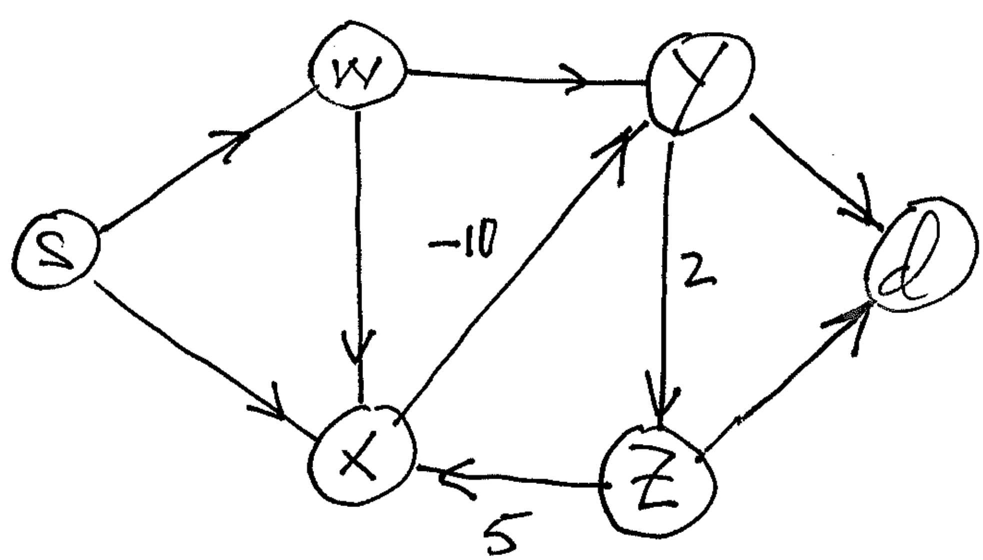
cse 101 3-12-26

Pa8: ext. 1 last day

Ex. negative weight cycle reachable from source $s \in V(G)$.



$w(x, y, z, x) = -3$

- Dijkstra: outlaw Negative Weight edges.

- BellmanFord: detects a neg. Weight cycle, reachable from source, returns false. returns true otherwise

Common Infrastructure

$P[x]$: Parent/Predecessor of x.

$d[x]$: estimate of $\delta(s,x)$

Predecessor subgraph:

$$V_p = \{x \in V \mid P[x] \neq nil\} \cup \{s\}$$

$$E_p = \{(P[x], x) \mid P[x] \neq nil\}$$

find shortest paths by

PrintPath(G, s, x).

Sub-routines

Initialize(G, s)

1.) for all $x \in V$
2.) $\quad d[x] = \infty$
3.) $\quad p[x] = nil$
4.) $d[s] = 0$

