Graph Algorithms

Chapter 22
Graphs

- A flexible data structure for a wide range of applications
- Consists mainly of Vertices (nodes) and Edges (arcs)
- Vertices and/or Edges can be annotated with further information
- Applications?
Graphs

▶ Social Network

▶ Knowledge Base

Gavin Newsom
Governor-of
Resident-in
California

Washington

Capital-of
US

State-in
Outline

- Types of graphs
- Representations of graphs
  - Adjacency list
  - Adjacency matrix
- Elementary graph algorithms
  - Bread-first Search (BFS)
  - Depth-first Search (DFS)
  - Connectivity
  - Cycle Detection
Types of Graphs

- Directed and Undirected graphs
- Weighted and Unweighted graphs
- Connected graphs
- Bipartite graphs
- Acyclic graphs
Undirected Graph

- No direction in edges
- An edge can be traversed in both ways
- E.g., Facebook friends

![Diagram of an undirected graph with nodes 1, 2, 3, 4, 5 connected in a cycle.]
Directed Graph

- Direction on edges
- An edge can be traversed in one direction
- E.g., Twitter follows
Weighted Graph

Vertices and/or edges can be assigned weights
E.g., road network

Unweighted Graph

Weighted Graph
Connected Graphs

Connected Graph

Disconnected Graph
Bipartite Graph

A graph where the vertices can be partitioned into two groups where there are no edges within a group and all the edges are from one group to the other.
Acyclic Graph

- A graph that has at least one cycle

Acyclic Graph

Non-acyclic Graph
Adjacency List
Graph Representations
Adjacency List
Adjacency Matrix
Adjacency Matrix

```
1  2  3  4  5  6
1 0 1 0 1 0 0
2 0 0 0 0 1 0
3 0 0 0 0 1 1
4 0 1 0 0 0 0
5 0 0 0 1 0 0
6 0 0 0 0 0 1
```
Graph Traversals
Breadth-first Search
Depth-first Search
Cycle Detection

- A path is a sequence of edges that can be traversed from a source node to a destination node
- A cycle is an infinite path in a graph
- Given an undirected graph, how to detect if it has a cycle?
- Find the vertices that comprise a cycle
- Given a directed graph, how to detect if it has a cycle?
Minimum Spanning Tree
MST Problem Definition

Given a weighted undirected graph $G = (V, E, W)$, where $V$ is the set of vertices, $E$ is the set of edges, and $W$ is the weighting function which defines the weight $w(u, v)$ for any pair of vertices $\langle u, v \rangle$. We want to find an acyclic subset $T \subseteq E$ that connects all of the vertices and whose total weight $w(T) = \sum_{(u,v) \in T} w(u, v)$, is minimized.
Example
Non-minimum Spanning Tree

Total weight = 21
A Minimum Spanning Tree

Total weight = 16
Another MST

Total weight = 16
Kruskal’s Algorithm

- Sort all the edges by weight
- Scan the edges by weight from lowest to highest
- If an edge introduces a cycle, drop it
- If an edge does not introduce a cycle, pick it
- Terminate when n-1 edges are picked
Kruskal’s MST in Action
Kruskal’s MST in Action
Kruskal’s MST in Action
Kruskal’s MST in Action
Kruskal’s MST in Action
Kruskal’s MST in Action
Kruskal’s MST in Action
Kruskal’s MST in Action
Kruskal’s MST in Action
Kruskal’s MST in Action
Prim’s Algorithm

- Start from any node and mark it as visited
- Set the weight of each node to the lowest weight of an incident edge with a visited node
- Find the node with the lowest weight and visit it
- Repeat until all the n nodes are visited
Prim’s MST in Action

A

B

C

D

E

F

\[ \begin{array}{cccc}
\infty & 1 & \infty & \infty \\
3 & \infty & 2 & 4 \\
4 & 4 & \infty & 5 \\
\infty & \infty & 7 & \infty \\
\end{array} \]
Prim’s MST in Action
Prim’s MST in Action
Prim’s MST in Action
Prim’s MST in Action
Prim’s MST in Action
Prim’s MST in Action
Prim’s MST in Action
Prim’s MST in Action
Generic-MST

- Generic-MST(G, w)
  - $A = \{\}$
  - While $A$ does not form a spanning tree
    - Find an edge $(u, v)$ that is safe for $A$
    - $A = A \cup \{(u, v)\}$
  - Return $A$

- If $A$ is a subset of a MST, then an edge $(u, v)$ if safe is $A \cup \{(u, v)\}$ is also a subset of a MST.
The edge (A,B) crosses the cut.

\[ S = \{A, D, E\} \]
\[ V - S = \{B, C, F\} \]

Cut \((S, V - S)\)
Light edge/Safe edge

- A light edge: is a crossing edge with the minimum weight
- A safe edge: is an edge that can be added to $A$ while keeping it a subset of the MST
- Theorem: A light edge is a safe edge
Kruskal’s Algorithm

\textsc{Kruskal}(G, w)

\begin{align*}
O(1) & \quad A = \emptyset \\
O(1) & \quad \text{for each vertex } v \in G.V \quad // \ V \text{ iterations} \\
O(1) & \quad \text{MAKE-SET}(v) \\
O(\log E) & \quad \text{sort the edges of } G.E \text{ into nondecreasing order by weight } w \\
& \quad \text{for each } (u, v) \text{ taken from the sorted list } // \ E \text{ iterations} \\
O(\alpha(V)) & \quad \text{if } \text{FIND-SET}(u) \neq \text{FIND-SET}(v) \\
O(1) & \quad A = A \cup \{(u, v)\} \\
O(\alpha(V)) & \quad \text{UNION}(u, v) \\
\text{return } A
\end{align*}
Prim’s Algorithm

\( \text{PRIM}(G, w, r) \)

\( Q = \emptyset \)

\textbf{for} each \( u \in G.V \) \quad // \ V \ \text{iterations}

\( u.\text{key} = \infty \)

\( u.\pi = \text{NIL} \)

\( O(\log V) \quad \text{INSERT}(Q, u) \)

\( O(\log V) \quad \text{DECREASE-KEY}(Q, r, 0) \quad // \ r.\text{key} = 0 \)

\textbf{while} \( Q \neq \emptyset \)

\( O(\log V) \quad u = \text{EXTRACT-MIN}(Q) \)

\textbf{for} each \( v \in G.\text{Adj}[u] \) \quad // \ V \ \text{iterations}

\textbf{if} \( v \in Q \) and \( w(u, v) < v.\text{key} \)

\( v.\pi = u \)

\( O(\log V) \quad \text{DECREASE-KEY}(Q, v, w(u, v)) \)