LEC11 Weighted Shortest Paths

3.28 https://github.com/GUMI-21/MIT6.006_note

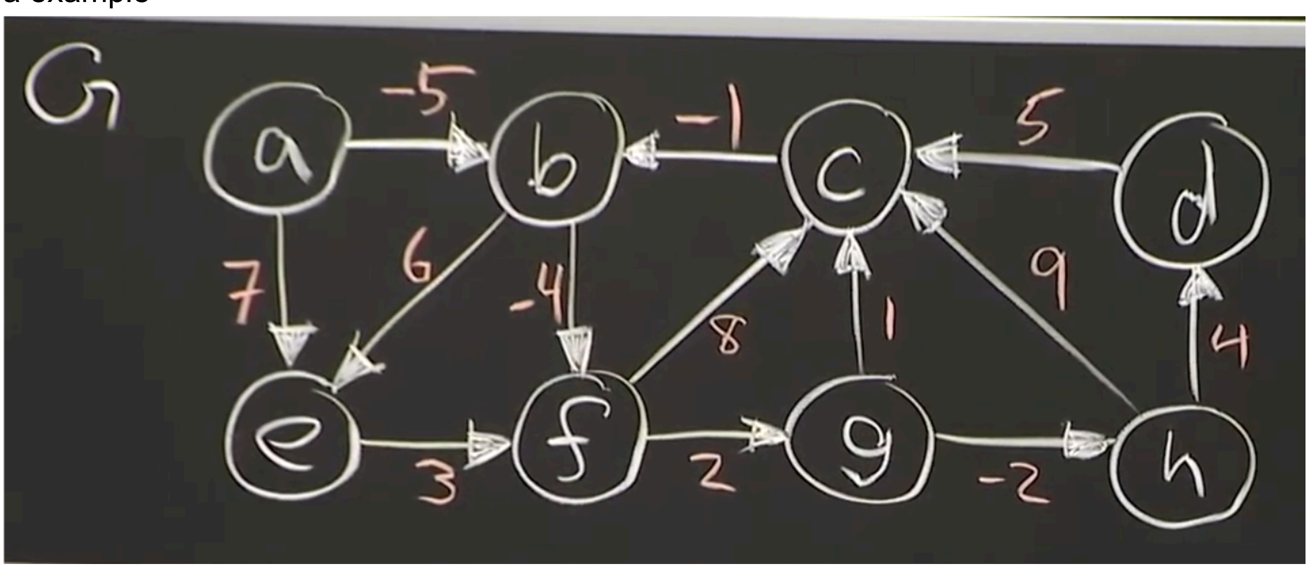
PreReview

- Single-Source Shortest
  - Paths with BFS in O(|V | + |E|) time (return distance per vertex)
- Single-Source *Reachability*
  - with BFS or DFS in O(|E|) time (return only reachable vertices)
  - when talk about Reachability, I only need to juge reachable, doesn't need to maintain parent array
- Connected components
  - with Full-BFS or Full-DFS in O(|V | + |E|) time
- Topological Sort of a *DAG* with Full-DFS in O(|V | + |E|) time
  - every vertex go forward in order
- Previously: distance = number of edges in path
- Today: generalize meaning of distance

Weight Graphs

A weighted graph is a graph G = (V, E) together with a weight function $w : E \to Z$
$e = (u, v), w(e) = w(u, v)$

- a example

  w(b,f) = -4

application

-distance in road network
-latency in network connections
-strength of a relationship in a social network

represent weights computaionally

1.store in adjacenry list. 2.any separate set mapping edge to weigh.

- Inside graph representation: store edge weight with each vertex in adjacency lists
- Store separate Set data structure mapping each edge to its weight
  read weight of edge runtime - O(1)

Weighted Path

weight w(π) if path $\pi = \sum_{e \in \pi} w(e)$

a shortest path(weighted)

is a minimum weight path from S to T.
$\delta(s,t) = inf\{w(\pi)|path\ \pi\ from\ s\ to\ t\}$ (*if no path $\delta(s,t) = \infty$*)
inf: infinity
there is a problem in *Negative-weight Cycles*, if there is a path from s to v that goes through a vertex on a negative weight cycle. $\delta(s,v) = -\infty$
this will be talked in next lec

Weighted Shortest Paths Alg

in next four lectures

- BFS
  -if graph has positive weights, and all weights are the same
  BUT if I have a positive weight edge, sunch as w(u,v) = 4, I can just put four edges of weight 1 in series between u & v. Buf If the weghts is too big can't use this method.