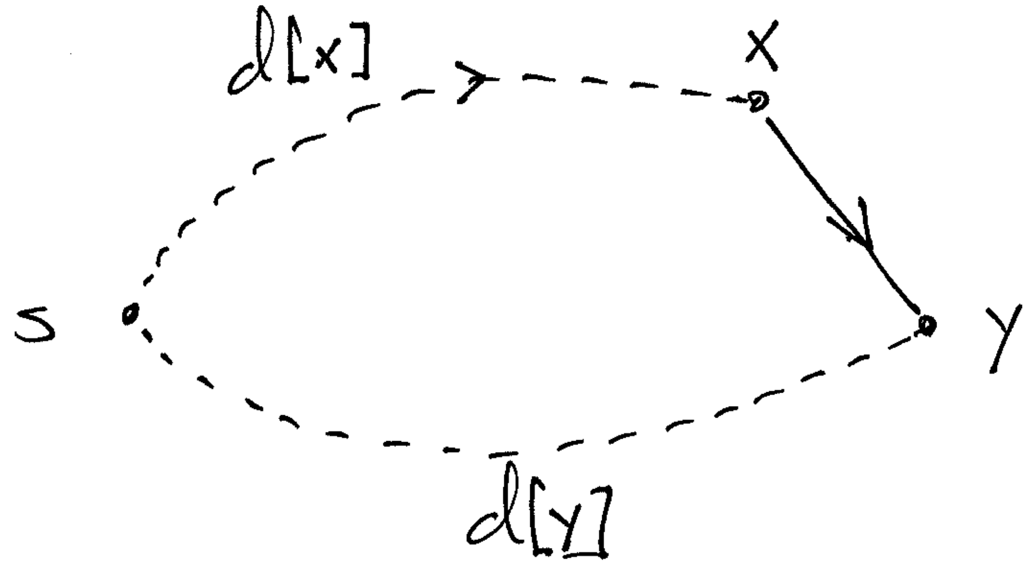
<u>Relax(x, y)</u> Pre: $y \in adj[x]$

1.) if $d[y] > d[x] + w(x, y)$

2.) $d[y] = d[x] + w(x, y)$

3.) $P[y] = x$

<u>Picture:</u>



<u>Lemma 1</u>

let $x \in V$ and suppose that after Initialize(G, s) some sequence of calls to Relax$(\cdot, \cdot)$ results in $d[x]$ being finite. Then G contains an s-x Path of Weight $d[x]$.

Lemma 2

After Initialize(G, s), the inequality

$$\delta(s, x) \leq d[x] \quad \left(\begin{array}{c}\text{for all} \\ x \in V\end{array}\right)$$

is maintained over <u>any</u> sequence of calls to Relax$(\cdot, \cdot)$.

Lemma 3 (Path relaxation Property)

If

$$P : s = x_0, x_1, x_2, \ldots, x_k$$

is a min weight $s - x_k$ path, and the edges of P are relaxed in order

$$(x_0, x_1), (x_1, x_2), \quad , (x_{k-1}, x_k)$$

then $d[x_k] = \delta(s, x_k)$. This remains true regardless of any other relaxation steps that may occur, even if interleaved with the above.

24.1 BellmanFord

see pseudo-code

Runtime: $n = |V|$, $m = |E|$

total cost $= \Theta(nm)$

24.3 Dijkstra

See pseudo-code

Runtime: $n = |V|$, $m = |E|$

P.Q. Q is implemented as heap

- Extract min: $\Theta(\log n)$
  total $= \Theta(n \log n)$
- Relax(.,.): $\Theta(\log n)$
  total $= \Theta(m \log n)$

total runtime :

