LECTURE 7

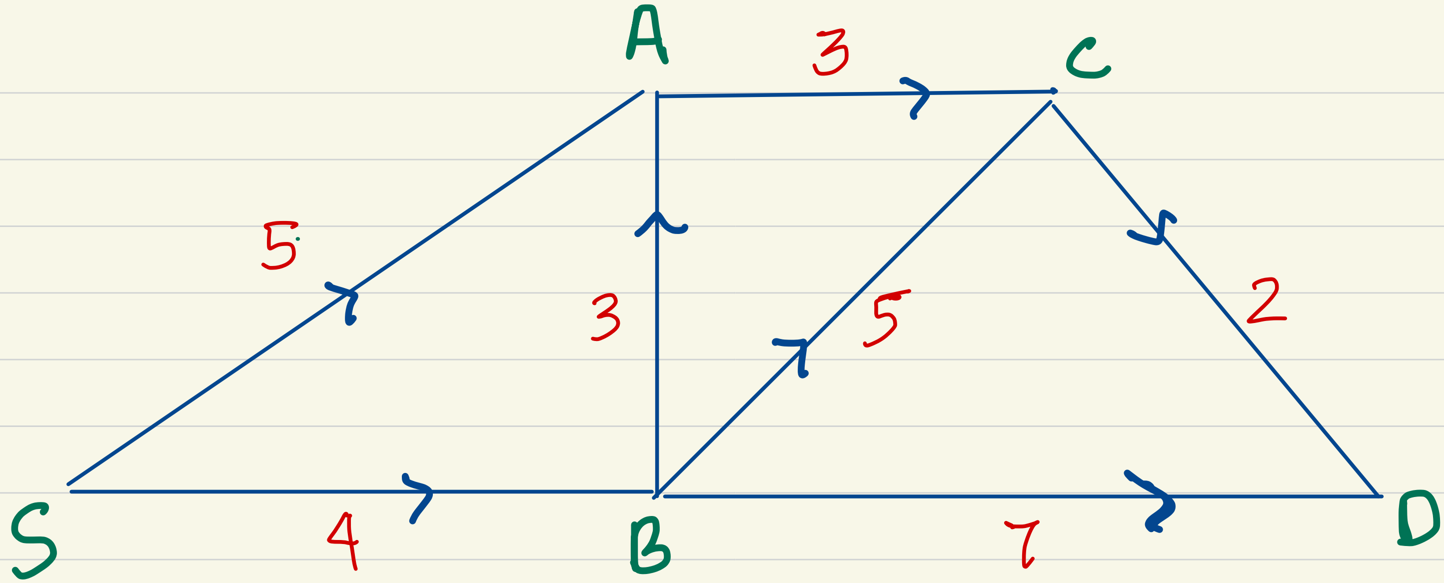
DIJKSTRA's ALGORITHM:

INPUT: 1) GRAPH $G = (V, E)$ with weights w_e for edge E.

2) Start vertex s.

OUTPUT: $dist[u]$ = distance from s to u

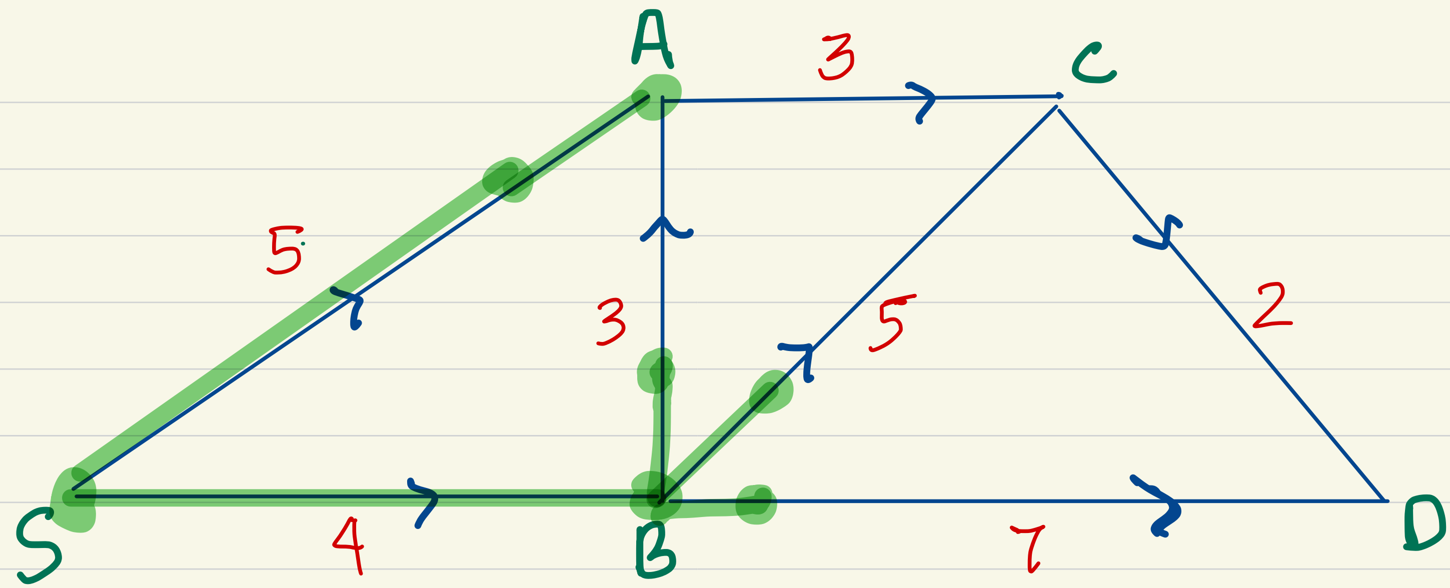
$\forall$ vertices u reachable from u.



