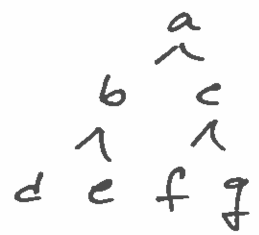
Graphs

4-10-17

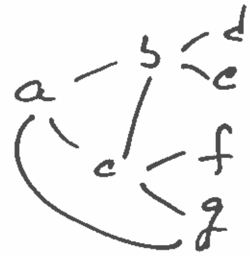
1



Tree example

Root, parent-child relationship

a is parent of c.



Graph example

Edges connect vertices.

Connected vertices are adjacent.

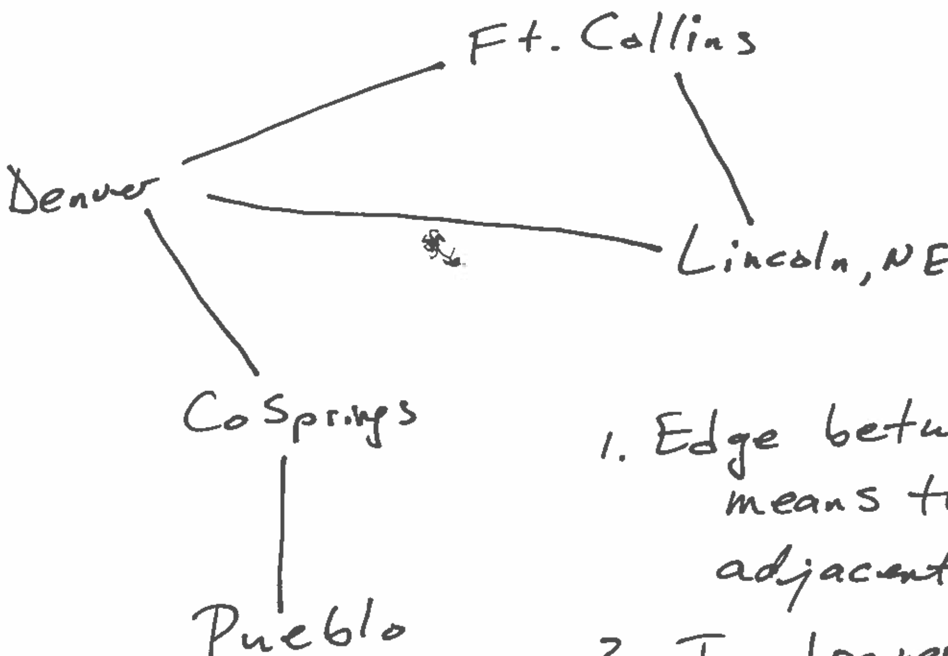
a is adjacent to c.

Any vertex can be the root.

Can get to any adjacent vertex from a vertex.

Represent a Road Map

Set of cities and roads that connect them.



1. Edge between two cities means that they are adjacent
2. In traversing the graph, can only go to an adjacent city from a current city

e.g. Denver to Pueblo is

Denver → Co Springs → Pueblo

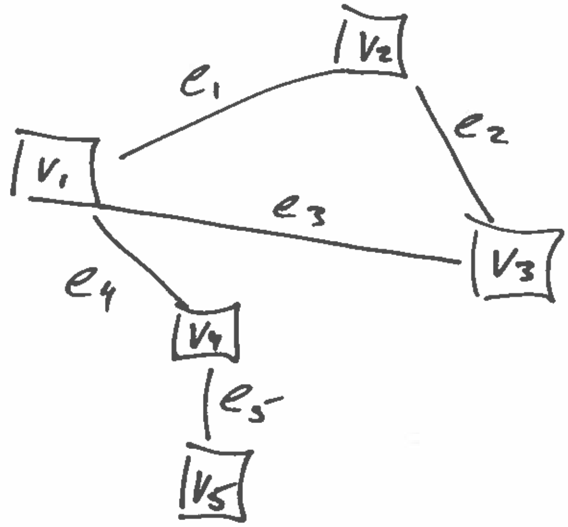
Graph

4-10-17

Formal definition:

Graph defined as

$G = \{V, E\}$ where V is a set of vertices $\langle v_1, v_2, \ldots v_k \rangle$ and E is a set of edges $\langle e_1, e_2, \ldots e_n \rangle$.



Road map graph

Graph representations

Adjacency matrix - good to know
Adjacency list - this is what we'll focus on in the list.

