Single-Source Shortest Path :

Let $G=(V,E)$ is a weighted directed graph ; with weight function $w : E \longrightarrow R$ (mapping edges to real valued weights) (time, cost, penalties, loss)

⊙ **Weight** : The weight of a path $p = \langle v_0, v_1, v_2 - - - v_k \rangle$ is the sum of the weights of its constituent edges :

$$w(p) = \sum_{i=1}^{k} w(v_{i-1}, v_i)$$

⊙ **Shortest path** : A shortest path from vertex u, to vertex v is then defined as any path p, with weight $w(p) = \delta(u,v)$ where

$$\delta(u,v) = \begin{cases} \min \{ w(p) : u \xrightarrow{p} v \} & \text{if there is a path from } u \text{ to } v \\ \infty & \text{otherwise} \end{cases}$$

⊙ **Single source shortest path** : we require to find a shortest path from a given source $\underline{\underline{s \in V}}$; to each vertex $\underline{\underline{v \in V}}$.

(a) <u>Single destination SP</u> : find shortest path to a given destination vertex t from each vertex v. (Reverse direction)

(b) <u>Single pair shortest path</u> : find shortest path from $\underline{u}$ to $\underline{v}$
Solve single source problem for (u).

(c) <u>All pair shortest path</u> : find shortest path from u to v ; for all u, v pairs.
Can be solved by running single source algo once for all $\underline{u}$.

<u>Property</u>: A shortest path between two vertices contains other $\underline{SP}$'s within it.
Subpaths of shortest paths are shortest paths.

Graph $\Rightarrow G = (V, E)$; weight function $w : E \to R$, with $p = \langle V_1, V_2 - - V_K \rangle$ shortest path from V_1 to V_K.

then for any $\underline{i}$ and $\underline{j}$; such that $1 \leq i \leq j \leq K$

let $p_{ij} = \langle V_i, V_{i+1}, - - V_j \rangle$ be the subpath of $\underline{p}$ from V_i to V_j

Then p_{ij} is a shortest path from $\underline{V_i}$ to $\underline{V_j}$

<u>Proof</u>: p is decomposed as $V_1 \overset{p_{1i}}{\rightsquigarrow} V_i \overset{p_{ij}}{\rightsquigarrow} V_j \overset{p_{jK}}{\rightsquigarrow} V_K$

$$w(p) = w(p_{1i}) + w(p_{ij}) + w(p_{jK})$$

Assume there exist p'_{ij} with lesser weight i.e

$w(p'_{ij}) < w(p_{ij})$ $\Longrightarrow$ This new path i.e $(p_{1i}, p'_{ij}, p_{jK})$ gives weight $= w(p_{1i}) + w(p'_{ij}) + w(p_{jK}) < w(p)$

this contradicts.

~~Neg~~ <u>Negative-weight edges</u> : Edges may have -ve edges.

<u>-ve weight cycle</u> : A cycle in a graph with -ve weight.

For any graph $G=(V,E)$ Containing <u>No -ve weight cycles</u> reachable from source s, then for all <u>$v \in V$</u>, Shortest path weight $\delta(s,v)$ remains well defined, [even if it has a -ve value].

(*) If there is a -ve weight cycle reachable from s, then Shortest path weights are <u>NOT</u> well defined.
$\delta(s,v) = -\infty$ (in Case of -ve weight cycle)

$\Rightarrow$ A shortest path Cannot Contain a -ve weight Cycle, nor Can it Contain a +ve weight cycle. ?

(Cycle removal/inclusion Can produce another path with Same Source and destination with lower path weight.)

But what about zeroweight cycles ? we can remove a 0-weight cycle from any path to produce another path with same weight.

(*) Repeatedly remove 0-weight cycles from the path until we get shortest cycle free path

Shortest path algorithm

Dijkstra's algorithm

It assumes that all edge weights in the i/p graph are non negative.

Bellman-ford Algorithm

Allow -ve weight edges in the input graph and produce a correct answer as long no "-ve" cycles are reachable from source

Shortest path tree : for graph $G = (V, E)$; shortest path tree rooted at s, is a directed subgraph $G' = (V', E')$ where $V' \subseteq V$ & $E' \subseteq E$ s.t

1. V' is set of vertices reachable from s in G
2. G' forms a rooted tree with root s, and
3. for all $v \in V'$, the unique simple path from s to v in G' is a shortest path from s to v in G.

