Data Structures

Topic: Graphs

By

Ravi Kant Sahu

Asst. Professor,

Lovely Professional University, Punjab

Contents

Introduction

- A Graph G is a collection of:

1. A set V of elements called Nodes or Vertices.
2. A set E of Edges such that each edge e in E is identified with a unique pair [u, v] of nodes in V, denoted by e = [u, v].

$$G = (V, E)$$

- Nodes u and v are called end points of edge e and also known as adjacent nodes or neighbors.

Basic Terminology

- Degree of a node: Degree of a node, deg(u), is the number of edges containing u.

- If deg(u) = 0, then node u is called Isolated Node.

- A Path P of length n from node u to a node v is defined as a sequence of n+1 nodes.

$$P = (v_0, v_1, v_2, \ldots, v_n)$$

Path

- **Simple Path:** The path is said to be simple is all the nodes are distinct.

- **Closed Path:** A path is said to be closed if first and last node are same i.e. $v_0 = v_{n.}$

- **Cycle:** A cycle is a closed simple path with length 3 or more.
- A cycle with length k is called k-cycle.

Graph

- **Connected Graph:** A graph G is connected iff there is a simple path between any two nodes in G.

- **Complete Graph:** A graph G is said to be complete if every node *u* in G is adjacent to every other node v in G. i.e. each node u is directly connected to all other nodes v in Graph G.

- A complete graph with *n* nodes will have $n(n-1)/2$ edges.

Labeled Graphs...

- A graph G is said to be labeled is its edges are assigned data.

- Weighted Graph: Graph G is said to be weighted if each edge e is assigned a non-negative numerical value *w(e)* called the weight or length of edge.

- If no other information about weights are given in a graph, then assume the weight w(e) = 1 for each edge.

Multi-Graph

- Multiple Edges: Distinct edges *e* and *e'* are called multiple edges if they connect the same endpoints,

  i.e. if e = [u, v] and e' = [u, v].

- Loops: An edge *e* is called a loop if it has identical endpoints ,

  i.e. if e = [u, u] .

- Note: Definition of a graph does not allow any loop or multiple edge in Graph.

- A Graph with loops or multiple edges is called a Multi-graph.

Degree of Graph

- **Outdegree:** Outdegree of a node u in G is the number of edges beginning at u.

- **Indegree:** Indegree of a node u in G is the number of edges ending at u.

- **Source:** A node u is called a source if it has a positive outdegree but zero indegree.

- **Sink:** A node u is called a sink if it has a positive indegree but zero outdegree.

Sequential Representation

- There are two ways of representing a graph in memory:
  - Sequential Representation
    - Adjacency Matrix
    - Path Matrix
  - Linked Representation

Adjacency Matrix

- Let G is a simple directed graph with m nodes and the nodes of G have been ordered and are called $v_1, v_2, \ldots, v_m$.

- The adjacency matrix $A = a_{ij}$ of graph G is the m×m matrix defined as below:

$a_{ij} = 1$, if v_i is adjacent to v_j, i.e. if there is an edge (v_i, v_j)

$a_{ij} = 0$, otherwise

- Suppose G is an undirected graph, then the adjacency matrix A of G will be a symmetric matrix i.e. one in which $a_{ij} = a_{ji}$.

Adjacency Matrix

- Let A be the adjacency matrix of a graph G. Then $a_K(i, j)$, the ij entry in the matrix A^K, gives the number of paths of length K from v_i to v_j.

Path Matrix

- Let G is a simple directed graph with m nodes $v_1, v_2, \ldots, v_m$.

- The Path matrix or Reachability matrix $P = p_{ij}$ of graph G is the m×m matrix defined as below:

$p_{ij} = 1$, if there is a path from v_i to v_j

$p_{ij} = 0$, otherwise

Path Matrix...

- Let A be the adjacency matrix and P be the path matrix of a diagraph G. Then $P_{ij} = 1$, iff there is a non-zero number in the ij entry of the matrix

$$B_m = A + A^2 + A^3 + \ldots + A^m$$

- Path matrix is obtained by replacing the non-zero entries in B_m by 1.

- Graph G is strongly connected iff path matrix P of G has no zero entries.

Warshall's Algorithm: Path Matrix

- A directed graph G with M nodes is maintained in memory by its adjacency matrix A. This algorithm finds the path matrix P of the graph G.

1. Repeat for i, j = 1, 2,..., M:
   If A[i, j] = 0, then: Set P[i, j] = 0.
   Else: Set P[i, j] = 1.
2. Repeat Step 3 and 4 for k = 1, 2, ... , M:
3. Repeat Step 4 for i = 1, 2, ... , M:
4. Repeat for j = 1, 2, ... , M:
   Set $P[i, j] = P[i, j] \vee (P[i, k] \wedge P[k, j])$
   [End of Step 4 Loop.]
   [End of Step 3 Loop.]
   [End of Step 2 Loop.]
5. Exit.

Work Space

Floyd-Warshall Algorithm: Shortest Path

Floyd–Warshall algorithm is an algorithm for finding shortest paths in a weighted graph with positive or negative edge weights.

1. Repeat for i, j = 1, 2,..., M:
   IF: W[i, j] = 0, then Set Q[i, j] = ∞.
   Else: Set Q[i, j] = W[i, j].
2. Repeat Step 3 and 4 for k = 1, 2, ... , M:
3. Repeat Step 4 for i = 1, 2, ... , M:
4. Repeat for j = 1, 2, ... , M:
   Set Q [i, j] = Min (Q[i, j] , (Q[i, k] + Q[k, j]))
   [End of Step 4 Loop.]
   [End of Step 3 Loop.]
   [End of Step 2 Loop.]
5. Exit.

Work Space

Graph Traversal

- There are two different ways of Traversing a Graph.
    - Breadth First Search (BFS) : uses Queue
    - Depth First Search (DFS) : uses Stack

- During Traversal, each node N of G will be in one of the three states:
    - STATUS = 1: Ready State (initial state)
    - STATUS = 2: Waiting State (waiting in Queue/Stack)
    - STATUS = 3: Processed State

Breadth First Search

1. Initialize all nodes to Ready State (STATUS = 1).
2. Put the statrting node A in Queue and change its status to Waiting State (STATUS = 2).
3. Repeat step 4 and 5 until Queue is empty:
4. Remove the front node N of the Queue. Process N and set the status of N to STATUS=3.
5. Add to the Rear of Queue all the neighbors of N that are in STATUS = 1. and change their status to STATUS = 2.

   [End of step 3 Loop.]
6. Exit.

Depth First Search

1. Initialize all nodes to Ready State (STATUS = 1).
2. Push the statrting node A onto STACK and change its status to Waiting State (STATUS = 2).
3. Repeat step 4 and 5 until STACK is empty:
4. POP the TOP node N from the STACK.
   Process N and set the status of N to STATUS=3.
5. PUSH onto STACK all the neighbors of N
   that are in STATUS = 1, and change their status to
   STATUS = 2.

   [End of step 3 Loop.]
6. Exit.

Questions