DJIKSTRA'S ALGORITHM:

IDEA: Imagine a liquid spill at node S that spreads along edges of graph.

In each step, liquid spreads by distance 1



CURRENT	UPCOMING EVENTS QUEUE
SPILL S t=0	(B,4) (A,5) REACH B at t=4 REACH A at t=5.
REACH B t=4	(A,5) NEW EVENTS (A,7), (B,9), (C,11) (A,5) (B,9) (C,11)
REACH A t=5	(B,9) (C,11) NEW: (C,8)

	(C,8) (B,9)
REACH C t=8	(B,9) NEW: (D,10) [(B,9) (D,10)]
REACH B t=9	

DATA STRUCTURE: Events: (Key, Value)

Key → time

Value → Name of vertex.

1) Insert (vertex, time)

2) DecreaseTime (time t, vertex v)

: decrease the time value for vertex

3) Deletemin (Return element with smallest key)

4) Makequeue

DJIKSTRA'S ALGORITHM (GRAPH $G=(V,E)$, Weights w_e, Source s)

$dist[u] \leftarrow \infty \quad \forall u \in V$

$dist[s] \leftarrow 0$

$Q \leftarrow$ make queue $(V, dist)$

while Q is not empty

    $u \leftarrow Q.\text{deletemin}()$

    for all edge $u \rightarrow v$

        if $dist[v] > dist[u] + w_{uv}$

            $dist[v] \leftarrow dist[u] + w_{uv}$

        $Q.\text{decreaseKey}(v, dist[v])$

$|V|$ delete min

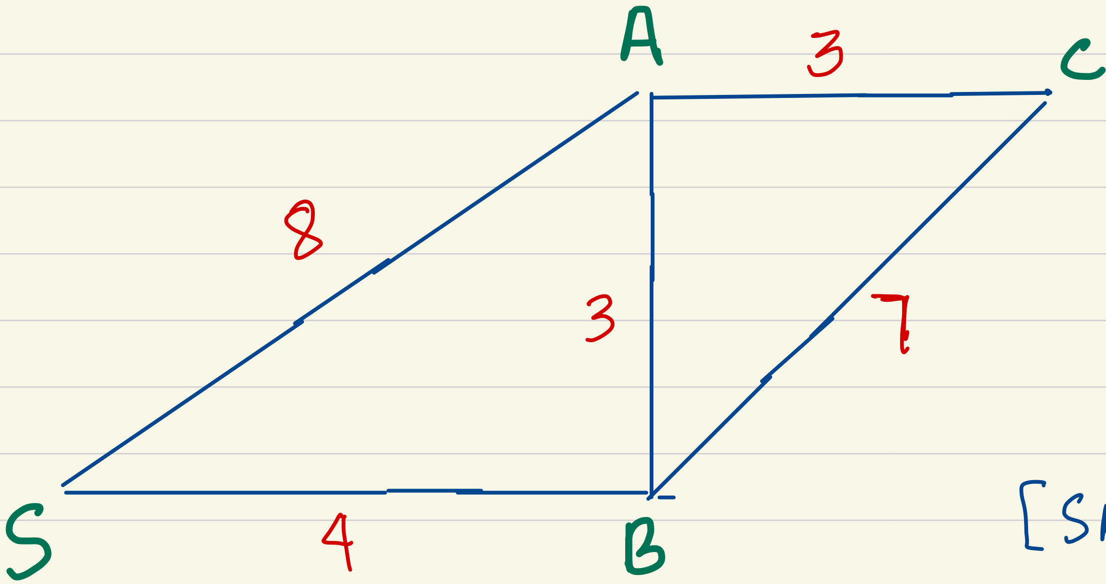
$|V| + |E|$ decrease key / insert

$= \Theta((|V|+|E|)\log|V|)$

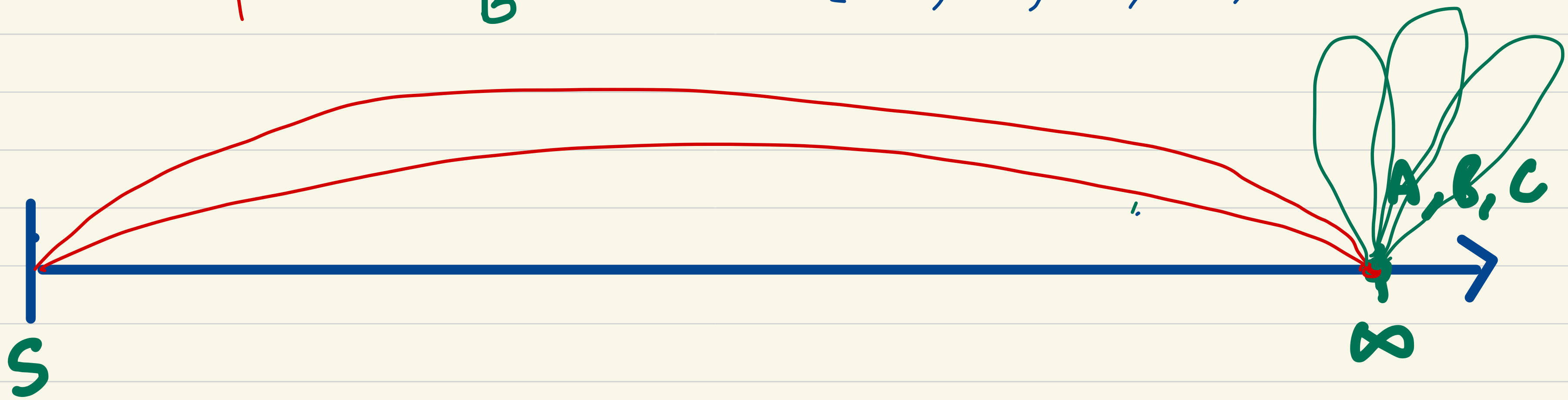
using binary heap

Excercise: Revise binary heaps data structure.

BELLMAN-FORD ALGORITHM:

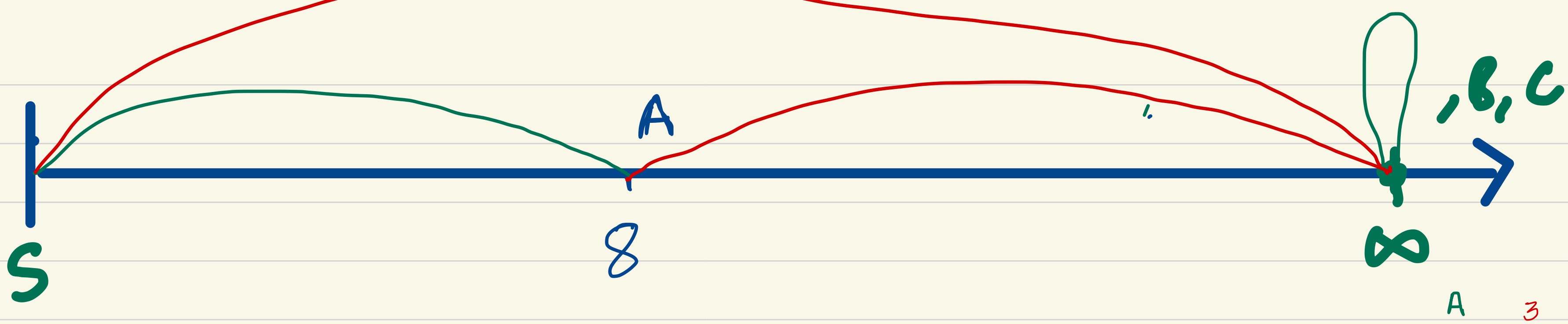


[SA, SB, AB, BC, AC]