Dijkstra Algorithm : (a) Initialization :

INI_SS (G, S)

1. for each vertex $v \in V[G]$
2. do
3. $d[v] \leftarrow \infty$
4. $\pi[v] \leftarrow NIL$
5. $d[s] \leftarrow 0$

An estimate of shortest path from source s to v is saved in $d[v]$; which is via $\pi[v]$ acting as a predecessor.

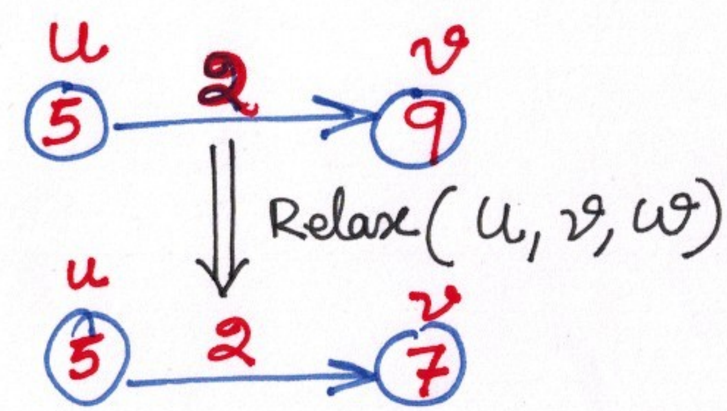
Source estimate.

(b) Relaxation of an edge :

A relaxation of an edge (u,v) may decrease $d[v]$ and update $\pi[v]$.

Relax (u, v, w)

1. If $d[v] > d[u] + w[u,v]$
2. then $d[v] \leftarrow d[u] + w[u,v]$
3. $\pi[v] \leftarrow u$



DIJKSTRA(G, w, s)

```
INI_SS(G, s)
SS ← ∅
Q ← V[G]
while Q ≠ ∅
    do
        u ← Extract_Min(Q)
        SS ← SS ∪ {u}
        for each vertex v ∈ Adj[u]
            do Relax(u, v, w)
```

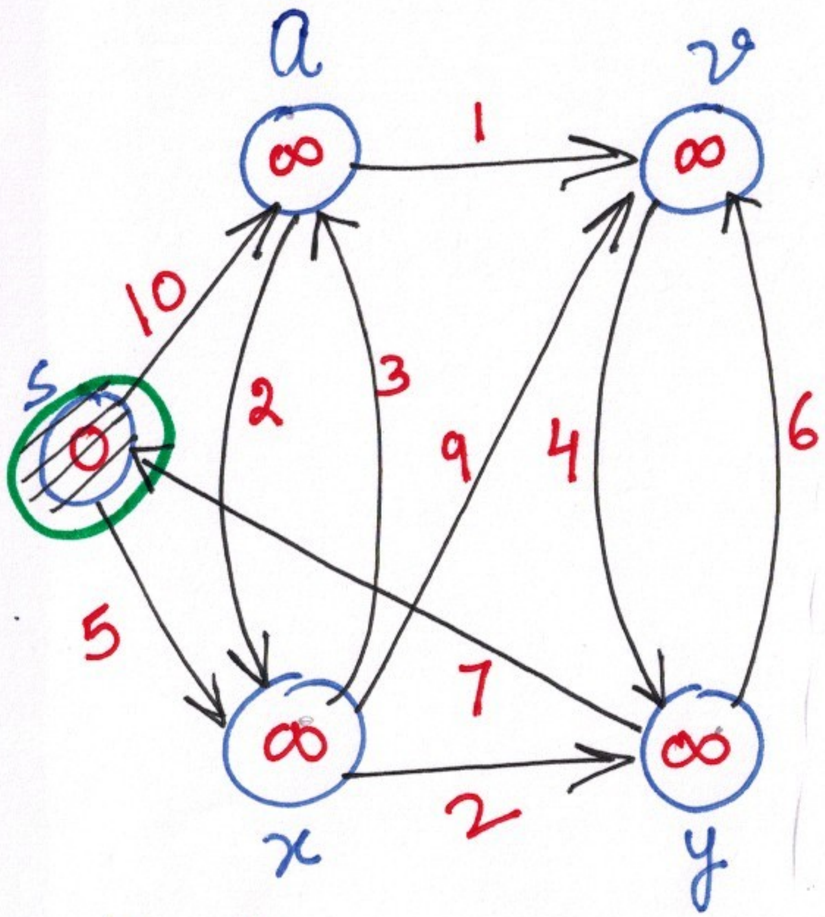
Managed as a min priority queue (Q) of vertices keyed by their (d) value.

