CSE 417: Algorithms and Computational Complexity

Winter 2001

Lecture 12

Instructor: Paul Beame

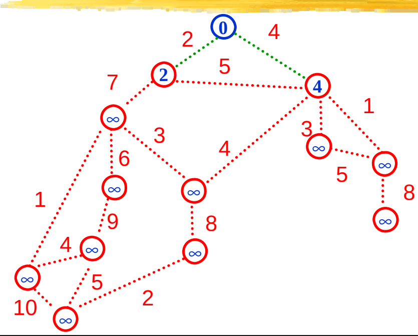
Single-source shortest paths

- Given an (un)directed graph G=(V,E) with each edge e having a weight w(e) and a vertex v
- Find length of shortest paths from v to each vertex in G
  - -∞ if there is a (directed) cycle with negative weight
    - **go around the cycle over and over**
  - Assume first that there are no negative cost edges

A greedy algorithm

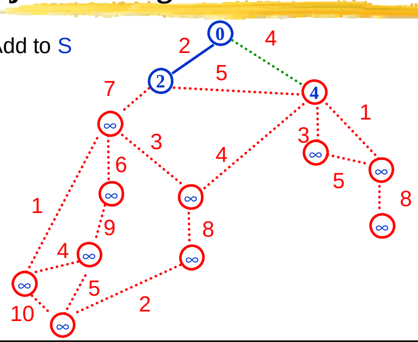
- Dijkstra's Algorithm:
  - Maintain a set **S** of vertices whose shortest paths are known
    - initially **S={v}**
  - Maintaining current best lengths of paths that only go through **S** to each of the vertices in **G**
    - path-lengths to elements of **S** will be right, to **V-S** they might not be right
  - Repeatedly add vertex **u** to **S** that has the shortest path-length of any vertex in **V-S**
    - update path lengths based on new paths through u