$$\Theta(n + n\log n + m\log n)$$

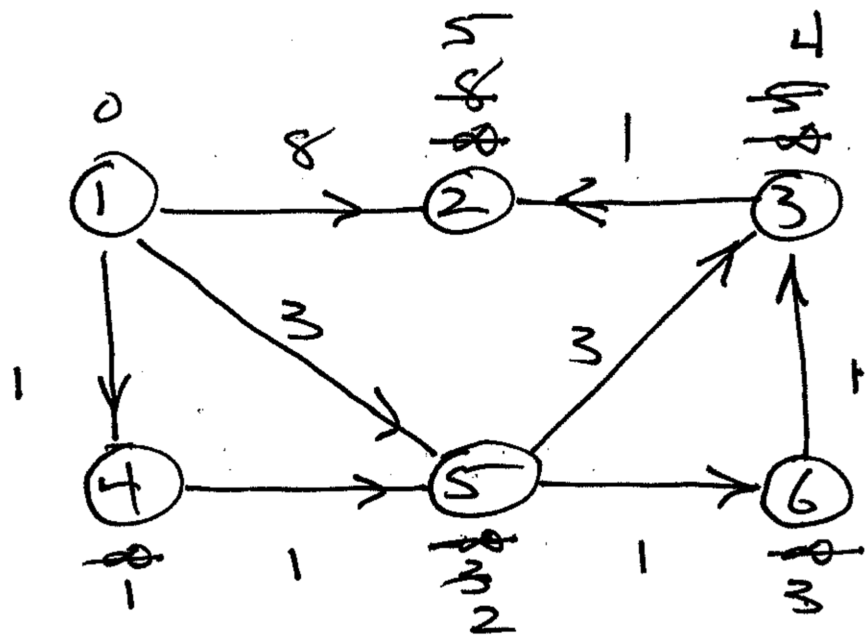
$$= \Theta((n+m)\log n)$$

better than

Bellman Ford

$\Theta(mn)$

Ex. Dijkstra $\boxed{s=1}$

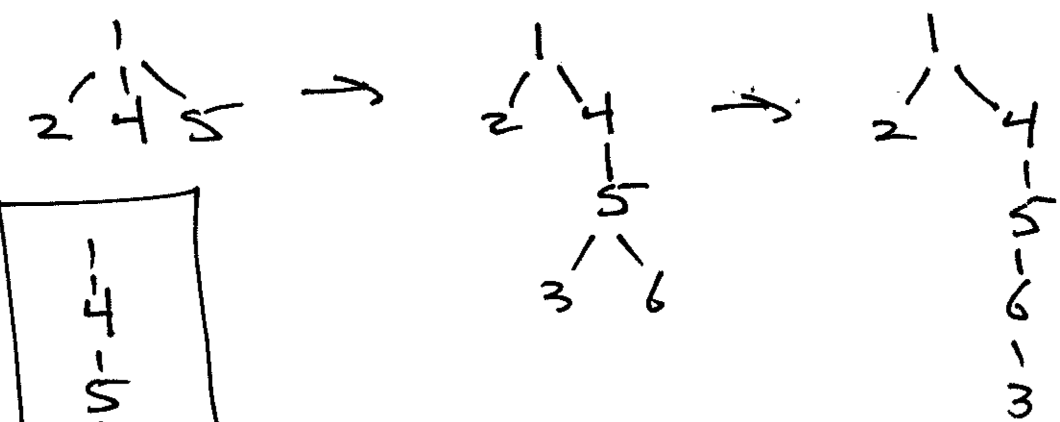


Extract

1
4
5
6
3
2

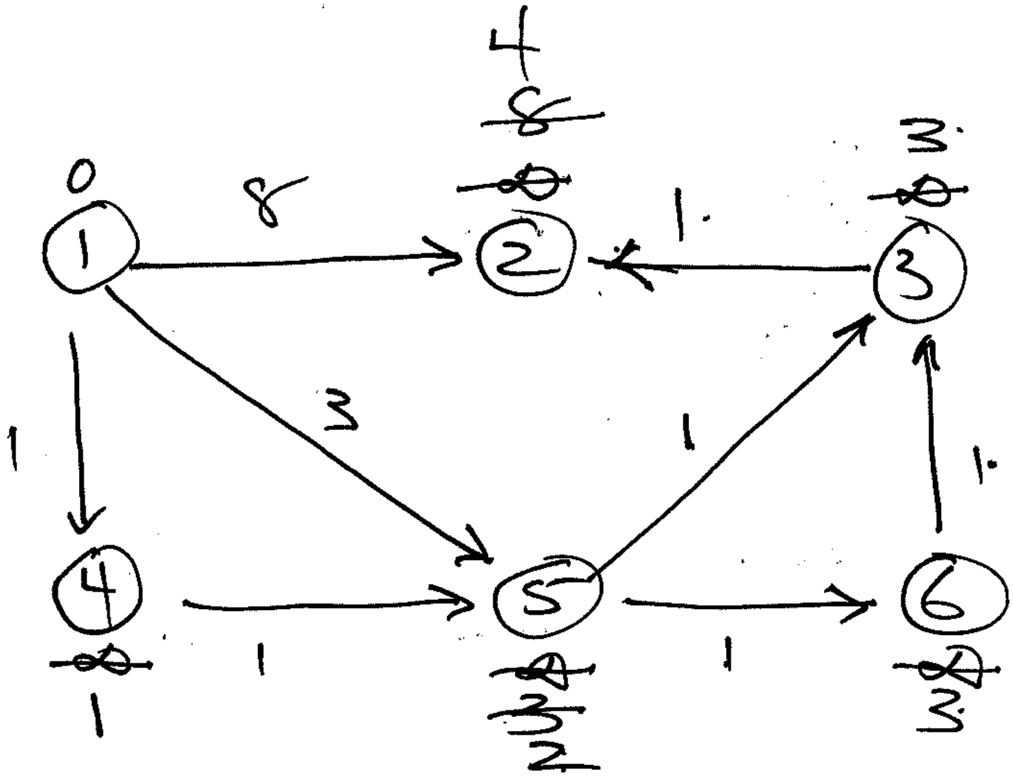
PQ:	~~1~~	~~2~~	~~3~~	~~4~~	~~5~~	~~6~~
d:	0	~~8~~ 5	~~8~~ 4	1	~~3~~ 2	3
p:	n	~~1~~ 3	~~5~~ 6	1	~~1~~ 4	5

Tree



→ $\boxed{1-4-5-6-3-2}$

Ex. Dijkstra $s = 1$



Ext

1
4
5
3
6
2

Pq:	~~1~~	~~2~~	~~3~~	~~4~~	~~5~~	~~6~~
d:	0	~~8~~4	3	1	~~3~~2	3
p:	n	~~1~~3	5	1	~~1~~4	5

tree