Graph Traversal
Some Graph Types
- Undirected vs. Directed
- Weighted vs. Unweighted
- Cyclic vs. Acyclic
- Simple vs. Non-Simple
- Implicit vs. Explicit
- Labeled vs. Unlabeled
Data Structures for Graphs

Datastructures

- Adjacency Matrix
- Adjacency Lists
- Table of Edges

Which DS to use?

- Depending on the application
- Size vs. Speed
- What type of queries? (is there an edge from $a$ to $b$ etc)
Concrete Implementation

- adjacency matrix
  - bool adj[n][n];
  - vector<char> adj(n*n,0); // edge(i,j) at adj[i*n+j]
Concrete Implementation

- adjacency matrix
  
  ```cpp
  bool adj[n][n];
  vector<char> adj(n*n,0); // edge(i,j) at adj[i*n+j]
  ```

- adjacency list
  
  ```cpp
  vector<vector<int> > adj_list (n);
  int adj_list[n][n]; int num_edges[n];
  ```
Concrete Implementation

- adjacency matrix
  ```cpp
  bool adj[n][n];
  vector<char> adj(n*n,0); // edge(i,j) at adj[i*n+j]
  ```

- adjacency list
  ```cpp
  vector<vector<int> > adj_list (n);
  int adj_list[n][n]; int num_edges[n];
  ```

- list of edges
  ```cpp
  pair<pair<int,int>,double> edges[m];
  vector<pair<int,int>,double> > edges (m);
  ```
# Graph Traversal Algorithms

## Algorithms
- Breadth First Search
- Depth First Search

## Applications
- Connected Components
- Finding Shortest Paths in Unweighted Graphs
- Finding Cycles
- Topological Sorting
- Finding Articulation Nodes
BFS (connected graph)

```cpp
vector<vector<int> > g (n);
// ...
vector<char> visited (n, false);
queue<int> q;
q.push (start);
visited[start] = true;
while(!q.empty()) {
    int curr = q.front(); q.pop();
    // do something with curr
    for (vector<int>::iterator it (g[curr].begin());
         it != g[curr].end(); ++it) {
        if (!visited[*it]) {
            visited[*it] = true;
            q.push(*it);
        }
    }
}
```
Articulation Vertex / Cut Vertex

Cut vertex

- cut vertex: if removed more connected components
- consider the dfs tree on the graph
- \(d[v]\) dfs number (in order)
- \(T_v\) = subtree rooted at vertex \(v\)
- \(low[v] = \min\left\{\begin{array}{l}d[y] \quad (x, y) \text{ is a non-tree edge and } x \in T_v \\ d[v] \end{array}\right\}\)
- The root of the dfs-tree is a cut vertex iff it has \(\geq 2\) children.
- an internal vertex \(v\) is a cut vertex iff \(v\) has a child \(x\) with \(low[x] \geq d[v]\)