Restrictions		SSSP Algorithm		
Graph	Weights	Name	Running Time $O(\cdot)$	Lecture
General	Unweighted	BFS	$\|V\| + \|E\|$	L09
DAG	Any	DAG Relaxation	$\|V\| + \|E\|$	L11 (Today!)
General	Any	Bellman-Ford	$\|V\| \cdot \|E\|$	L12
General	Non-negative	Dijkstra	$\|V\| \log \|V\| + \|E\|$	L13

Shortest-Path(weighted) Trees

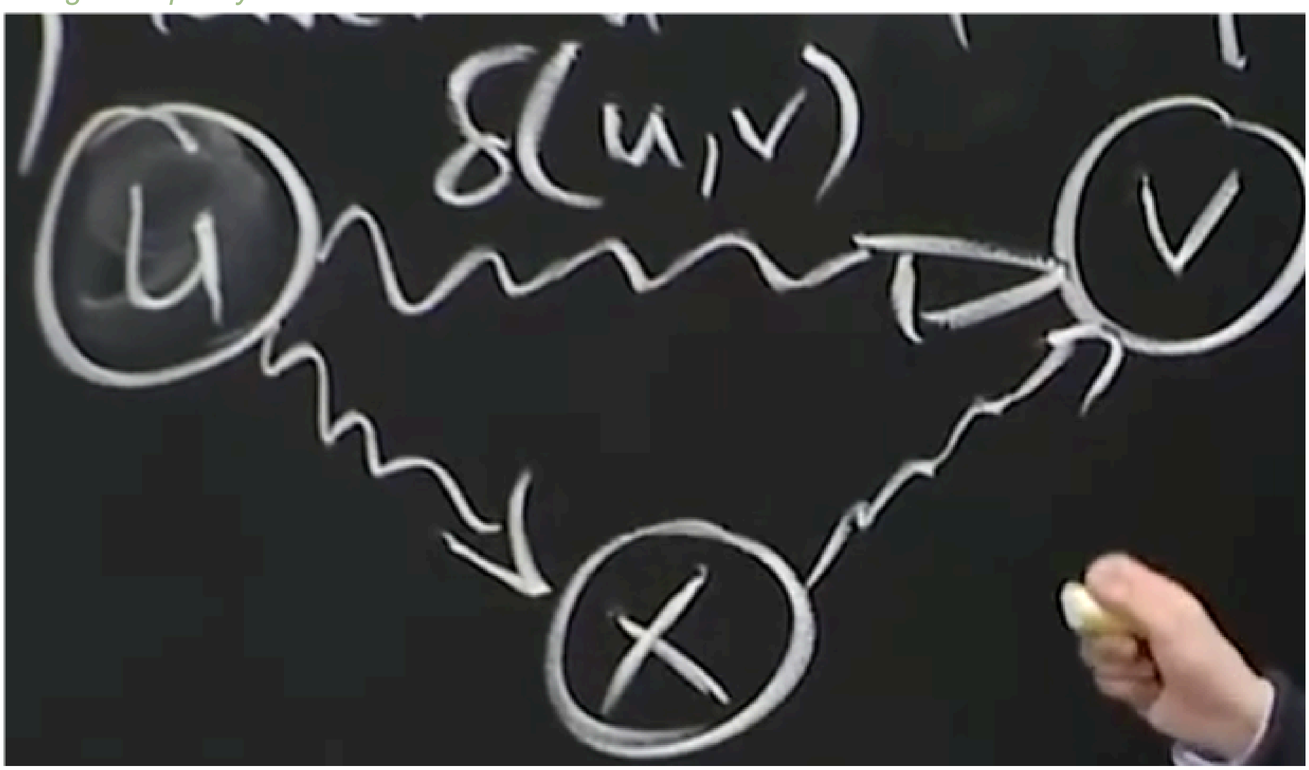
(for weigted shortest-path, only need P(v) for v with finite $\delta(s,v)$)
-Init P empty , P(s) = None
-For each vertex u $\in$ V where $\delta(s,u)$ is finite:
-For each v in Adj+(u): if v not in P and $\delta(s,v) = \delta(s,u) + w(u,v)$, then exist shortest path that uses(u,v), so set P(v) = u.