Set of vertices whose final shortest-path weights from the source (s), have already been determined.

Adjacent nodes of (u)

This algorithm repeatedly selects the vertex $u \in V - S$ with the minimum shortest path estimate; Add u to SS; and relaxes all edges leaving u.

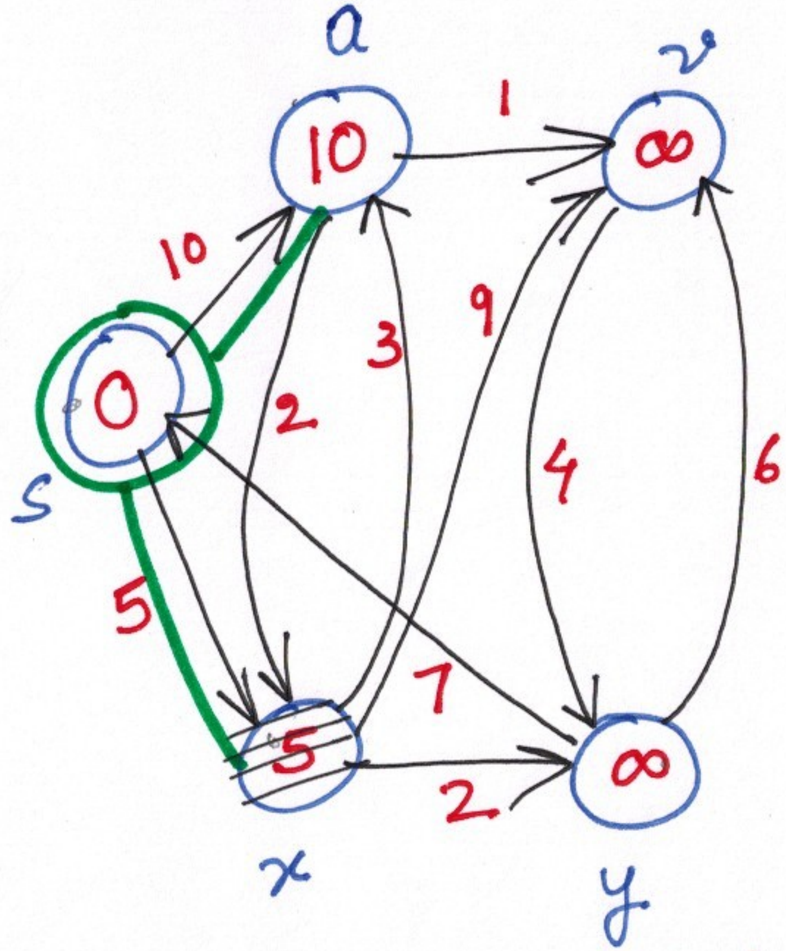
Example:



$u = s$

Relax
sa & sx

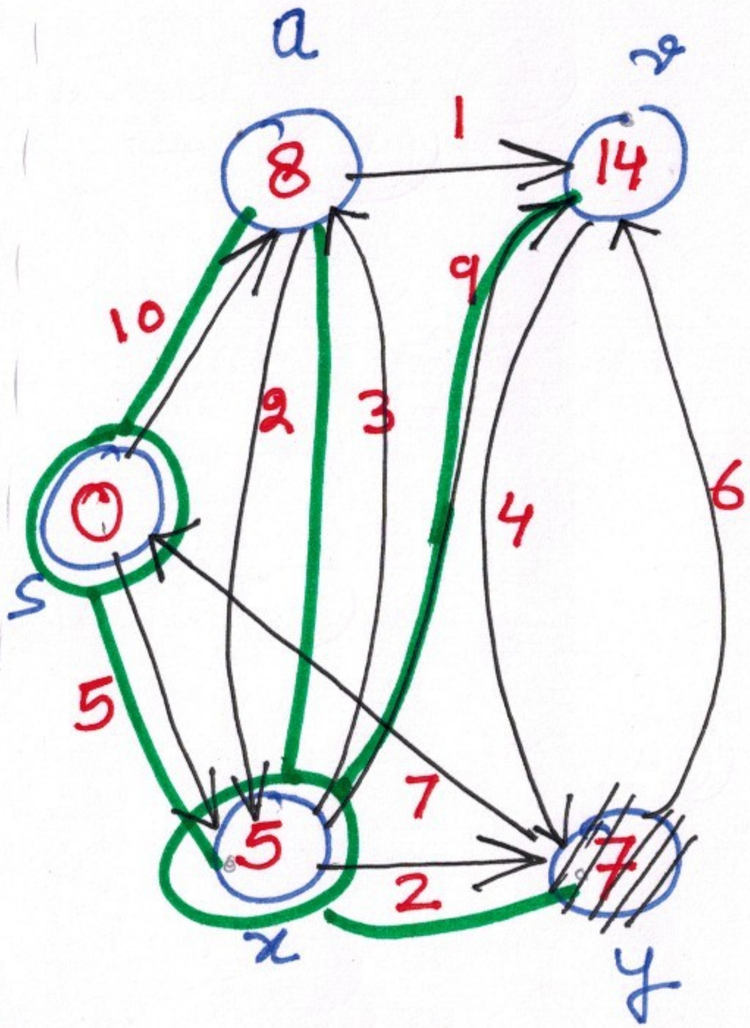
A



$u = x$

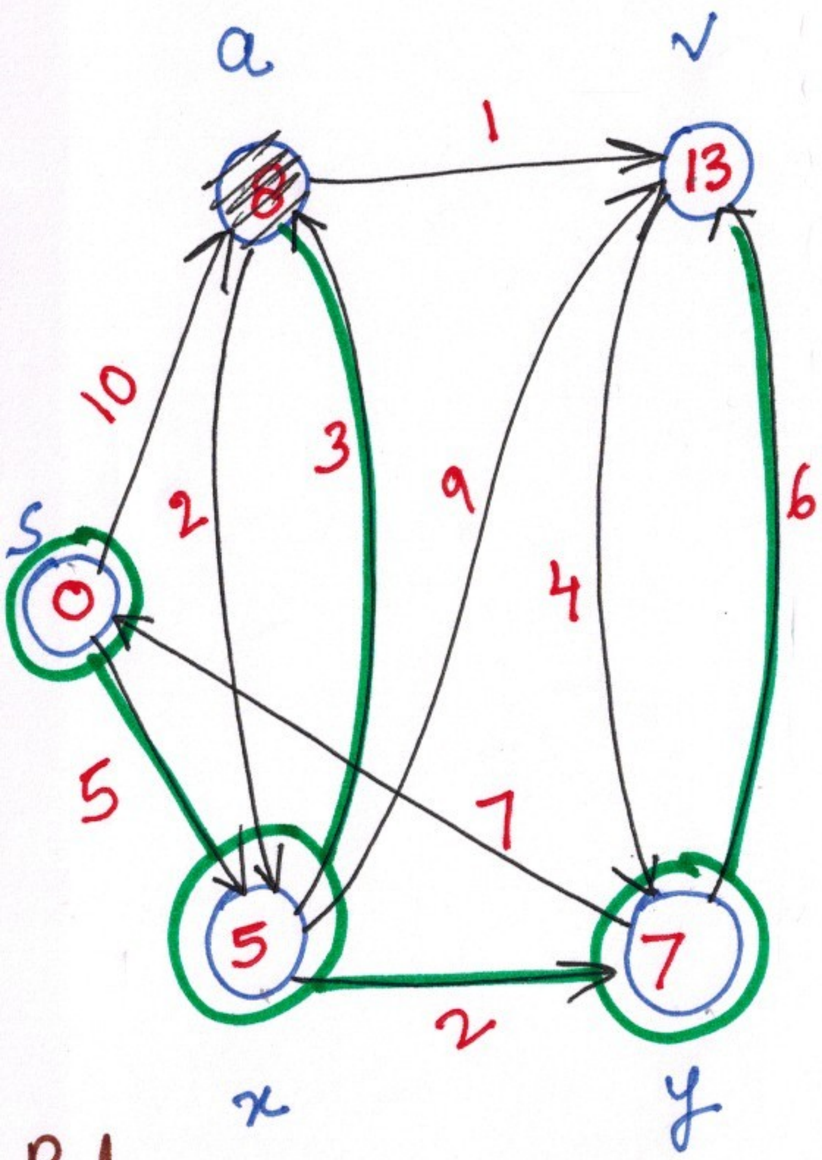
Relax
xa, xv, xy

B



$u = y$

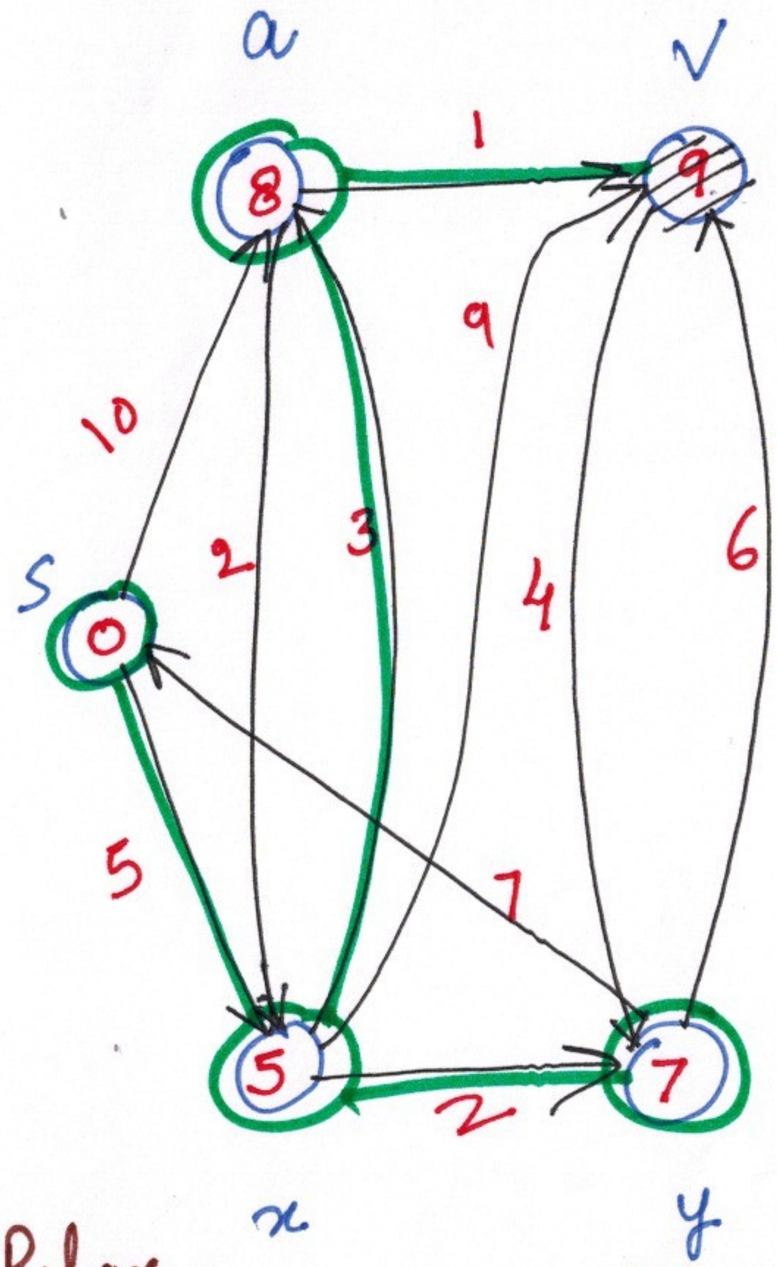
C



Relax
yv & ys

$u = a$

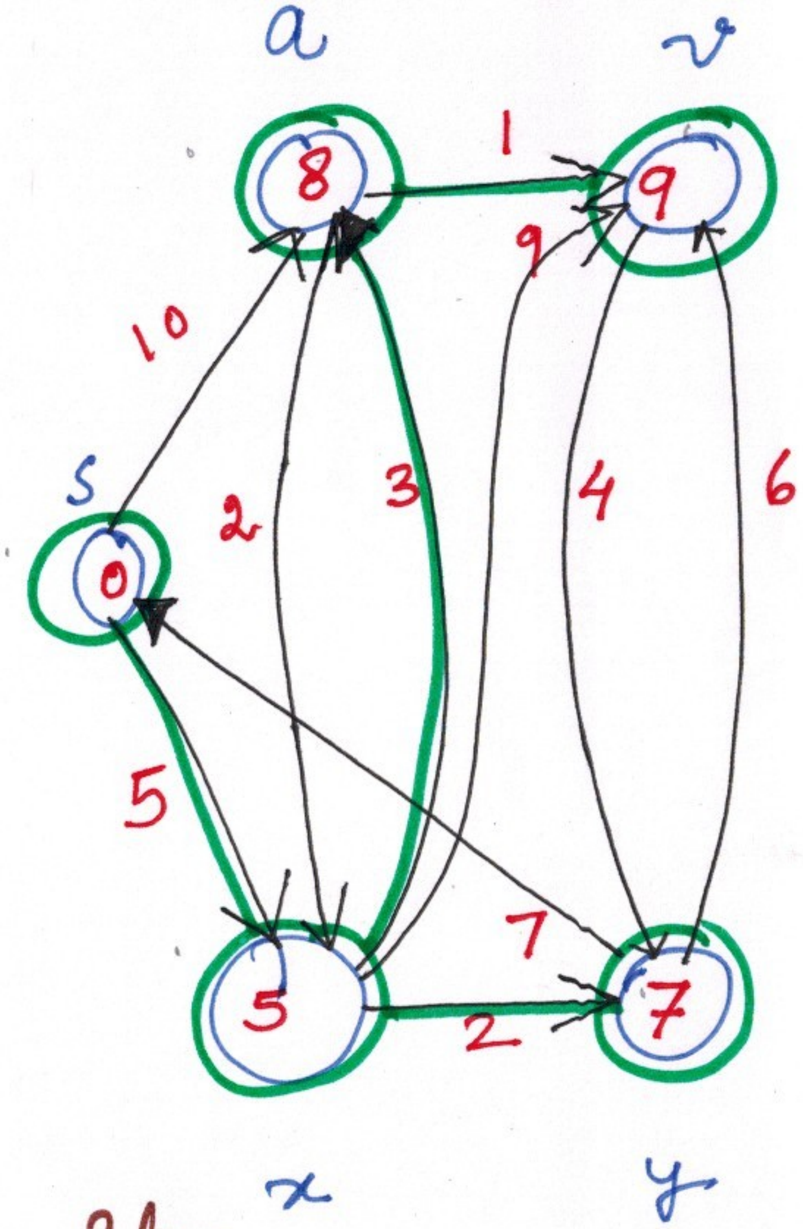