Weighted and unweighted graphs

Unweighted - all edges have a uniform weight of 1
Weighted - Each edge has a value, e.g. distance if vertices are cities

Adjacency matrix, unweighted

2d matrix. All vertices listed on horizontal and vertical axis.

axis is starting vertex.

vertical

horizontal axis is ending vertex

	V_1	V_2	V_3	V_4	V_5
V_1	0	1	1	1	0
V_2	1	0	1	0	0
V_3	1	1	0	0	0
V_4	1	0	0	0	1
V_5	0	0	0	1	0

1 = edge

0 = no edge

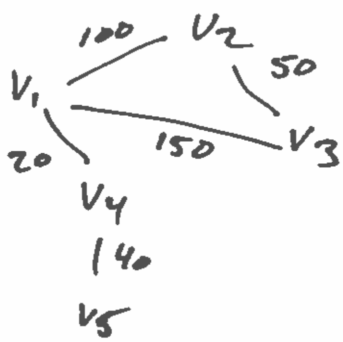
Assume edge V_b to V_a also means edge V_a to V_b

Weighted graph

No edge = -1

Replace 1 with the edge weight

Ex:



Where edge weight might be distance, flow of goods between companies, strength of connection of some sort

	V_1	V_2	V_3	V_4	V_5
V_1	0	100	150	20	0
V_2	100	0	50	-1	-1
V_3	150	50	0	-1	-1
V_4	20	-1	-1	0	40
V_5	-1	-1	-1	40	0

Using 0 for V_a to V_a weight

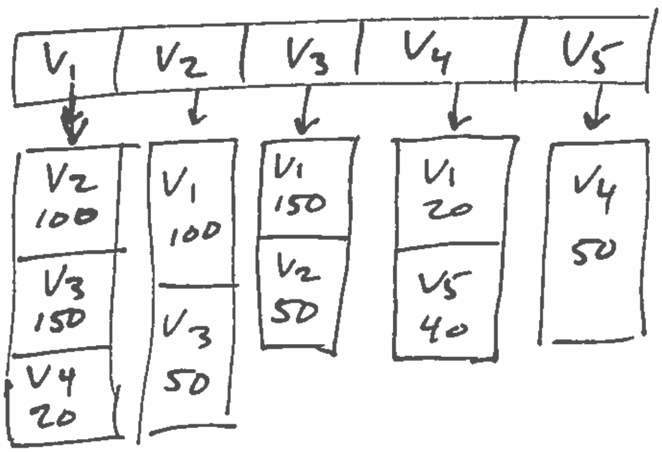
Adjacency List

4-10-17

If there aren't many edges, there will be many zeros or -1 in the adjacency matrix. Could mean spending time looping through a matrix uneccessarily.

Adjacency list only stores information about adjacent edges.

For each vertex, store a list of adjacent vertices, including weight in a weighted graph.



Weighted example

Implementation example

In Lecture notes for Friday, 4/7, we looked at example using vectors of vectors. One vector, called vertices, was all vertices in the graph. Each vertex in vertices also included a vector of adjacent vertices, which contained a pointer to a vertex and could also include the edge weight.

Directed vs undirected graph

Undirected - edges go in both directions

Ex: Edge from $V_1 \Rightarrow V_2$ means there is also an edge $V_2 \Rightarrow V_1$.

Directed - edge goes in one direction and may not exist in other direction.

Ex: Edge $V_1 \Rightarrow V_2$ doesn't guarantee edge $V_2 \Rightarrow V_1$.

Graph Implementation

Create a vertex struct. Has a key and a vector for adjacent vertices

```
struct vertex {
    string key;
    vector<adjacent> adj;
};

struct adjacent {
    vertex *v;
    int weight;
};
```

Graph ADT

4-10-17

What needs to be included in a Graph

```
Graph:
    private:
        vertices

    public:
        Graph()
        insertVertex(value)
        insertEdge(startValue, endValue, weight)
        deleteVertex(value)
        deleteEdge(startValue, endValue)
        printGraph()
        search(value)
```

vertices includes the adjacency list for each vertex.

```
insertVertex(value)
    Pre: value is valid key/search value
    Post: vertex added to vertices if it doesn't
            already exist
    bool found = false;
    for(int i=0; i < vertices.size(); i++){
            // could use iterator if vertices is a vector
            if(vertices[i].key == value){
                    found = true;
                    break;
            }
    }
```

```
    if(found==false){
        vertex v;
        v.key=value;
        vertices.push_back(v);
    }
}
```

code assumes we don't have fixed number of vertices. Can build graph dynamically. No edges yet.

Example:

```
Graph g;
g.insertVertex("B")
g.    "       ("C")
g.    "       ("D")
```

Can think of it as:

"B"	"C"	"D"

or

C

B

D

Vertices, but no edges.