Dijkstra's Algorithm



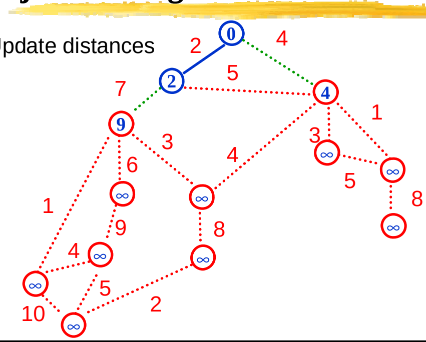
Dijkstra's Algorithm

Add to S



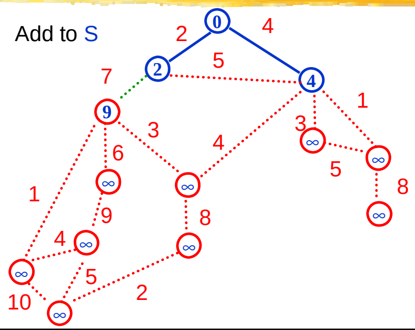
Dijkstra's Algorithm

Update distances



Dijkstra's Algorithm

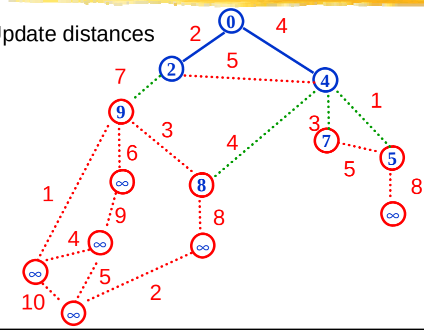
Add to S



7

Dijkstra's Algorithm

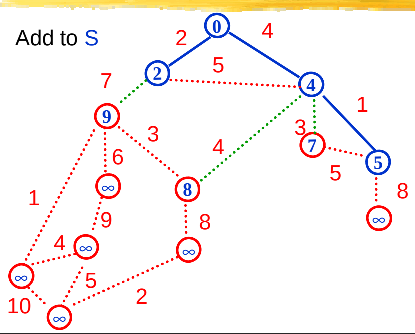
Update distances



8

Dijkstra's Algorithm

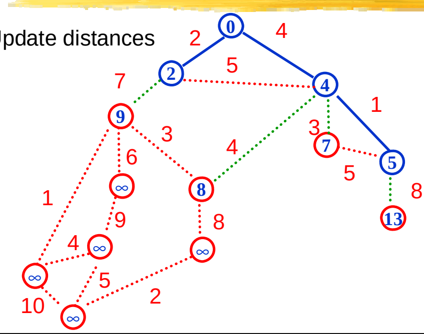
Add to S



9

Dijkstra's Algorithm

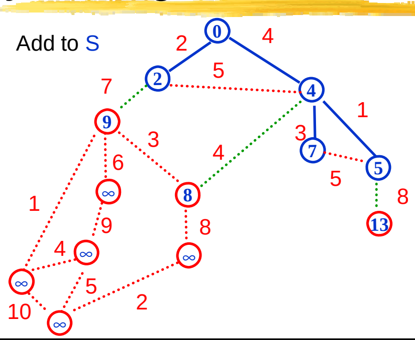
Update distances



10

Dijkstra's Algorithm

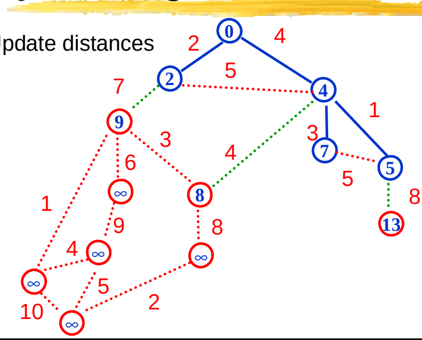
Add to S



11

Dijkstra's Algorithm

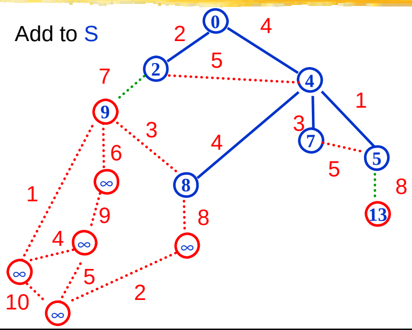
Update distances



12

Dijkstra's Algorithm

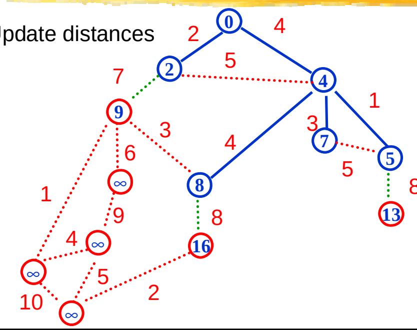
Add to S



13

Dijkstra's Algorithm

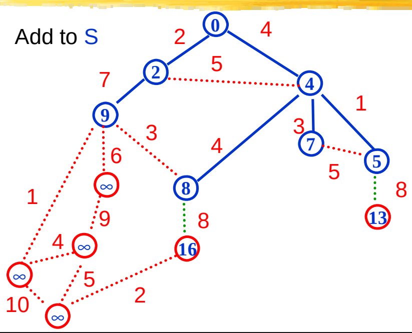
Update distances



14

Dijkstra's Algorithm

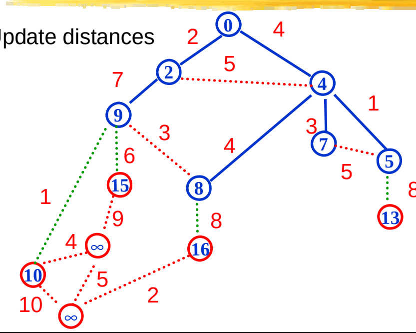
Add to S



15

Dijkstra's Algorithm

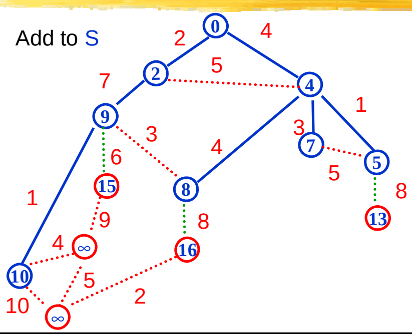
Update distances



