GraphsIntro

August 7, 2020

1 Graphs

https://opendsa-server.cs.vt.edu/ODSA/Books/CS3/html/GraphIntro.html

1.1 Table of Contents

- **Section ??**
- **Section ??**
- **Section ??**
- **Section ??**

1.2 Introduction

- most flexible and important data structure
- consists of a set of nodes (vertices), and a set of edges
  - each edge connects two nodes
- trees and lists can be be viewed as special cases of graphs
  Figure of a graph

1.3 Graphs Vs Trees

- https://freefeast.info/difference-between/difference-between-trees-and-graphs-trees-vs-graphs/
- Tree is a restricted form of graph which is minimally connected having only one path between any two vertices
- Trees have direction (parent/child relationships) without cycles (Directed Acyclic Graph, DAG)
  - a child can have only one parent
- Graphs can have more than one edges connecting vertices
  - more than one path
  - can have uni-directional or bi-directional paths (edges) between nodes

1.4 Applications

- used to model both real-world systems and abstract problems, e.g.:
- modeling connectivity in computer networks
- representing maps as a set of locations with distances between locations (GPS shortest route finder)
- modeling flow capacities in transportation networks (finding bottlenecks)

- modeling a path from a starting condition to a goal condition (used in AI and video games)
- modeling relationships such as family trees, social networks, scientific taxonomies

1.5 Terminologies

- **graph** $G = (V, E)$ consists of a set of **vertices** V and a set of **edges** E
  - each edge in E is a connection between a pair of vertices in V
- $\|V\|$ - number of vertices
- $\|E\|$ - the number of edges
  - $\|E\|$ can range from zero to a maximum of $\|V^2\| - \|V\|$
- **adjacent** vertices with a direct connection is written (a, b)
  - a is adjacent to b and vice-versa
- **path** - sequence of vertices $v_1, v_2, ..., v_n$ with length $n-1$ if there's path from v_1 to v_n
- **simple path** - all vertices on its path are unique
  - in left figure, path 0->1->3 is a simple path
  - in middle figure, the path 0->1->3->2->4->1 is NOT a simple path
- **length** of a path is the number of edges it contains
  - in left figure, length of the path 0->1->3 is 2
- **cycle** is a path of length three or more that connects some vertex v_i to itself
  - in the right figure, the path 1->3->2->4->1 is a cycle
- **simple cycle** - simple path except that the first and last vertices are the same
  - in the right figure, the path 1->3->2->4->1 is a simple cycle

1.6 Types of Graphs

1.6.1 Undirected graph

- graph (a) whose edges are not directed is called undirected graph
- **degree** the number of edges a node has
  - e.g., the degree of center node in (a) is three

1.6.2 Directed graph (digraph)

- graph (b) whose edges are directed from one vertex to another
- **out degree** of a vertex is the number of edges going out from it
  - in (c) above, out degree of 1 is one
- **in degree** of a vertex is the number of edges coming in to it
  - in (c) above, the in degree of vertex 1 is two

1.6.3 Weighted graph (labeled graph)

- graph (c) whose edges have associated weight (cost) or simply some labels
- weighted graphs can be either directed or undirected

1.6.4 Connected graph

- an undirected graph which has at least one path from any vertex to any other

1.6.5 Sparse graph

- graph with relatively few edges
  Sparse graph

1.6.6 Dense graph

- graph with relatively many edges
  Dense and complete graph

1.6.7 Complete graph

- graph with all possible edges (see above figure)
- no. of edges in a complete graph = $\frac{\|V\|(\|V\|-1)}{2}$

1.6.8 Subgraph

- a subgraph G_1 is part of of a graph G (with vertices and edges appearing exactly as in the the graph G
- e.g., $G_1 = $ nodes $\{0, 2, 3\}$ and edges $\{(0, 2), (2, 3), (0, 3)\}$
  Subgraph connected with red edges is clique

1.6.9 Clique

- a complete subgraph where all vertices are interconnected
- e.g. the red colored subgraph above

1.6.10 Acyclic graph

- graph without cycles
- e.g. see figure b below

1.6.11 Directed acyclic graph (DAG)

- directed graph without cycles
- e.g. see figure a below

1.6.12 Free tree

- connected, undirected graph with no simple cycles
  - connected acyclic graph with $\|V\| - 1$ edges
  - e.g., figure (b) acyclic graph above is free tree

1.7 Graph Representations

- two common methods
- adjacency matrix and adjacency list

1.7.1 Adjacency matrix

- a graph of $\|V\| \times \|V\|$ matrix (2-D array)
- vertices are typically labeled from 0 through $\|V\| - 1$
- row i of the matrix contains entries for vertex v_i
- column j in row i is marked if there's an edge from v_i to v_j
  - matrix is initialized with 0s
- space requirement is $\Theta(\|V\|^2)$

1.7.2 Adjacency list

- array/vector of linked lists
- the length of array is $\|V\|$, with index i storing a pointer to the linked list of edges for vertex v_i
  - each linked list represents the edges of the vertices that are adjacent to vertex v_i
- space requirement is $\Theta(\|V\| + \|E\|)$

1.7.3 Directed graph representations

1.7.4 Undirected graph representations

1.7.5 Weighted graph representations

- easier with adjacency matrix where the entry is simply the weight
- weight needs to be explicitly stored in the node of the linked list

1.8 Adjacency Matrix Vs Adjacency List

- which graph representation is more space efficient depends on the number of edges in the graph
- adjacency matrix cost $O(\|V\|^2)$
- adjacency list cost $O(\|V\| \times \|E\|)$
- as the graph becomes denser, the adjacency matrix becomes relatively more space efficient
  - storing pointers and extra space for weight can be costly for adjacency list
- adjacency list is more space efficient when the graph is sparse

1.9 Exercises

1. What is the degree of vertex 3 in above graph?
2. Which of the following correctly describe the graph?
   1. Connected Graph
   - Dense Graph
   - Directed Graph
   - Acyclic Graph
   - Sparse Graph
   - Complete Graph
   - Undirected Graph
3. A simple path:
   1. must have all vertices unique except that the first and last vertices are the same
   - must have all vertices be unique

- allows repetition of vertices so long as the length of the path is less than 5
- None of the above

4. What are different simple paths between vertices 0 and 4 in above graph?

```
[ ]:
```