Practice Quiz 12

CSCI 136: Spring 2022

Your name: _____

This week's quizzes cover readings, handouts, labs, and lecture materials up to and including May 9nd. Answer the following questions as practice for your final exam—no graded quiz this week!

In the adjacency list representation of a graph, each vertex stores a `SinglyLinkedList` of its indicent edges. The following questions are about a (hypothetical) *adjacency vector* representation, in which each vertex stores a `Vector` of its incident edges. As in an adjacency list representation, the neighbors are not stored in any particular order. Assume in all of these questions that the graph is directed.

1. To add a new edge from `V v1` to `V v2`, we first find the `GraphListVertex` corresponding to `v1`. We then make an `Edge` object to hold the new edge, and call `addFirst()` on the `Vector` for vertex `v1`. What is the *worst-case* running time of this method call in terms of d, where d is the degree of `v1` (i.e. d is the number of neighbors of `v1`)?

   Your answer: `addFirst()` runs in $O(d)$ time

2. To add a new edge from `V v1` to `V v2`, we first find the `GraphListVertex` corresponding to `v1`. We then make an `Edge` object to hold the new edge, let's say we instead call `addLast()` on the `Vector` for vertex `v1`. What is the *worst-case* running time of this method call in terms of d, where d is the degree of `v1`?

   Your answer: `addLast()` may run in as much as $O(d)$ time, in the case that the underlying array must be extended

3. In the above two questions we discussed the ramifications of calling `addFirst()` or `addLast()` to add a new vertex to the graph that uses an adjacency vector. Which method is better to use? Briefly expain why

   Your answer: `addLast()`. Even though the worst-case running time is the same for a single operation, after adding all d objects to the vector, `addFirst()` will take $O(d^2)$ time in total, whereas `addLast()` will take $O(d)$ time.

4. What is the space usage of a graph stored using an adjacency vector? Suppose n, the number of vertices, and m, the number of edges, are both given. Answer in terms of n and m and be sure to provide a brief explanation.

   Your answer: $O(n+m)$: each vertex must be stored in the hash table ($O(n)$ space). For each vertex we have a singly linked list of all neighboring edges. Each edge appears in one such list, for $O(m)$ total space. Total: $O(n+m)$ space.