduler sequence and visited set while doing the traversal
- With a partner, discuss how you can use a hashmap to efficiently build a path

PA2: Implementation

- For this assignment, the most efficient way to use the hashmap to build the path is to:
  - Assign the keys to be the id of an intersection (string)
  - Assign the values to be the pointer to the edge that lead to the intersection identified as the key
    - For the starting node you can have the value be “Null”
  - Once you find the goal intersection you can use your hashmap to build your path backwards
- Why is this approach better than saving the whole path as the value?