D



Relax
ax & av

$u = v$

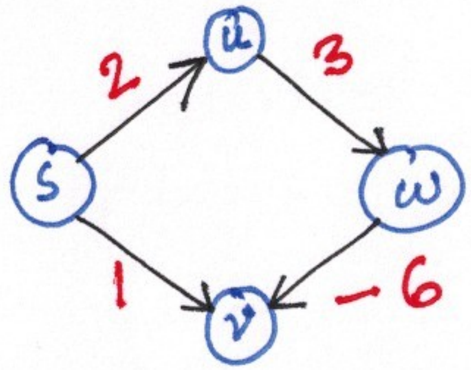
E



Relax
vy

F

Why Dijkstra fails : ① The key assumption is that the shortest path from s to v is the single edge path.



② This is false in case of -ve edges

③ Since a path may start on an expensive edge but compensate with subsequent edges with -ve cost.

Dijkstra style greedy algorithm do not work here.

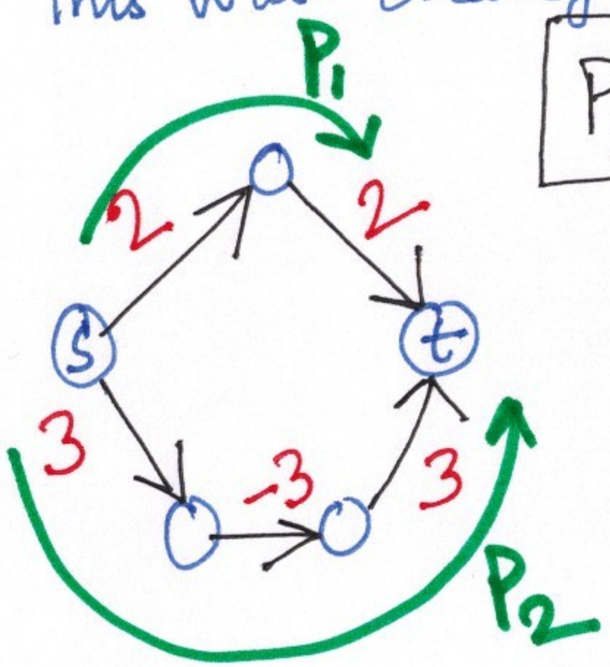
One can always convert -ve weights to +ve weights : Then apply Dijkstra.

