CSE 373: Graphs

Chapter 9

Graphs

- A data structure useful for representing relationships between things
- A graph $G$ is represented as $G = (V,E)$
  - $V$ = a set of vertices (nodes)
  - $E$ = a set of edges connecting vertices from $V$

- More general and arbitrary than trees (trees are a restricted type of graph)
**Directed/Undirected Graphs**

- In *directed* graphs, edges have a specific direction:

  ![Directed Graph Example](image)

- In *undirected* graphs, they don’t:

  ![Undirected Graph Example](image)

- Vertices $u$ and $v$ are *adjacent* if $(u,v) \in E$

---

**Weighted Graphs**

*weighted graphs* store a weight with each edge:

![Weighted Graph Example](image)

(nodes sometimes have weights too…)

---

UW, Autumn 1999  CSE 373 – Data Structures and Algorithms  Brad Chamberlain
**Paths**

A path is a list of vertices $v_1, v_2, \ldots, v_n$ such that $(v_i, v_{i+1}) \in E$.

$p = \text{Seattle, Salt Lake City, Phoenix, San Antonio, Phoenix, Seattle}$

**Path Length and Cost**

- The length of a path is the number of edges
- Its cost is the sum of the edges’ weights

$\text{length}(p) = 5 \quad \text{cost}(p) = 11$
Simple Paths and Cycles

A simple path repeats no vertices (except the first can also be the last):

- Seattle, Salt Lake City, Phoenix, San Antonio
- Seattle, Salt Lake City, San Antonio, Phoenix, Seattle

A cycle is a path that starts and ends at the same node

- Seattle, Salt Lake City, San Antonio, Phoenix, Seattle
  (For undirected graphs edges cannot appear twice)

Directed Acyclic Graphs

Directed Acyclic Graphs (DAGs) are directed graphs that contain no cycles

main()  
\[ \text{solve}() \rightarrow \text{mult}() \]
\[ \text{pow}() \rightarrow \text{sqrt}() \]

trees $\subseteq$ DAGs $\subseteq$ directed graphs
Connectivity

- Undirected graphs are *connected* if there is a path between any two vertices.

- Directed graphs are *strongly connected* if there is a path between any two vertices.

- It is *weakly connected* if it’s connected when direction is ignored.

- A *complete* graph is one that has an edge between every pair of vertices.

Graph Implementations

*Adjacency Matrix:* A $|V| \times |V|$ array in which:
- element $(u,v)$ is 1 if there is an edge $(u,v)$
- it is 0 otherwise
- for weighted graphs, store weights rather than 1/0

- space requirements?
Graph Implementations

Adjacency Lists: A $|V|$-ary array in which each entry stores a list of all adjacent vertices

```
Han  Luke
Leia
```

Space requirements?

How could we index into an adjacency list or matrix when nodes are named?

---

Graph Applications

- Storing things that are graphs by nature (AI)
  - distances between cities
  - airline flights, travel options
  - relationships between people, things
  - distances between rooms in the game Clue
- Compilers
  - callgraph – which functions call which other ones
  - dependence graphs – which variables are defined and used at which statements