Complexity -

Graph represented as adjacency list

Priority queue " min heap

$|V| - 1 \rightarrow$ deletion from priority queue

$((v-1) + \underset{\substack{\downarrow \\ \text{no of verifications}}}{E}) \log v$

→ Changing the priority in a min heap takes place atmost $|V|$ times.

Complexity - $\log|V|$.

→ Complexity - $(|V|-1 + |E|) \log V = |E| \log V$.

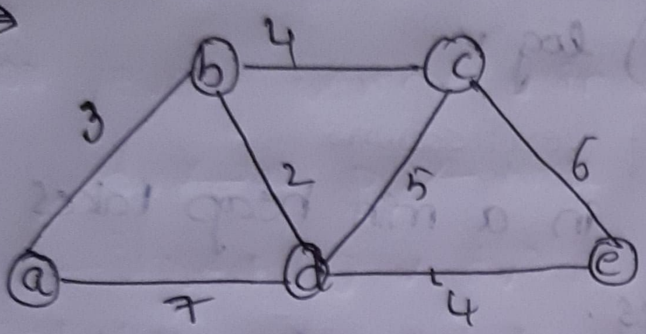
(as $|V| - 1 < |E|$ in a connected graph).

DIJKSTRA'S ALGORITHM (Single source shortest path)

Applicable to undirected and directed graphs with non-negative weights only.

Used in network.

→



a(—, 0)

b(~~source vertex~~, dist from Source vertex)

b(a, 3)

c(—, ∞)

d(a, 7) a, b, c, e

e(—, ∞) d, c - nthng is there in shortest p

Shortest path
a(—, 0)

b(a, 3) → Shortest.
c(—, ∞)
d(a, 7)
e(—, ∞)

Shortest path
a(—, 0)
b(a, 3)

c(b, 7) dist = $\underset{ab}{3} + \underset{bc}{4}$

d(b, 5) a, b, c, e

bd - 3+2 = 5 → min

ad = 7.

e(—, ∞) c, d - nthng in shortest p

Least is d(b, 5). Add to shortes path.

d(b, 5) → 5 is the distance frm source vertex to d through an intermediate vertex b.

Shortest path

a(-,0)

b(a,3)

d(b,5)

c(b,7)

~~b~~ ~~3+4=7~~

~~ed~~ c(b,7) ; c(d,12)

✓ : Shortest path

d c

e(d,9)

dist = (ab + bd) + de

= (3 + 2) + 4 = 9

c(b, 7) - Least → Add it to shortest path

Shortest path

a(-,0)

b(a,3)

d(b,5)

c(b,7)

e(d,9)

c, d

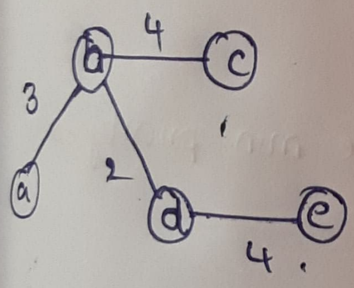
e(c, 13) ab + bc + ce

= 3+4+6 = 13.

✓ e(d,9) = ab + bd + de

= 3+2+4 = 9

Add e(d,9) to shortest path.



Shortest path

a(-,0)

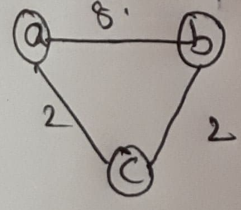
b(a,3)

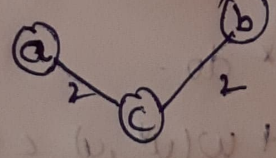
d(b,5)

c(b,7)

e(d,9)

⇒





ALGORITHM -

→ Dijkstra's algorithm for single source shortest paths.

→ Input - A weighted connected graph $G = \langle V, E \rangle$ with non negative weights and its vertex s.