$$\boxed{C'_{ij} = C_{ij} + M} \quad \forall (i,j) \in E$$

(*) Add the smallest weight to all edges.

(*) Can this provide correct answer..?

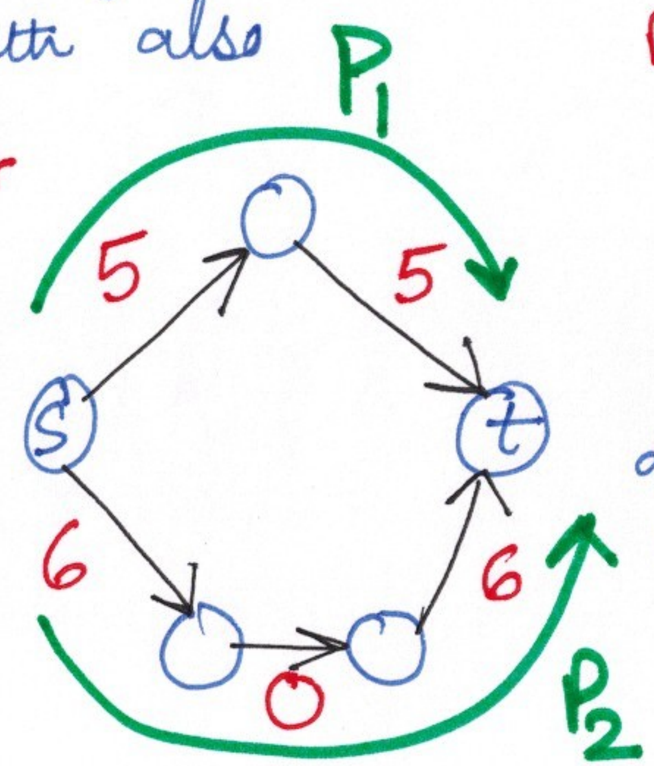
This will change the shortest path also



$P_1 = 4$	$P_2 = 3$

Better

$\xrightarrow{(+3)}$



Better

$P_1 = 10$	$P_2 = 12$

Hence this transformation do not help as it will change the graph semantics.

Bellman ford SP algorithm : { -ve edges are allowed, -ve cycles are NOT. }

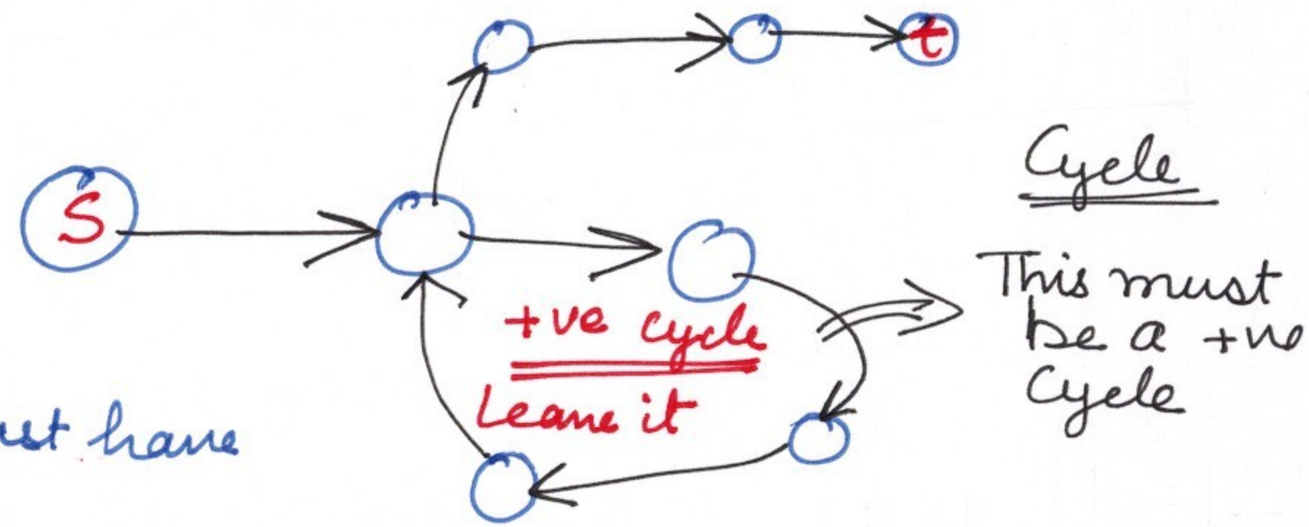
Basic observation :

