6.006
Spring 2008
L17

Shortest Paths III: Special cases

Shortest paths in DAGs
Shortest paths in graphs w/o negative edges
Dijkstra's algorithm

Readings: CLRS 24.2, 24.3

DAGs

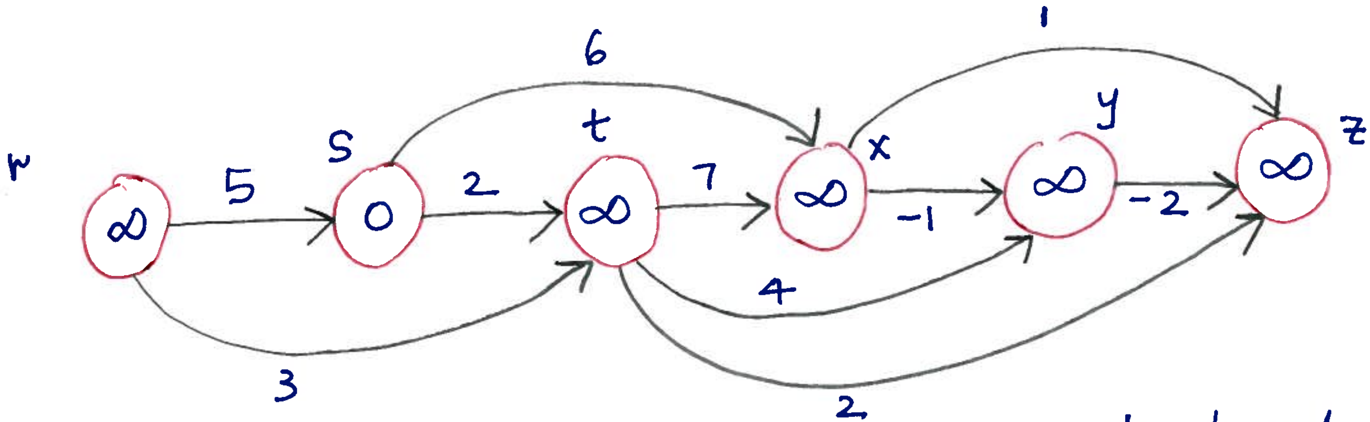
Can't have negative cycles because there are no cycles!

1) Topologically sort the DAG
Path from u to v implies that u is before v in the linear ordering

2) One pass over vertices in topologically sorted order relaxing each edge that leaves each vertex

$\Theta(V+E)$ time

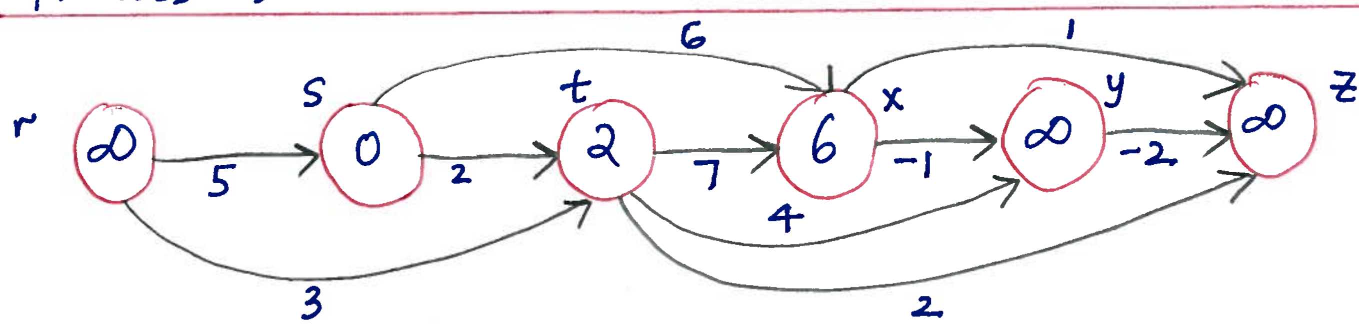
EXAMPLE



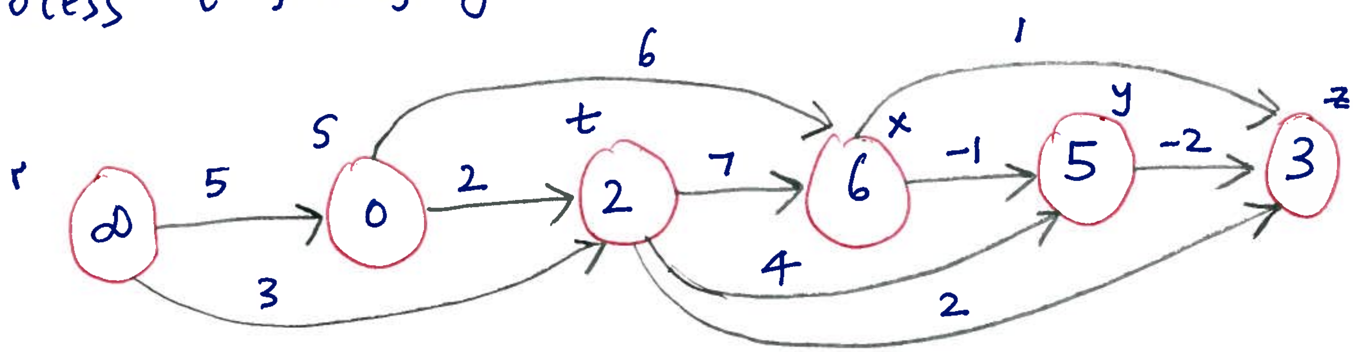
vertices sorted left to right in topological order