16

Dijkstra's Algorithm

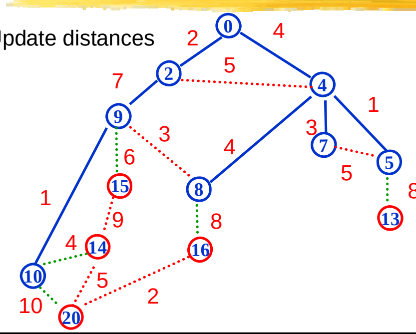
Add to S



17

Dijkstra's Algorithm

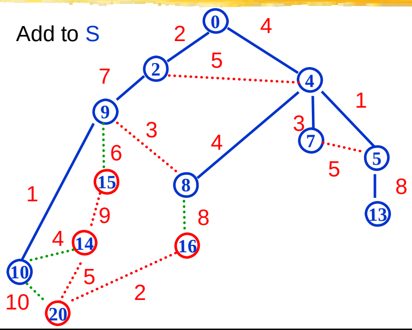
Update distances



18

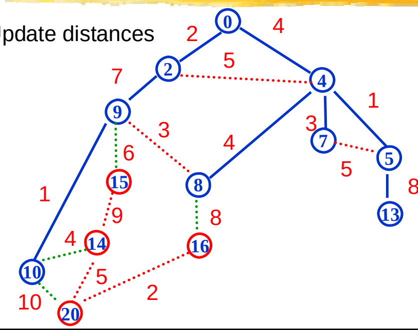
Dijkstra's Algorithm

Add to S



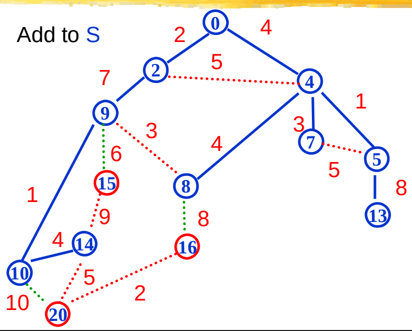
Dijkstra's Algorithm

Update distances



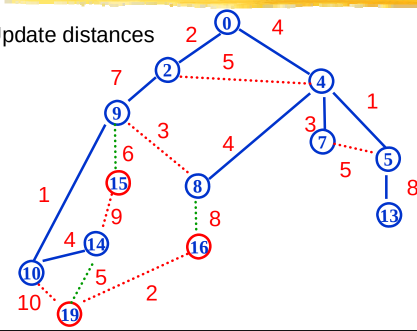
Dijkstra's Algorithm

Add to S



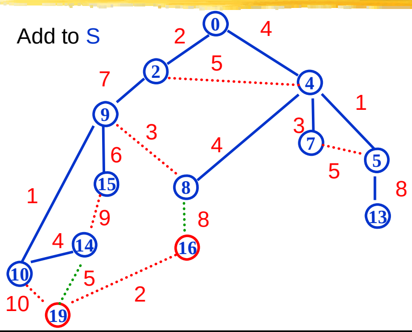
Dijkstra's Algorithm

Update distances



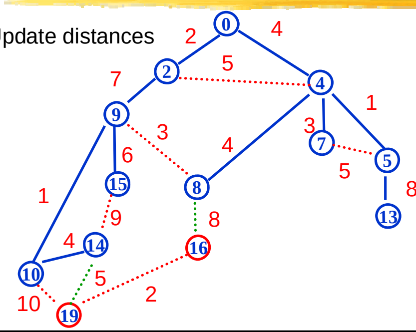
Dijkstra's Algorithm

Add to S



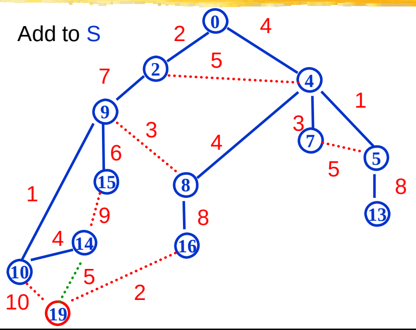
Dijkstra's Algorithm

Update distances



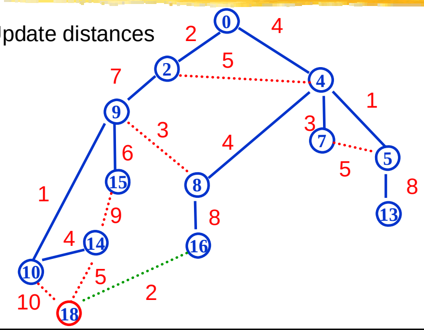
Dijkstra's Algorithm

Add to S



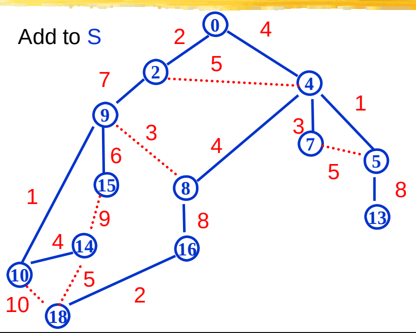
Dijkstra's Algorithm

Update distances



Dijkstra's Algorithm

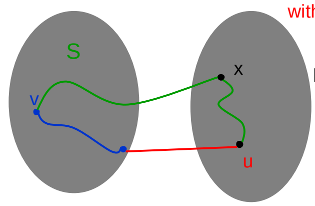
Add to S



Dijkstra's Algorithm Correctness

Suppose all distances to vertices in S are correct
and u has smallest current value in V-S

∴distance value of vertex in V-S=length of shortest path from v with only last edge leaving S



Suppose some other path to u and x= first vertex on this path not in S

d(u)≤ d(x)

x-u path length ≥ 0

∴ other path is longer

Therefore adding u to S keeps correct distances

Dijkstra's Algorithm

- Algorithm also produces a tree of shortest paths to v
  - From w follow its ancestors in the tree back to v
- If all you care about is the shortest path from v to w simply stop the algorithm when w is added to S

Implementing Dijkstra's Algorithm

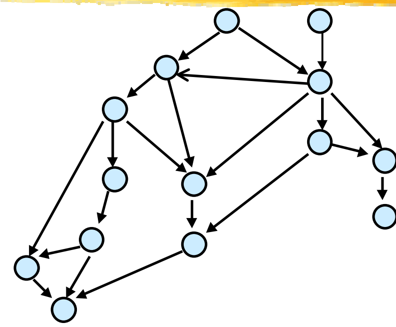
- Need to
  - keep current distance values for nodes in V-S
  - find minimum current distance value
  - reduce distances when vertex moved to S
- Same operations as priority queue version of Prim's Algorithm
  - only difference is rule for updating values
    - node value + edge-weight vs edge-weight alone
  - same run-times as Prim's Algorithm O(m log n)

Topological Sort

- **Given:** a directed acyclic graph (DAG)G=(V,E)
- **Output:** numbering of the vertices of G with distinct numbers from 1 to n so edges only go from lower number to higher numbered vertices
- Applications
  - nodes represent tasks
  - edges represent precedence between tasks
  - topological sort gives a sequential schedule for solving them

31

Directed Acyclic Graph



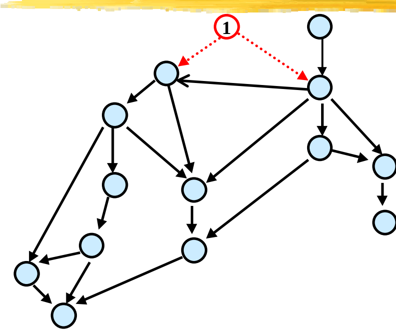
32

Topological Sort

- Can do using DFS (see book)
- Alternative simpler idea:
  - Any vertex of in-degree 0 can be given number 1 to start
  - Remove it from the graph and then give a vertex of in-degree 0 number 2, etc.