DAG Relaxation

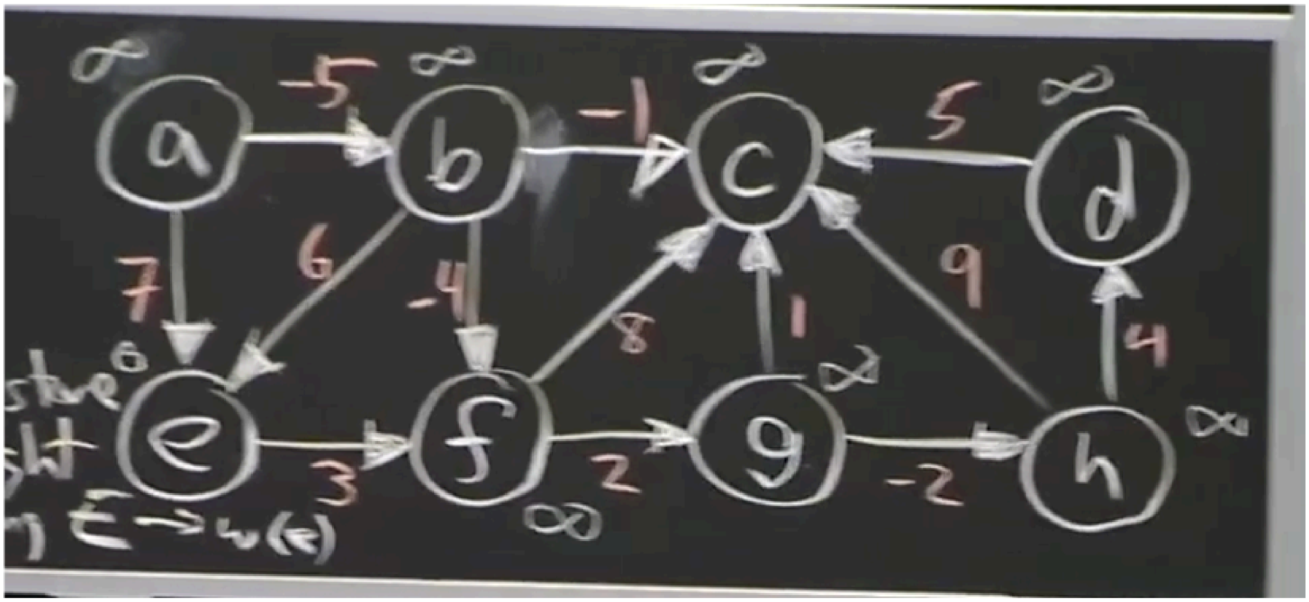
- **the core think of DAG Relaxation Algorithm**

```
Just like iterate all paths and find the minimum path, but on all cross path,
the repeated edges just counted once => store counted path weight.
(the core thought of Dynamic Rrogramming)
```

like a greedy algorithm

- DAG Relaxation Algorithm
  maintain distance estimates(预估) *d(s,v)*(init infinite)
  estimates upper-bound $\delta(s,v)$, gradually lower until equal.
- *triangle inequality:*
  

  $\delta(u,v) <= \delta(u,x) + \delta(x,v)$
  if (u,v) $\in E$, s.t. d(s,v) > d(s,u) + w(u,v)
  "relax"e dge ny lowering d(s,v) to w(u,v)

- Relaxation is Safe
  each d(s,v) is weight of some path from s to v or infinite
  Relax(u,v) => assign d(s,v) to weight of some path.
- process
  => Sets d(s,v) = infinitem then set d(s,s) = 0, process each vetex u in a *topological sort order.*
  => for each outgoing neighbour v in Adj+(u): if d(s,v) > d(s,u) + w(u,v): relax(u,v), i.e. set d(s,v) = d(s,u) + w(u,v)
  

  => Claim: At end, all d(s,v) = $\delta(s,v)$
  can be proofed by induction.