Topological Sort

- Given a directed acyclic graph, construct an ordering of the vertices such that if there is a path from \( v_i \) to \( v_j \), then \( v_i \) appears after \( v_j \) in the ordering.
- indegree of \( v \): \# of edges \((u, v)\).

```cpp
void Graph::toposort(){
    Vertex v, w;
    for (int counter=0; counter < NUM_VERTICES; counter++){
        v = findNewVertexOfDegreeZero();
        if (v == NOT_A_VERTEX)
            throw CycleFound();
        v.topologicalNum = counter;
        for each w adjacent to v
            w.indegree--;
    }
}
```

Shortest Path Algorithms

- ("single-source" shortest path)
  - Given a graph \( G = (V, E) \) and a single distinguished vertex \( s \), find the shortest weighted path from \( s \) to every other vertex in \( G \).

  weighted path length of \( v_1, v_2, \ldots, v_N \):
  \[
  \sum_{i=1}^{N-1} c_{i,i+1}, \text{ where } c_{i,j} \text{ is the cost of edge } (v_i, v_j)
  \]

Unweighted Shortest Path

- Special case of the weighted problem: all weights are 1.
- Solution: breadth-first search. Similar to level-order traversal for trees.
void Graph::unweighted (Vertex s){
    Queue q(NUM_VERTICES);
    Vertex v, w;
    q.enqueue(s);
    s.dist = 0;
    while (!q.isEmpty()){
        v = q.dequeue();
        for each w adjacent to v if (w.dist == INFINITY)
            w.dist = v.dist + 1;
        q.enqueue(w);
    }
}

Weighted Shortest Path

• no negative weight edges.
• **Dijkstra’s algorithm**: uses similar ideas as the unweighted case.

**Greedy algorithms:**

*do what seems to be best at every decision point.*

Analysis

• How long does it take to find the smallest unknown distance?
  – simple scan using an array:
  – binary heap:
• Total running time:
  – simple scan:
  – binary heap: