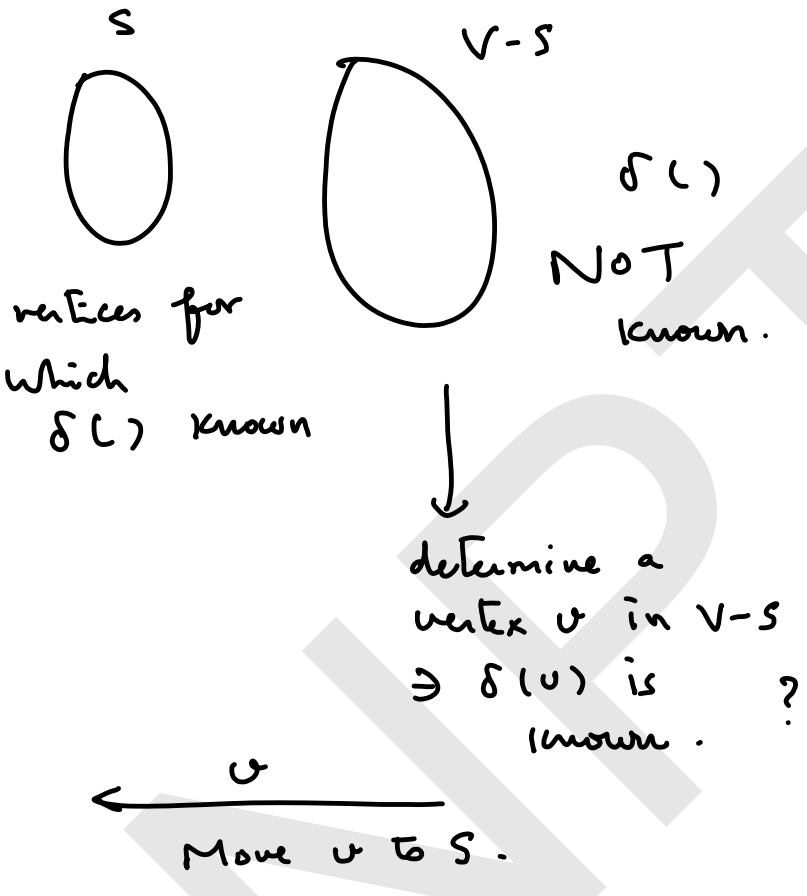
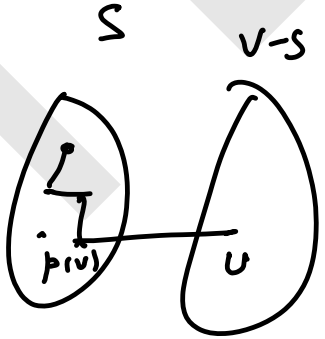
Note Title



for every $v \in V-S$,

$d[v]$ = wt of shortest special path.

(A path from s to v with all vertices other than v are in S)



$p(v) = u$, (u,v) is the last edge of the shortest special path.

v is a vertex in $V-S$

$\Rightarrow$ $d[v] \leq d[u]$ $\forall u \in V-S$.

" v is a vertex with minimum $d[]$ value.

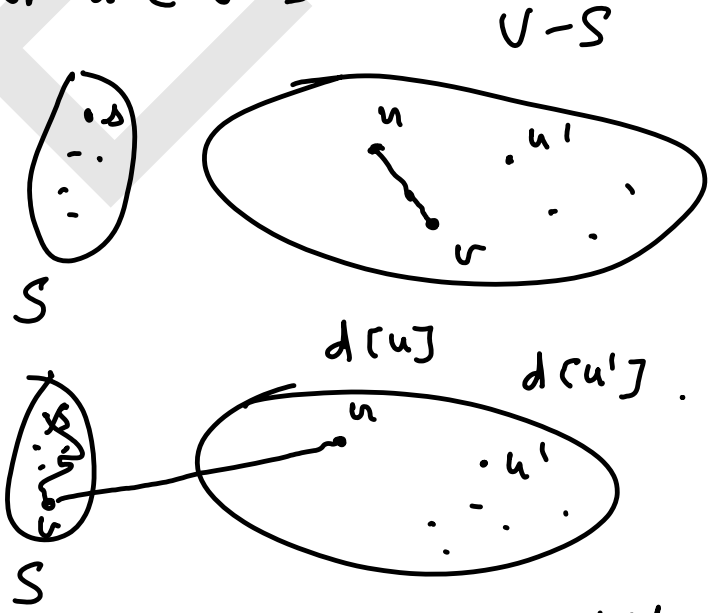
Then $d[v] = \delta(v)$.