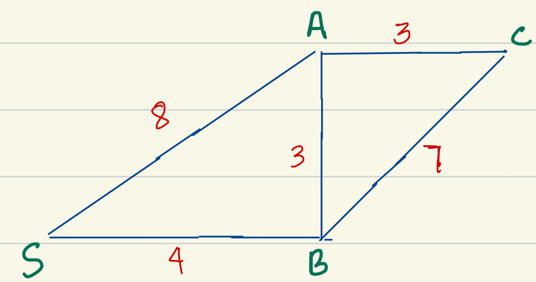


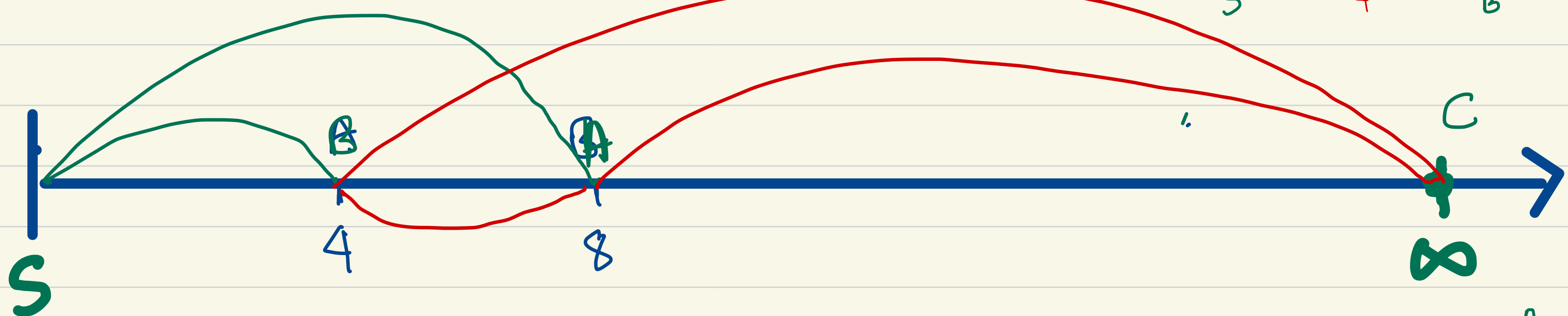
update(S, A) = move right endpoint to unstretch the band

ALGORITHM: Repeatedly update all edges in some order [SA, SB, AB, BC, AC] until there are no stretched bands.

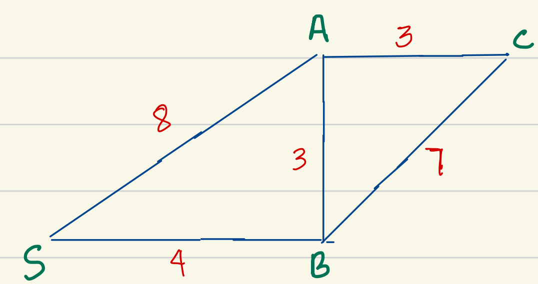


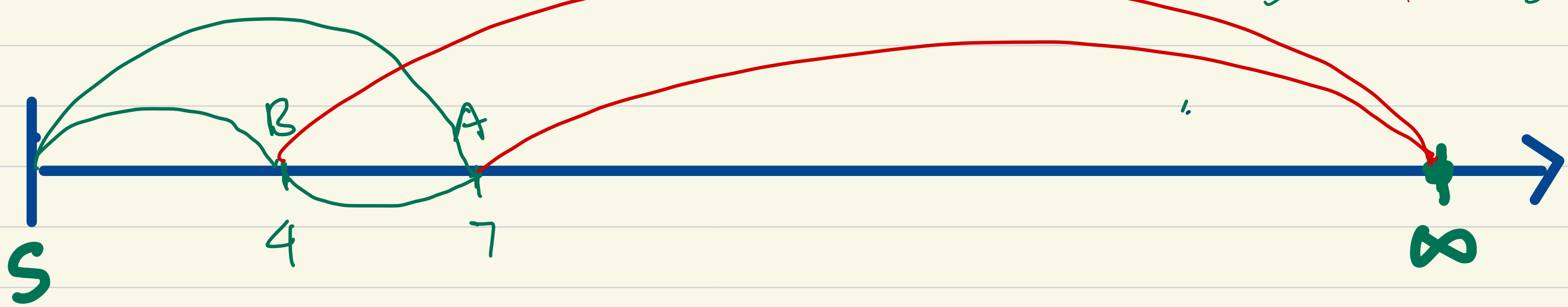
update(S, B)



