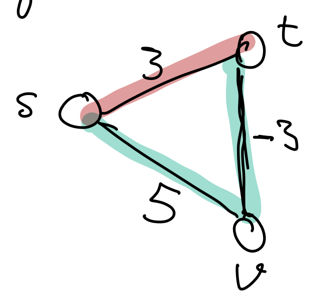
Bellman-Ford

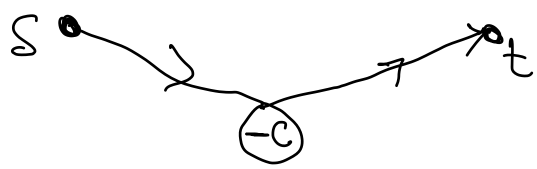
Shortest Path on weighted Graphs. w/t neg. edge weights



$\leftarrow OPT(s \to t) = 2$

$s \to v \to t$

* Neg. Cycle: if there exists a cycle w/t neg. Sum of edges $\to OPT(s \to t) = -\infty$



* Assumption: The graph does not have neg. Cycles

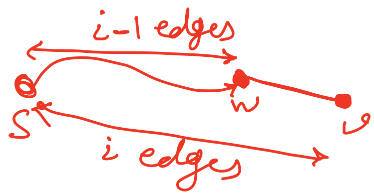
$\Rightarrow$ optimal Path is Simple.

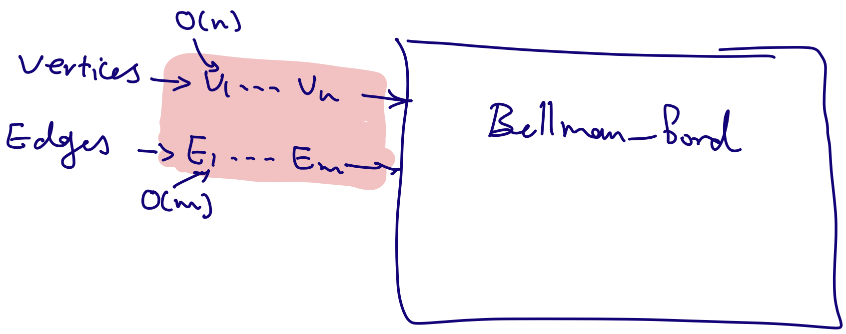
$s \to v$

$OPT(i, v)$: The optimal Solution from s to v that Contains at a most i edges.

$OPT(n-1, t)$: The shortest Path to t, Containing at most $n-1$ edges

$$OPT(i,v) = \begin{cases} \infty & i=0,\ v \neq s \\ \min\Big(OPT(i-1, v), \\ \quad \min\limits_{(v,w)\in E} \big(OPT(i-1, w) + C_{wv} \big) \Big) \end{cases}$$



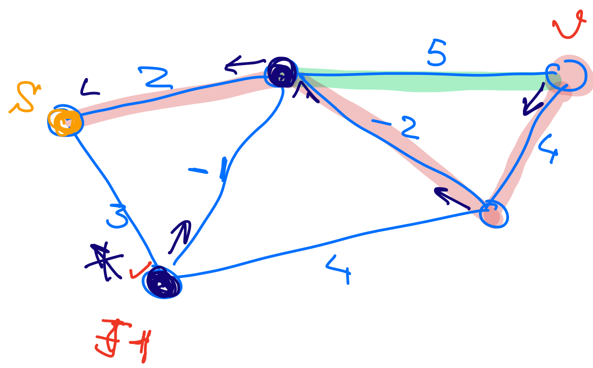


Space: $O(n^2)$

Time: $O(nm)$

```
for v ∈ V : M[0, v] = ∞
M[0, s] = 0
for i = 1 to n-1
    for v ∈ V:
        tmp = min( M[i-1, w] + C_wv )
                ∀ (v,w) ∈ E
        M[i,v] = min(tmp, M[i-1, v])
return M[n-1, *]
```

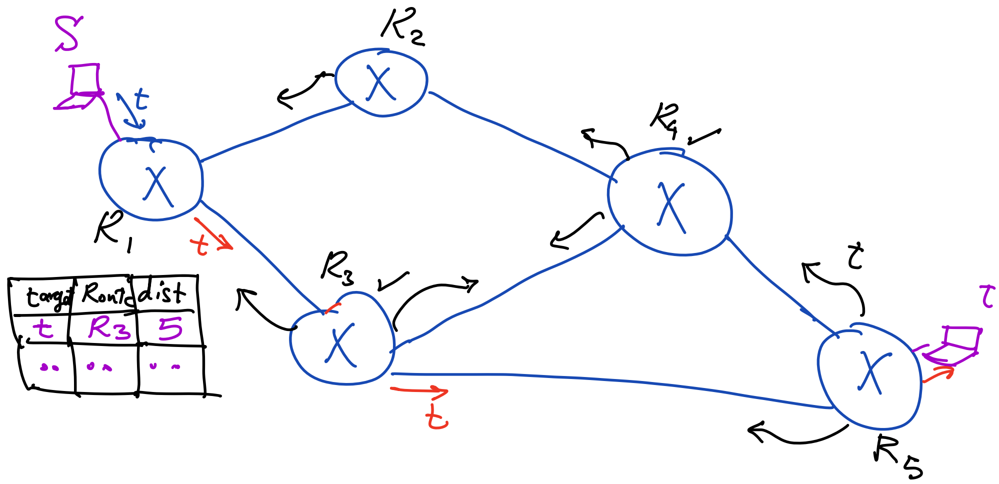
$O(nm)$ $O(m)$



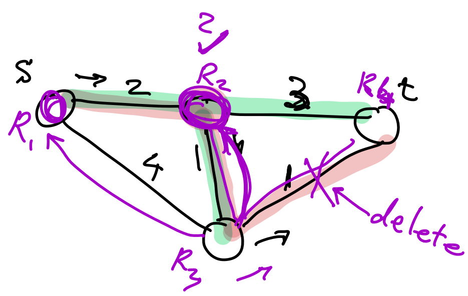
Observation: only the neighbors of the nodes that get updated in the current iteration may get updated in the next iteration

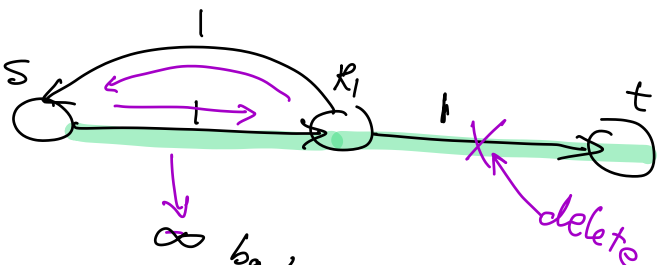
```
for v ∈ V : M[v] = ∞ ; Successor = Null
M[s] = 0 ; U = {s}
for i = 1 to n-1
    U_tmp = {}
    for w ∈ U :
        for (v, w) ∈ E :
            if (M[w] + C_vw < M[v])
                M[v] ← M[w] + C_vw
                Successor[v] ← w
                add v to U_tmp
    U = U_tmp
return M, Successor
```

Distance Vector Protocol



e.g.





∞ back n forth messages are sent.

Resolution: Store the entire Path to t at every intermediate node.

→ Path Vector Protocol

Detect the Neg. cycle:

If $\exists v$, s.t. $OPT(n, v) < OPT(n-1, v)$

$\Rightarrow$ There exists a neg. cycle

otherwise

No neg. cycle.