→ Output - The length d_v of a shortest path from s to v and its penultimate vertex p_v for every vertex v in V

Initialize (Q) (initialize priority queue to empty)

for every vertex v in V,

$d_v \leftarrow \infty$; $p_v \leftarrow$ NULL

Insert (Q, v, d_v)

Initialize vertex priority in priority queue

$d_s \leftarrow 0$;

Decrease (Q, s, d_s)

update priority of s with d_s

$V_T \leftarrow \phi$

for i ← 0 to |V| − 1 do

$u^* \leftarrow$ DeleteMin (Q) delete min priority elt

$V_T \leftarrow V_T \cup \{u^*\}$.

For every vertex u in $V - V_T$ that is adjacent to u^* do -

if $du^* + w(u^*, u) < du$

$du \leftarrow du^* + w(u^*, u)$

$P_u \leftarrow u^*$

Decrease (Q, u, du)

dv - distance of vertex from source

Pv - neighbouring vertex

v - vertex

Q - Priority queue

ds - distance from source vertex

V_T - shortest path

$V - V_T$ - Remaining vertices

u^* - deleted one

ery

dv - distance of vertex from sourc

Pv - neighbouring vertex

v - vertex

Q - Priority queue

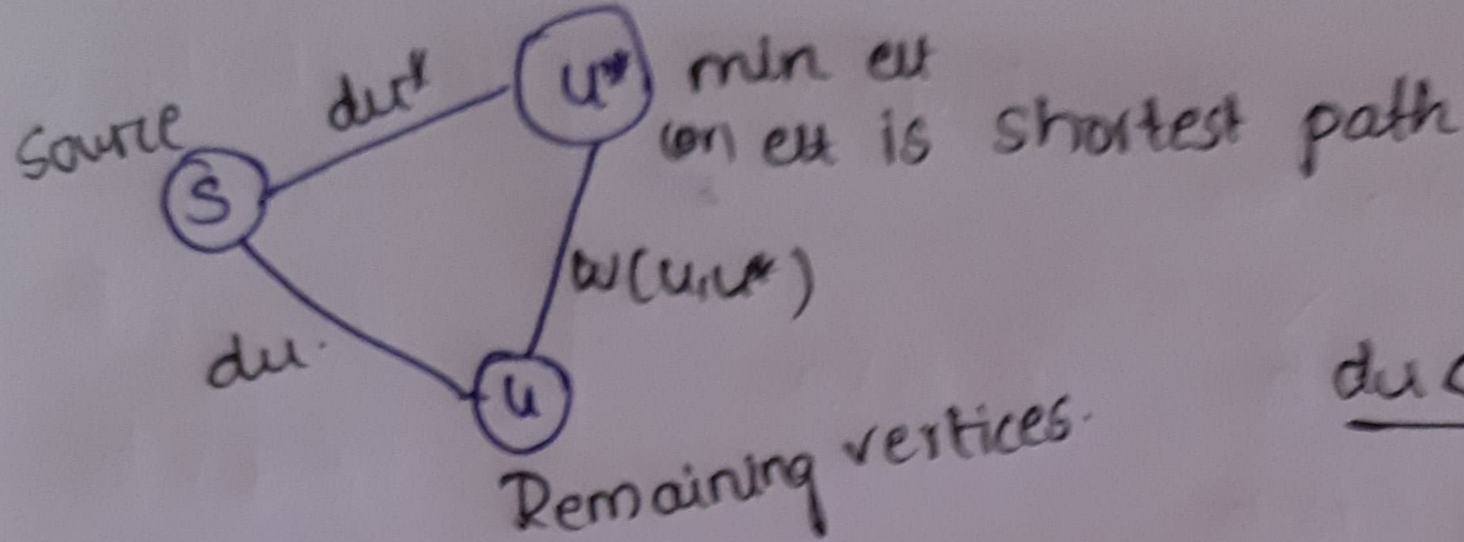
ds - distance from source vertex

V_T - shortest path

y

$V - V_T$ - Remaining vertices

u^* - deleted one



$d_u < d_{u^*} + w(u, u^*)$

↓

take d_u

or else take

$d_{u^*} + w(u, u^*)$

Complexity -

→ Adjacency matrix and priority queue as an unordered array - $\Theta(|V|^2)$

→ Adjacency lists and the priority queue implemented as a min-heap - $O(|E| \log |V|)$.