672115014 Nattikorn Sae-sue

Tutorial 7 Graph

Basic Concepts

1. **What is a graph in data structures? How is it different from a tree?**

   Graph Data Structure is a collection of vertices connected by edges. It's used to represent relationships between different entities that can have cycles and disconnect components. Graph has 2 types: A directed graph and an undirected graph. While Tree is a data structure representing hierarchy structure consisting of nodes connected by edges cannot have cycles and all nodes must be connected with exactly one edge for two nodes.

2. **What are the two main ways to represent a graph in memory? Briefly explain each.**
   1. **Adjacency Matrix** is a two-dimensional array (n*n). If the edge (v_i, v_j) is in E(G), adj_mat[i][j]=1. If there is no such edge in E(G), adj_mat[i][j]=0.
   2. **Adjacency Lists** is a data structure used to represent a graph where each node in the graph stores a list of its neighboring vertices.

3. **What is the difference between a directed and an undirected graph? Give an example of each.**

   A directed graph has edges with direction that indicate a two-ways relationship (in-out). An Example of a directed graph is

   An undirected graph has edges with no direction.

   An Example of an undirected graph is Social Networks.

4. **What is a weighted graph? Where might weighted graphs be used in real-world applications?**
   A weighted graph is a special type of graph where the edges are assigned some weights which represent relative measure units such as cost, distance, time.
   Some examples of a weighted graph application are Artificial Intelligence for decision-making processes, and Transportation networks to figure out which part takes the least time, or the path with the least overall distance.

5. **Explain the difference between a connected graph and a disconnected graph.**
   A connected graph is a graph that is connected if any two vertices of the graph are connected by a path.

A disconnected graph is a graph with at least two vertices of the graph that are not connected by a path and has at least two subgraph components that are separated from each other.

6. **What is the difference between a cyclic and an acyclic graph? Give one example of where each might be used.**
   A cyclic graph is a graph that contains at least one cycle (path that begins and ends at the same node, without passing through any other node twice).
   An example of a cyclic graph is circuit designs.
   An acyclic graph is a graph that has no cycle.
   An example of an acyclic graph is family trees.

672115014 Nattikorn Sae-sue

Graph Traversal & Algorithms

1. **How does Breadth-First Search (BFS) traverse a graph? What data structure does it use?**
   Breadth-First Search uses queue (FIFO).
   1. Start from the source node and mark it as visited.
   2. Enqueue the source node into a queue.
   3. While the queue is not empty:
      a. Dequeue a node from the front of the queue.
      b. Process the node.
      c. Enqueue all unvisited adjacent nodes and mark them as visited.
   4. Repeat the process until all reachable nodes are visited.

2. **How does Depth-First Search (DFS) traverse a graph? What data structure does it use?**
   Depth-First Search uses stack (LIFO).
   For each vertex u:
   1. Mark u as discovered.
   2. Record the discovery time of u.
   3. Explore each adjacent vertex v:
      a. If v is not marked, recursively call DFS-Visit on v.
   4. Once all adjacent vertices have been explored, mark u as black.
   5. Record the finishing time of u.

3. **Which graph traversal algorithm is better for finding the shortest path in an unweighted graph, BFS or DFS? Explain why.**
   Breadth-First Search is better for finding the shortest path in an unweighted graph because it explores step by step, level by level. It finds the shortest way to reach a place first. While DFS goes deep first, so it might take a long time before finding the shortest one.

4. **Write a Java program to** create an undirected graph **using an** adjacency list **and print its connections.**

Input	**Output**
0 - 1 0 - 2 1 - 2 1 - 3 2 – 4	Adjacency List of the Graph: 0 -> 1 2 1 -> 0 2 3 2 -> 0 1 4 3 -> 1 4 -> 2

672115014 Nattikorn Sae-sue

The program is on my GitHub: https://github.com/collapseeee/AdjacencyList-Tutorial

Or can download from the assignment submission with this pdf.