If $G=(V,E)$ has no -ve cycle, then there is a shortest path from s to t that is a) Simple b) and hence has atmost $|V|-1 = (n-1)$ edges.

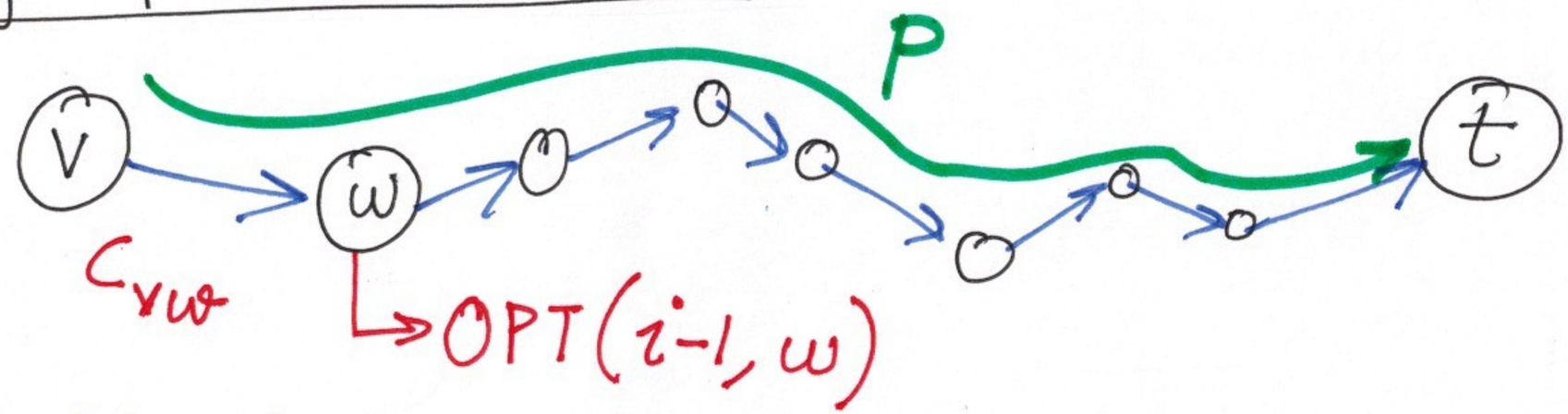
No repetion of nodes.

Proof by Contradiction :

1) Let S.P is not simple.
2) Hence node repetition occurs.
3) Hence cycle is there ; and it must have to be a +ve cycle.
4) Just leave that cycle from your SP and you still get a path from s to t even shorter and with fewer edges.



Setting up the recurrance:



→ $OPT(i, v)$ ⟹ Represents the minimum cost of any path P from v to t; using atmost i edges

Since -ve edges are present adding edges may compensate the cost

Hence our original problem's solution is $OPT(n-1, s)$

Minimum cost of st path using atmost $(n-1)$ edges.

how to derive a recurrence relation in terms of smaller subproblem.

Here $OPT(i, v)$ $\Longrightarrow$ Let it represent path P

Either P, uses atmost $(i-1)$ edges

$\Downarrow$

$$OPT(i, v) = OPT(i-1, v)$$

or P uses atmost (i) edges

$\Downarrow$

$$OPT(i, v) = OPT(i-1, w) + c_{vw}$$

Hence if $i > 0$ then recursively compute

$$OPT(i, v) = \text{Min}\left(OPT(i-1, v),\ \min_{w \in V} \left(OPT(i-1, w) + c_{vw} \right) \right)$$