$\Rightarrow$ $\delta(v)$ is known for v.

$\Rightarrow$ We move v to S.

Since $S \longrightarrow S \cup \{v\}$

we need to update $d[u]$ values for $u \in V-S$.



$d[v] + w(v,u)$ weight of new special path to u.

if $d[u] > d[v] + w(u,v)$

then

$d[u] = d[v] + w(u,v)$

$p(u) = v$.

$d[u']$ where $u' \notin Adj(v)$

does not change.

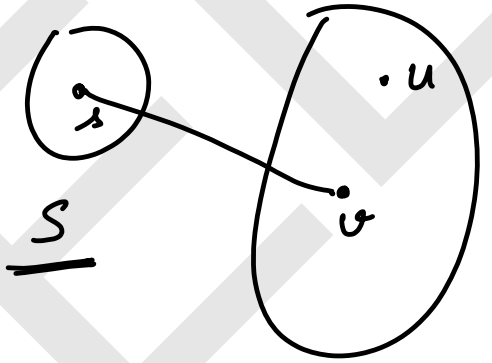
for all $u \in Adj(v)$

if ($u \in V - S$ and $d[u] > d[v] + w(v,u)$)

then

$d[u] = d[v] + w[v,u]$

$p[u] = v$.



$V - \{s\}$.

(s,v) is special path to v from s.

$d[v] = w(s,v)$.

if $u \notin Adj(s)$,

then No special path exist at this point from s.

$d[u] = \infty$.

$d[v] = w(s,v)$ if $v \in Adj(s)$

$= \infty$ if $v \notin Adj(s)$.

$d[s] = 0$

$p(s) = \perp$ (undefined).

$p(v) = s$ if $v \in Adj(s)$.

$= \perp$ if $v \notin Adj(s)$

Initialise $d[\,]$, $p[\,]$.

We represent S, $V-S$ by a Boolean Array.

$In\_s[v] = 1$ if $v \in S$.

$= 0$ if $v \notin S$.

$\Rightarrow In\_s[v] = 0$ for $v \in V-S$

"Move a vertex from $V-S$ to S" ?

Set $In\_s[v] = 1$.

Initialise $(S, V-S) \rightarrow \{s\}, \{V-\{s\}\}$

$In\_s[s] = 1$

$In\_s[v] = 0$ if $v \neq s$.

Dijkstra ($G = (V, E, w, s)$)

// $d[v] = \delta(v)$ as output.

// $p[v]$ is the previous vertex to v in the shortest path.

$(p[v], v)$ is a Bellman edge.

(1) Initialise $(S, V-S)$, $d[\,]$, $p[\,]$.

(2) while $(S \neq V)$

(2a) Find a vertex v in $V-S$ such that

$d[v] \leq d[u] \quad \forall u \in V-S$.

(2b) Move v to S.

(2c) update $d[u]$, $p[u]$ for every $u \in V-S$.

We expand (2c) as follows.

→ if $(u \in Adj(v))$ and $d[u] > d[v] + w(v,u)$

then

$d[u] = d[v] + w(v,u)$

$p[u] = v$.

---

(3) Return $d[v]$, $p[v]$ $\forall v \in V$

// $d[v] = \delta(v)$; $(p[v], v)$ is the Bellman Edge.

The while loop of step 2 will be executed $(n-1)$ time.

$|V| = n$,

Initially $S = \{s\}$.

Total sum of cost of 2a across all iterations is

$(n-1) + (n-2) + \cdots + 1 = O(n^2)$.

Total cost of 2b across all iterations in $(n-1)$.

Total cost of 2c across all iterations

$= \sum_{v \in V - \{s\}}$ outdegree (v).

$\Rightarrow$ Total cost of (2c) across all iterations is $O(m)$.

The complexity of Dijkstra's Algorithm is

$O(n^2 + m)$.