Process r: stays ∞. All vertices to the left of s will be ∞ by definition

Process s: $t: \infty \to 2$ $x: \infty \to 6$



Process t, x, y



preview of dynamic programming

DIJKSTRA'S ALGORITHM

For each edge $(u,v) \in E$, assume $w(u,v) \geq 0$

Maintain a set S of vertices whose final shortest path weights have been determined

Repeatedly select $u \in V - S$ with <u>minimum</u> shortest path estimate, add u to S, relax all edges out of u

---

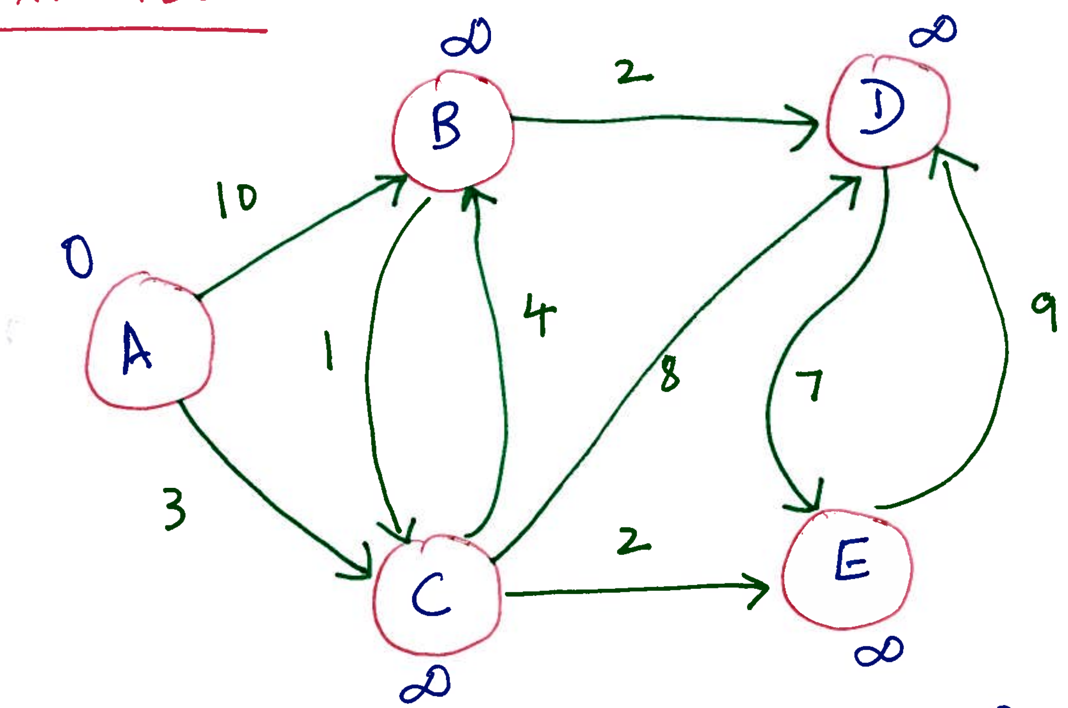
PSEUDO CODE

```
DIJKSTRA (G, w, s)          // uses priority queue Q
    Initialize (G, s)
    S ← ∅
    Q ← V[G]                // Insert into Q
    while Q ≠ ∅
    do u ← EXTRACT-MIN (Q)  // deletes u from Q
    S = S ∪ {u}
    for each vertex v ∈ Adj[u]
        do RELAX (u, v, w)
```

↖ implicit DECREASE-KEY operation

```
RELAX (u, v, w)
  if d[v] > d[u] + w(u,v)
    then d[v] ← d[u] + w(u,v)
         π[v] ← u
```

EXAMPLE



	$\{A$	B	C	D	$E\} = Q$	
$S = \{\ \}$	0	∞	∞	∞	∞	
$S = \{A\}$	0	10	3	∞	∞	← after relaxing edges from A
$S = \{A, C\}$	0	7	3	11	5	← after relaxing edges from C
$S = \{A, C, E\}$	0	7	3	11	5	
$S = \{A, C, E, B\}$	0	7	3	9	5	← after relaxing edges from B
	0	7	3	9	5	

Strategy: Dijkstra is a greedy algorithm: choose closest vertex in $V - S$ to add to set S

Correctness: Each time a vertex u is added to set S, we have $d[u] = \delta(s, u)$

COMPLEXITY

$\Theta(V)$ inserts into priority queue
$\Theta(V)$ EXTRACT-MIN operations
$\Theta(E)$ DECREASE-KEY operations

Array impl: $\Theta(V)$ time for extract min
$\Theta(1)$ for decrease key

Total: $\Theta(V.V + E.1) = \Theta(V^2+E) = \Theta(V^2)$

Binary min-heap: $\Theta(\lg V)$ for extract min
$\Theta(\lg V)$ for decrease key

Total: $\Theta(V \lg V + E \lg V)$

Fibonacci heap: $\Theta(\lg V)$ for extract min
$\Theta(1)$ for decrease key
amortized cost

↑ not covered in 6.006!

Total: $\Theta(V \lg V + E)$