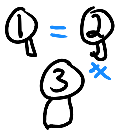
Hw

graph traversal problem

graph ideas before shortest path



1+2 ~ same
1+3 - diff
2+3 - ? same → inconsistent

m judgements

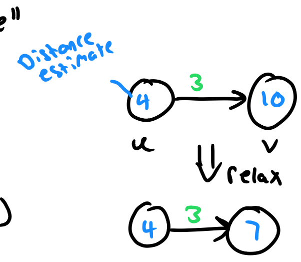
Lecture:

Output:
$\forall v \in V$
v.d - $\delta(s,v)$
v.π - predecessor on shortest path from s ⟶ v

init w/
v.d = ∞
v.π = NIL

All single-source algos do:
- init
- Relax edges until "done"

Relax(u, v, w)
  if v.d > u.d + w(u,v)
    v.d = u.d + w(u,v)
    v.π = u
    return true
  else
    return false

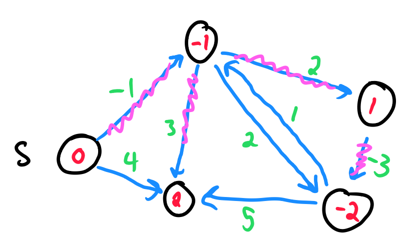


Bellman-Ford

Allows w<0 (negative weights)
Return True if no negative weight cycles reachable from S
false otherwise

Belman-Ford(G, w, s)
  initialize-Single-Source(G, s) O(V)
  for i=1 to |G.V|-1   } |V|-1 passes over |E| edges → Θ(VE) adjacency list
    up = false →
    for each (u,v) ∈ G.E
      up = Relax(u,v,w) or up
    → if !up break
  for each (u,v) ∈ G.E O(E)   } checks for negative weight cycle
    if v.d > u.d + w(u,v)
      return false
  return true

break from loop when done
(if relax doesn't update for one round, done)



Converges in 1 round because used a nice order

Need to show returns shortest paths
Converges in |V|-1 rounds
unless negative cycle

$O(V(V+E)) = O(V^2+VE)$

for each (u,v) ∈ G.E
  ⋮
for v=1 to n
  nei = G.Adj[v]
  while nei is not NIL
    Relax(v, nei, w)
    update nei

V^2 vs VE?

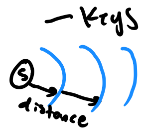
usually smaller: O(VE) = O(?)

|E| = Ω(V)

VE = Ω(V²)

Dijkstra

no neg. weight edges
Weighted BFS / like prim
- use priorityqueue for edges
- keys are v.d



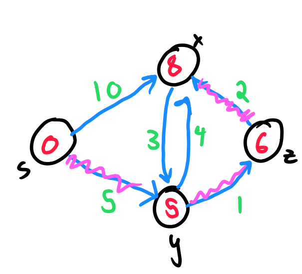
2 sets of vertices

S = {v ∈ V : δ(s,v) known} (= v.d)

V - S still working

Q = min heap
has insert
extract-min
decrease-key

Dijkstra(G, w, s)
  Initialize-Single-Source(G, s) // s.d = 0, v.d = ∞
  S = ∅
  Q = ∅
  for each u = G.V
    Insert(Q, u)
  while Q ≠ ∅
    u = Extract-Min(Q)
    S = S ∪ {u}
    for each v in G.Adj[u]
      Relax(u, v, w)
      if v.d decreased
        Decrease-Key(Q, v, v.d)



S
s : (s,x), (s,y)
y : (y,x), (y,z)
z : (z,x)
x : (x,y)