Graph Algorithms

- **Topological Sort**
  - Produce a valid ordering of all nodes, given pairwise constraints
  - Solution usually not unique
  - When is solution impossible?

- **Topological Sort Example: Getting an AB in CPS**
  - Express prerequisite structure
  - This example, CPS courses only: 6, 100, 104, 108, 110, 130
  - Ignore electives or outside requirements (can add later)
Topological Sort

- **Topological Sort Algorithm**
  1. Find vertex with no incoming edges
  2. Remove (updating incoming edge counts) and Output
  3. Repeat 1 and 2 while vertices remain
     - Complexity?

- **Refine Algorithm**
  - Use priority queue?
  - Complexity?

- **What is the minimum number of semesters required?**
  - Develop algorithm
Shortest Path (Unweighted)

- Mark all vertices with infinity (*) exec. starting vertex with 0
- Place starting vertex in queue
- Repeat until queue is empty:
  1. Remove a vertex from front of queue
  2. For each adjacent vertex marked with *,
     - process it,
     - mark it with source distance + 1
     - place it on the queue.
Shortest Path (Unweighted)

- Mark all vertices with infinity (*)
- Mark starting vertex with 0
- Place starting vertex in queue
- Repeat until queue is empty:
  1. Remove a vertex from front of queue
  2. For each adjacent vertex marked with *,
     - process it,
     - mark it with source distance + 1
     - place it on the queue.

How do we get actual “Path”?

CompSci 100E
Shortest Path (Weighted): Dijkstra

- Unmark all vertices and give infinite weight
- Set weight of starting vertex at 0 and place in priority queue
- Repeat until priority queue is empty:
  1. Remove a vertex from priority queue
     - Process and mark (weight now permanent)
  2. For each adjacent unmarked vertex
     - Set weight at lesser of current weight and (source weight + path weight)
     - (This may involve reducing previous weight setting)
     - Place in priority queue (if not there already)
Shortest Path (Weighted): Dijkstra
Shortest Path (Weighted): Dijkstra

Mark all vertices with infinity (*).
Mark starting vertex with 0.
Place starting vertex in queue.

Repeat until queue is empty:
1. Remove a vertex from front of queue.
2. For each adjacent vertex marked with *, process it, mark it with source distance + 1, place it on the queue.

How do we get actual “Path”?
Other Graph Algorithms

- Traveling Salesman
- Spanning Trees
- Paths with negative weights