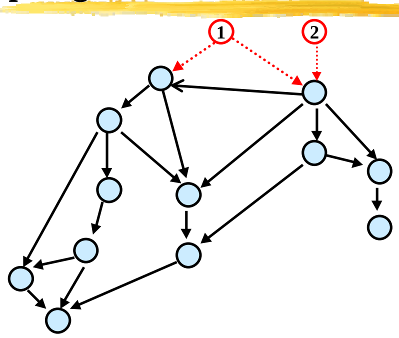
33

Topological Sort



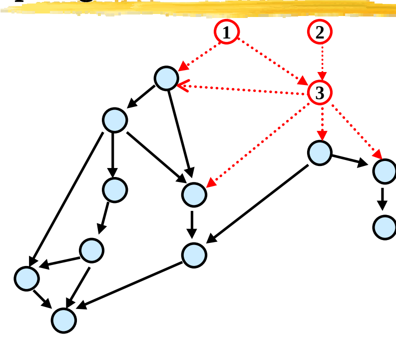
34

Topological Sort



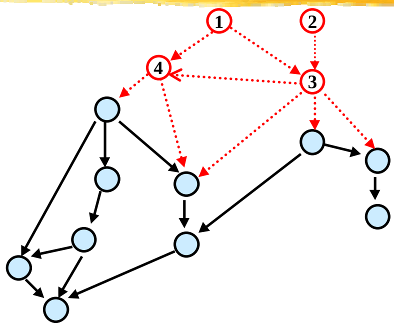
35

Topological Sort



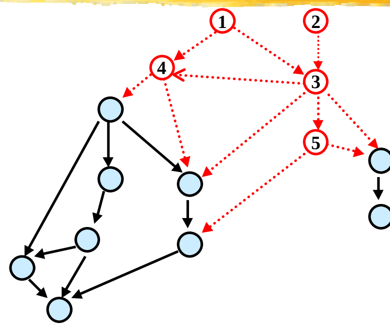
36

Topological Sort



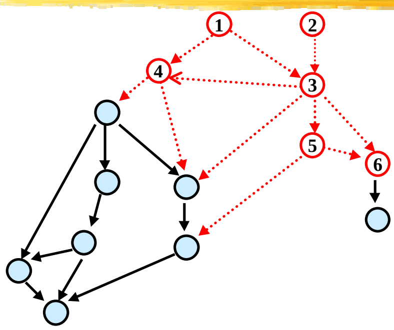
37

Topological Sort



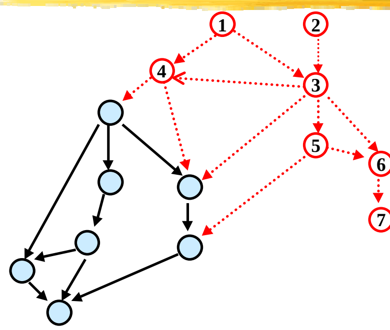
38

Topological Sort



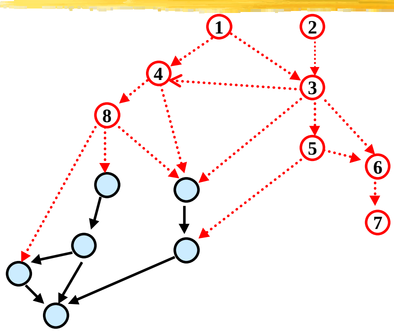
39

Topological Sort



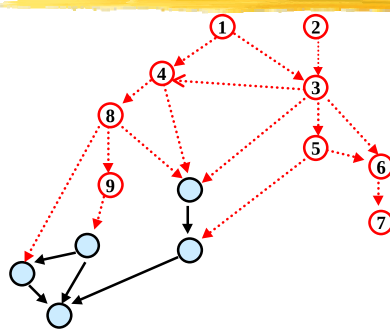
40

Topological Sort



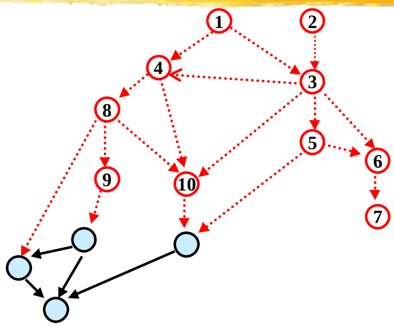
41

Topological Sort



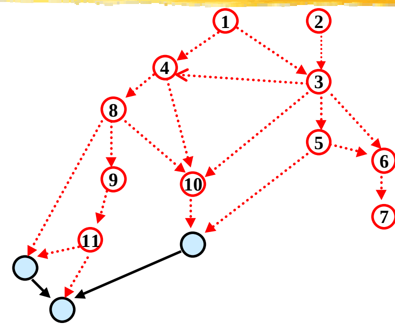
42

Topological Sort



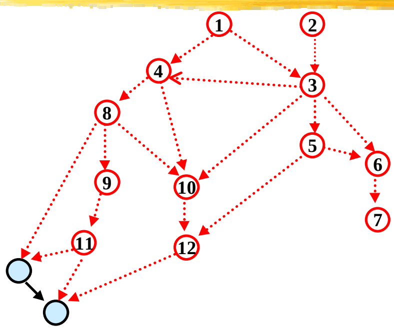
43

Topological Sort



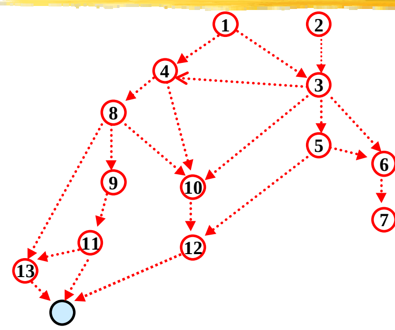
44

Topological Sort



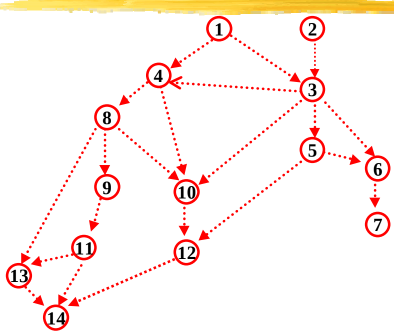
45

Topological Sort



46

Topological Sort



47

Implementing Topological Sort

- Go through all edges, computing in-degree for each vertex O(m+n)
- Maintain a queue (or stack) of vertices of in-degree 0
- Remove any vertex in queue and number it
- When a vertex is removed, decrease in-degree of each of its neighbors by 1 and add them to the queue if their degree drops to 0
- Total cost O(m+n)

48