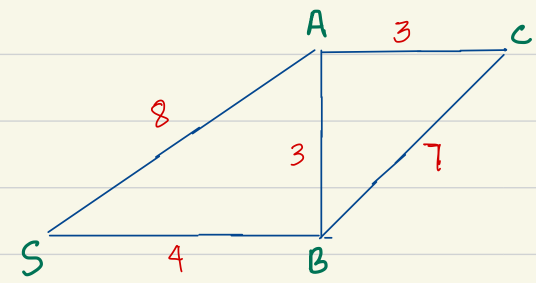


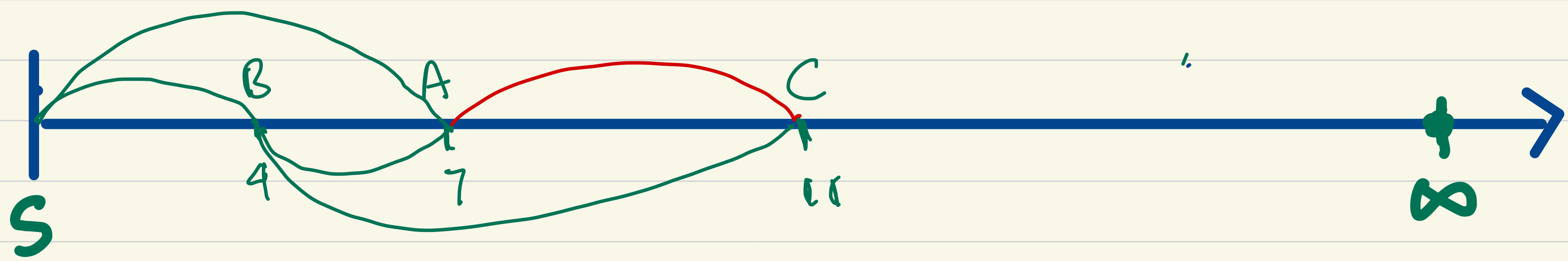
update(A, B)

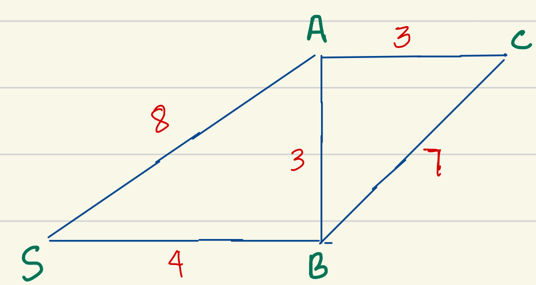


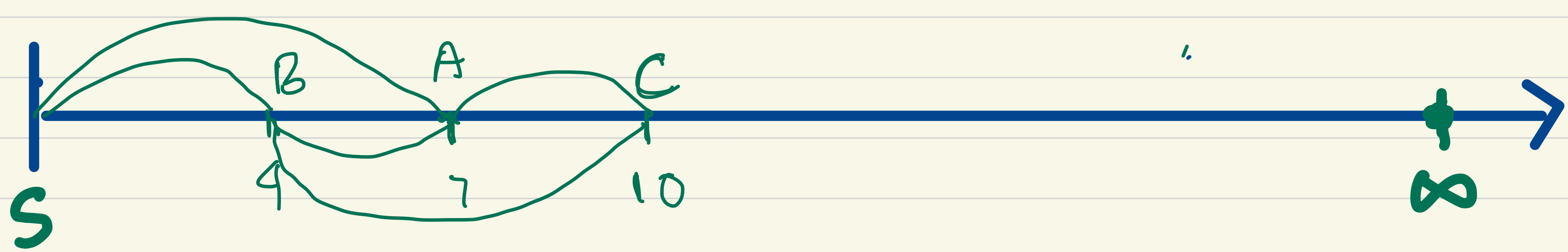


update (B, C)









