Congratulations! You passed!

Grade received 100% **Latest Submission Grade** 100% **To pass** 80% or higher

Go to next item

1. Let G = (V,E) be a simple graph consisting of a set of vertices V and a set of (undirected) edges E where each edge is a set of two vertices. Which one of the following is not a simple graph? **1 / 1 point**

- ○ G = (V = (a,b,c), E = ((a,b),(b,c),(a,c)))
- ○ G = (V = (a,b,c), E = ())
- ○ G = (V = (a,b,c), E = ((a,b)))
- ● G = (V = (a,b,c), E = ((a,b), (a,c), (b,a), (b,c), (a,c), (b,c)))

Correct
This is not a *simple* graph because the same edge between a and b appears twice, once as (a,b) and a second time as (b,a). Since these are sets, (a,b) == (b,a).

2. For a simple graph with n vertices, what is the worst case (largest possible) for the number of edges? **1 / 1 point**

- ● O(n^2)
- ○ O(n log n)
- ○ O(2^n)
- ○ O(n)

Correct

Recall that the adjacency matrix has one entry per edge in its upper triangular portion. There are n^2 elements in the n x n adjacency matrix, and about 1/2 n^2 elements in its upper triangular portion, and O(1/2 n^2) == O(n^2).

3. Which graph representation has a better worst-case storage complexity than the others for storing a simple graph of n vertices?

1 / 1 point

- ○ Edge List
- ○ Adjacency Matrix
- ○ Adjacency List
- ◉ All three graph representations have the same worst-space storage complexity for a simple graph of n nodes.

Correct

All three require O(n^2) storage in the worst case. The adjacency matrix requires O(n^2) space to store at least the upper-triangular portion of the n x n matrix. Both the edge list and adjacency list representations require O(n + m) storage, but in the worst case m is proportional to n^2 and O(n + n^2) = O(n^2).

4. Suppose you have a rapid data feed that requires you to add new data point vertices quickly to a graph representation. Which graph representation would you NOT want to utilize?

1 / 1 point

- ○ Edge List
- ◉ Adjacency Matrix
- ○ Adjacency List
- ○ All three graph representations have the same time complexity for adding vertices to a simple graph.

Correct

The adjacency matrix requires linear time, O(n), to add a vertex because the addition requires new entries to be placed in a new row and a new column of the matrix, and there are n elements in the new row and n elements in the new column. This means that as the number of vertices grows in the graph, it will take longer to add a new vertex, which is not a very good choice when processing a data feed.

5. Suppose you have a rapid data feed that requires you to remove existing data point vertices (and any of their edges to other vertices) quickly to a graph representation. Which graph representation would you WANT to utilize?

1 / 1 point

- ○ Edge List
- ○ Adjacency Matrix
- ◉ Adjacency List
- ○ All three representations have the same time complexity for removing a vertex from a simple graph of n vertices.

Correct

Since the adjacency list has a list of the edges the removed vertex shares with other vertices, it only needs time proportional to the degree of the removed vertex. In the worst case, that vertex could be connected to all of the other vertices and so require O(n) time, but in the typical case the degree will be less and the adjacency list is a better choice than the adjacency matrix.

6. Suppose you want to implement a function called neighbors(v) that returns the list of vertices that share an edge with vertex v. Which representation would be the better choice for implementing this neighbors() function?

1 / 1 point

- ○ Edge List
- ○ Adjacency Matrix
- ◉ Adjacency List
- ○ All three representations result in the same time complexity for the neighbor() function.

Correct

The adjacency list requires a simple walk through the list of pointers to adjacent edges to find the neighboring vertices. This representation has an "output sensitive" running time meaning it runs as fast as possible based on the minimum amount of time needed to output the result.

7. Suppose you want to implement a function called neighborsQ(v1,v2) that returns true only if vertices v1 and v2 share an edge. Which representation would be the better choice for implementing this neighborsQ() function?

1 / 1 point

- ( ) Edge List
- (•) Adjacency Matrix
- ( ) Adjacency List
- ( ) All three representations support the same time complexity for implementing the neighborQ() function.

Correct
The neighborsQ(v1,v2) function can simply lookup the appropriate v1,v2 entry in the adjacency matrix, which takes constant O(1) time. This representation supports the fastest method for implementing this query.

8. Which of these edge lists has a vertex of the highest degree?

1 / 1 point

- ( ) (a, b), (a, c), (a, d), (b, d)
- ( ) (a, c), (e, g), (c, e), (g, a)
- (•) (a,b), (b, c), (d, b), (g, b)
- ( ) (d,b), (g,a), (h,f), (c, e)

Correct
Vertex b has degree four.

9. Which adjacency matrix corresponds to the edge list: (1,2), (2,3), (3,4), (1,4) (where the rows/columns of the adjacency matrix follow the same order as the vertex indices)? **1 / 1 point**

○

0	0	1	1
	0	1	1
		0	0
			0

○

0	1	1	0
	0	1	1
		0	1
			0

○

1	0	0	1
	1	0	0
		1	0
			1

◉

0	1	0	1
	0	1	0
		0	1
			0

✓ **Correct**

10. Which graph representation would be the best choice for implementing a procedure that only needs to build a graph from a stream of events.

1 / 1 point

- (●) Edge List
- ( ) Adjacency Matrix
- ( ) Adjacency List
- ( ) All three representations would share the same storage and time complexity for the procedure.

Correct

The Edge List performs worse in general than the Adjacency Matrix and the Adjacency List representations, but it is much simpler and easier to implement. It also takes less space than the alternatives, and can insert vertices and edges in constant time. The adjacency list can also insert vertices and edges in constant time, but if those are the only operations needed, then one need not waste space and additional code on building the adjacency list on top of the edge list.