18 - Graph Search

**Applications**
- Street Maps
- Computer Networks
- Relationships
- Google Grid

**Questions**
- Shortest Path
- Is something in a graph?
- How can we connect everything with the least cost
- Maximum throughput
Representations

Adjacency Matrix

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Adjacency List

*6* → 1 → 4

*1* → 2

*2* → 0 → 1 → 4

*3* → 1

*4* → 1 → 4

Graph

V = *vertices*

E = *edges*

Undirected? ½ space
Weights on edges?
Transpose?
Graph Searches

Depth-first Search (DFS)  Breadth-first Search (BFS)
void DFS-visit(int i) {
    time ++;
    V[i].start = time;
    for all neighbors j of i {
        if( V[j].start == 0 ) {
            V[j].pred = i;
            DFS-visit(j);
        }
    }
    time++;
    V[i].finish = time;
}

**DFS**

Tree edges
Back edge - No finish time
Forward edge - Finish time, neighbor's start is later
Cross edge - Finish time, neighbor's start is earlier

\(O(V+E)\)
void BFS(int s) {
    V[s].depth = 0;
    Q = new Queue();
    Q.enqueue(s);
    while Q is non-empty {
        i = Q.dequeue();
        for all neighbors j of i {
            if( V[j].depth == -1 ) {
                V[j].depth = V[i].depth + 1;
                V[j].pred = i;
                Q.enqueue(j);
            }
        }
    }
}
Topological Sort

1. Find ID
   17/18

2. Find ATM card
   19/20

3. Get Dressed
   9/12

4. Pick up boy/girl-friend
   10/11

5. Curse bad driver with UNC sticker on his/her car
   13/14

6. Go to ATM
   1/6

7. Put on shoes
   7/8

8. Buy Drinks
   2/5

9. Go to Party
   3/4

10. Put on cool shades
    15/16
$O(V+E)$

Topological Sort only for acyclic, directed graphs.