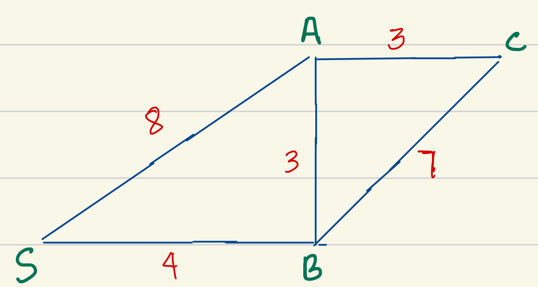
ALTERNATE VIEW ON BELLMAN FORD

Define

$D[\text{vertex } v, \text{ hop } h]$ = length of shortest path that uses at most h hops.

HOP = edge on the path

Example:



$D[A,0] = \infty$ [because there is no path with 0 hops]

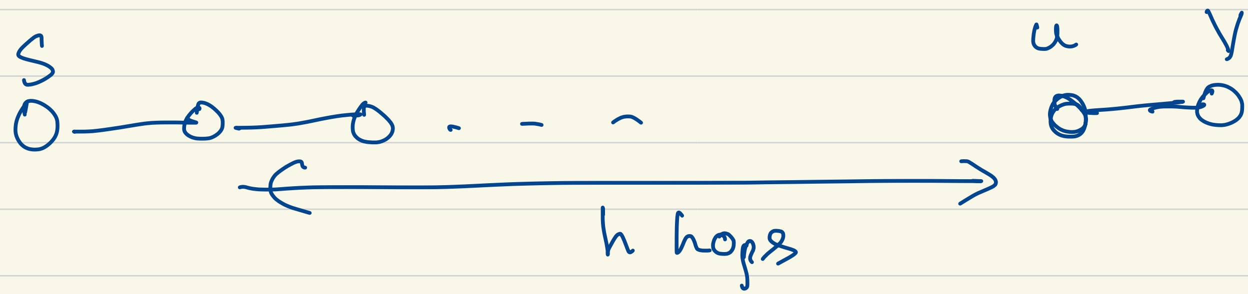
$D[A,1] = 8$ [SA]

$D[C,2] = 11$ [either $S \to A \to C$
$S \to B \to C$]

$D[C,3] = 10$ [$S \to B \to A \to C$]

Observe:

Suppose the following is shortest h hop path from $s \longrightarrow v$



Suppose u is vertex before v

Then: $s \rightsquigarrow u$ is the shortest $h-1$ hop path to u.

[Why? Convince yourself]

Therefore we will have

$$D[v, h] = D[u, h-1] + w_{u \to v}$$

First, we will write a Bellman-Ford algorithm that uses a 2-dimensional array $D[v, h]$

BELLMAN-FORD ALGORITHM (Graph G, weight w_e $\forall e \in E$, source s)

$D[v,0] \leftarrow \infty$ $\forall v$

$D[s,0] \leftarrow 0$

for $h = 1$ to $|V|-1$

$D[v,h] \leftarrow D[v,h-1]$ $\forall v \in V$

for each edge $u \to v \in E$

$D[v,h] = \min(D[v,h], D[u,h-1] + w_{u \to v})$

[$D[v,h]$ = at most h hops
so $h-1$ hops works]

return $D[v, |V|-1]$ $\forall v$.

The above algorithm uses a 2-dimensional array. $D[v,h]$

It computes $D[v,1]$ $\forall v$

then $D[v,2]$ $\forall v$

and so on.

Instead of using separate arrays for $D[:,i]$ for each i, we can observe that the algorithm also works fine if we overwrite the same 1-d array.

So we will have a 1-d array $dist[v]$ $\forall v$

AND overwrite the values for different h.

BELLMAN-FORD ALGORITHM (Graph G, weight $w_e \ \forall e \in E$, source s)

$\quad dist[u] \leftarrow \infty \quad \forall u \in V$

$\quad dist[s] \leftarrow 0$

$\quad$ for $h = 1$ to $|V|$

$\qquad$ for each edge $u \to v \in E$

$\qquad\quad dist[v] = \min(dist[v], dist[u] + w_{u \to v})$

$\quad$ return $dist[:]$

Note $dist[v] = \min(dist[v], dist[u] + w_{u \to v})$ is same as update $(u \to v)$ in the text book.

update $(u \to v)$

$\quad$ if $dist[v] > dist[u] + w_{u \to v}$

$\qquad dist[v] = dist[u] + w_{u \to v}$