CPSC 319 Assignment 5: Graphs

Name:Shanzi Ye

Instructor:Bajwa, Sohaib

Submission Date: August,8,2022

Q1. (1 Marks) Consider that you have implemented your graphs using adjacency matrix, how do you tell, by looking at its adjacency matrix, how many edges there are in an undirected graph?

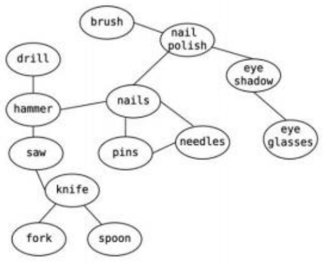
Answer:

We need to count the total number of 1's in the adjacency matrix and divide it by two to get the total number of edges in an undirected graph.

For example, if there are ten 1's in the adjacency matrix,the number of edges is five.

Q2. (1 Marks) The below graph powers an e-commerce store's recommendation engine. Each vertex represents a product available on the store's website. The edges connect each product to other "similar" products the site will recommend to the user when browsing a particular item.

If the user is browsing "nails," what other products will be recommended to the user?



Answer:

Pins, needles, hammer, nail polish

Q3. (2 Marks) Perform a depth first search on the graph starting from Vertex A. You can assume that when given the choice to visit multiple adjacent vertices, first visit the node that is earliest in the alphabet.

Answer:

A B E J F O C G K D H L M I N P

Q4. (2 Marks) Perform a breadth first search on the graph starting from Vertex A. You can assume that when given the choice to visit multiple adjacent vertices, first visit the node that is earliest in the alphabet.

Answer:

A B C D E F G H I J K L M N O P

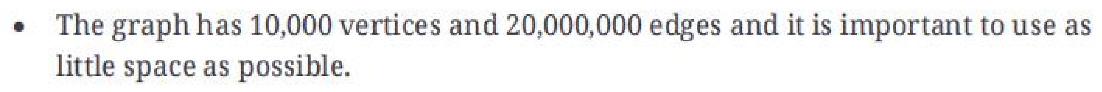
Q5: (9 Marks) Would you use the adjacency matrix structure or the adjacency list structure in each of the following cases? Justify your choice.

- The graph has 10,000 vertices and 20,000 edges and it is important to use as little space as possible.

Answer:

I would like to use adjacency list instead of adjacency matrix. According to the lecture notes, adjacency list is great for sparse edge graphs and it can save a lot space for edges that don't exist. In this case, there are 10000 vertices but only 20000 edges in the graph, indicating that it is a sparse graph. That's why we need to use adjacency list. However, if we use

adjacency matrix, the computer needs approximately 100000000(10000*10000) units of space. The computer has to create a 10000*10000 matrix with lots of 0's in it,because there are only 20000 edges. Lots of memory will be wasted if you use adjacency matrix.



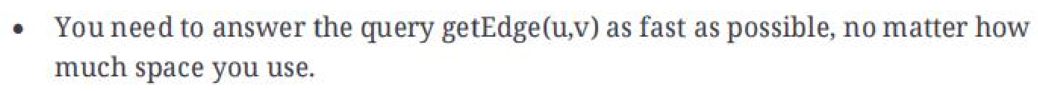
- The graph has 10,000 vertices and 20,000,000 edges and it is important to use as little space as possible.

Answer: **It depends.**

If you use adjacency matrix, the computer needs approximately 100000000(10000*10000) units of space.

If you use adjacency list, the computer needs approximately 20010000 units of space. We can see that both methods work great and we can't find the better option between them based on the requirement.

if you want to search for a edge, you should use adjacency matrix as it is great for adjacency check. However, if you want to add a node or delete a node, you should use adjacency list.



- You need to answer the query getEdge(u,v) as fast as possible, no matter how much space you use.

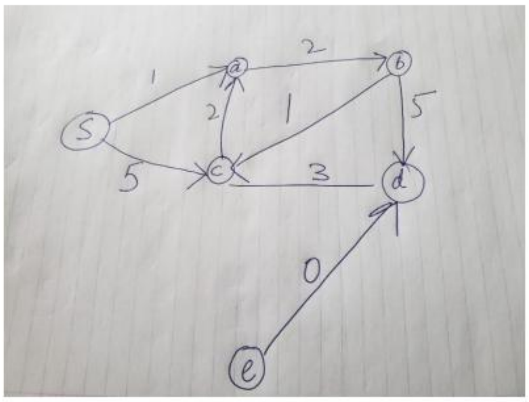
Answer:

Since we don’t need to care about the usage of memory, we should use adjacency matrix . For example, given the two vertices i and j, we have to find the existing edge between them. According to the lecture notes, adjacency matrix is good for adjacency check and its time complexity is O(1). So the time complexity of searching a edge between i and j is O(1).

If we want to use adjacency list to find a specific edge, we have to traverse the list to find the adjacent vertices, which is less efficient. The time complexity of searching a edge using adjacency list is O(v).

In conclusion, we should use adjacency matrix in case 3.

Q6. (5 Marks) Consider the below weighted-directed graph. Identify the shortest path by following Dijkstra algorithm. The starting source is labelled as s.



Answer:

Step1: v is vertex s.

	to be checked	currDist	pred
s	f	0	\
a	t	1	s
b	t	∞	\
c	t	5	s
d	t	∞	\
e	t	∞	\

Step2: v is vertex a.

	to be checked	currDist	pred
s	f	0	\
a	f	1	s
b	t	3	a
c	t	5	s
d	t	∞	\
e	t	∞	\

Step3: v is vertex b.

	to be checked	currDist	pred
s	f	0	\
a	f	1	s
b	f	3	a
c	t	4	b
d	t	8	b
e	t	∞	\

Step4: v is vertex c.

	to be checked	currDist	pred
s	f	0	\
a	f	1	s
b	f	3	a
c	f	4	b
d	t	7	c
e	t	∞	\

Step5: v is vertex d.

	to be checked	currDist	pred
s	f	0	\
a	f	1	s
b	f	3	a
c	f	4	b
d	f	7	c
e	t	∞	\

Step5: v is vertex e.

	to be checked	currDist	pred
s	f	0	\
a	f	1	s
b	f	3	a
c	f	4	b
d	f	7	c
e	f	∞	\

Conclusion:

The shortest path from s to s is 0.

The shortest path from s to a is 1.

The shortest path from s to b is 3.

The shortest path from s to c is 4.

The shortest path from s to d is 7.

The shortest path from a to e doesn’t exist. Vertex e is unreachable from s.

Citation:

https://stackoverflow.com/questions/2218322/what-is-better-adjacency-lists-or-adjacency-matrices-for-graph-problems-in-c

https://www.geeksforgeeks.org/comparison-between-adjacency-list-and-adjacency-matrix-representation-of-graph/