Breadth first Search :- Unweighted Graph

Input : unweighted graph and start vertex u.

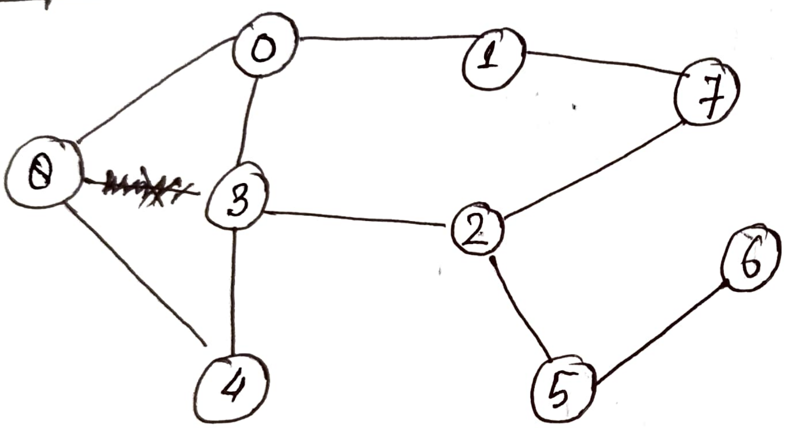
Idea : maintain a set R of vertices that have been reached but not searched and a set S of vertices that have been reached.

set R{} → use FIFO (queue)

Initialization : $R = \{u\}$, $S = \phi$, $d(u,u) = 0$

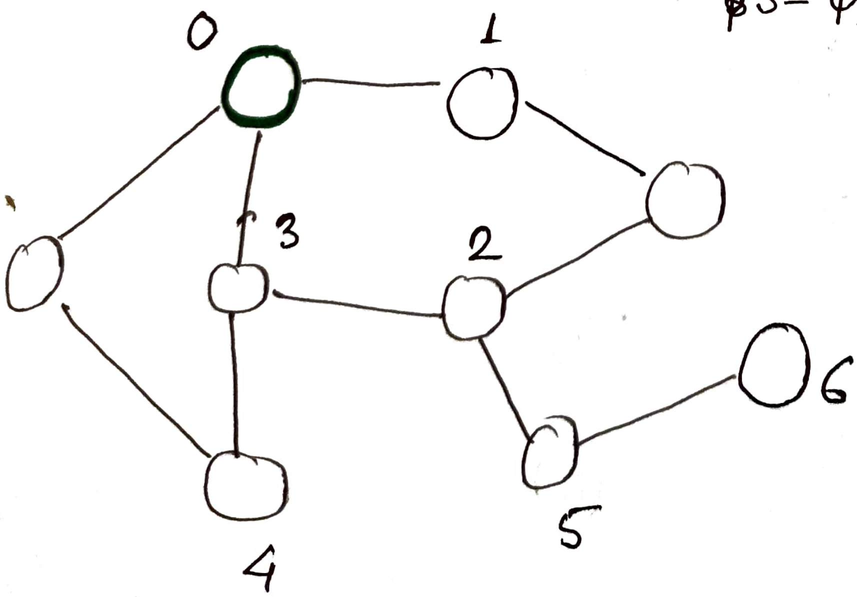
Iteration : $R \neq \phi$, search from the vertex v of R. The neighbors of v not in $S \cup R$ are added to R & assigned distance $d(u,v)+1$, and then v is removed from R & placed in S.

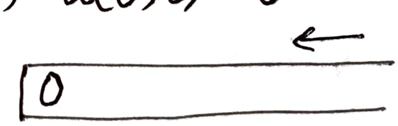
Example →



* green outlined ~~circle~~ refers queue
* Red internal refers visited.

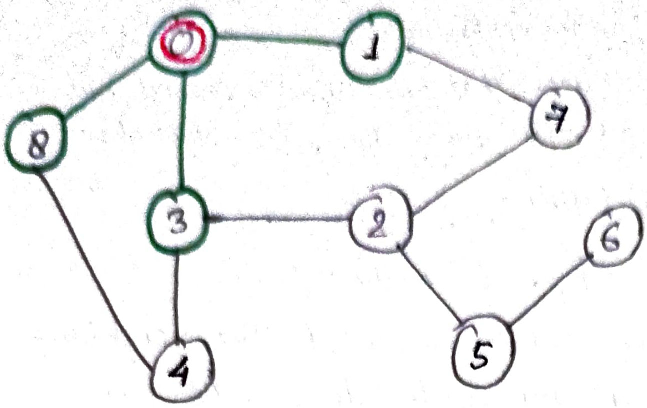
Step 1 :- $R = \{0\}$, 0 is starting vertex Now maintain a queue
$S = \phi$, $d(0,0) = 0$





0 is enqueue.

step 2: search neighbors of 0 vertex & those are placed in queue
0 is visited



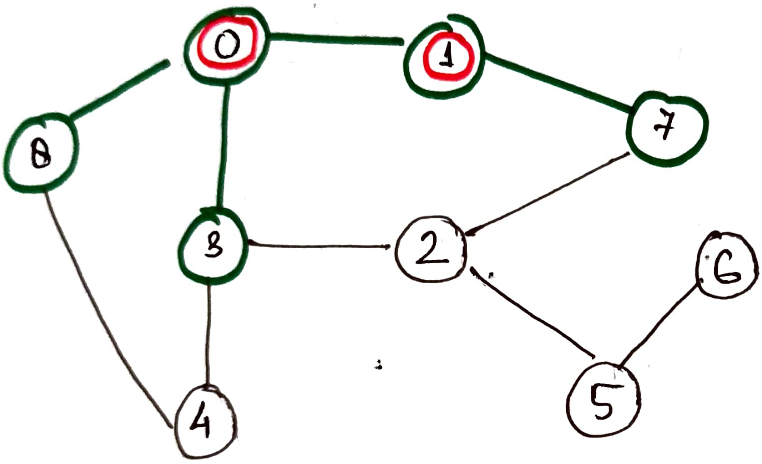
$R = \{0\}$
$S = \{1, 3, 8\}$
$d(0,0) = 0$

1	3	8

Now next node is 1

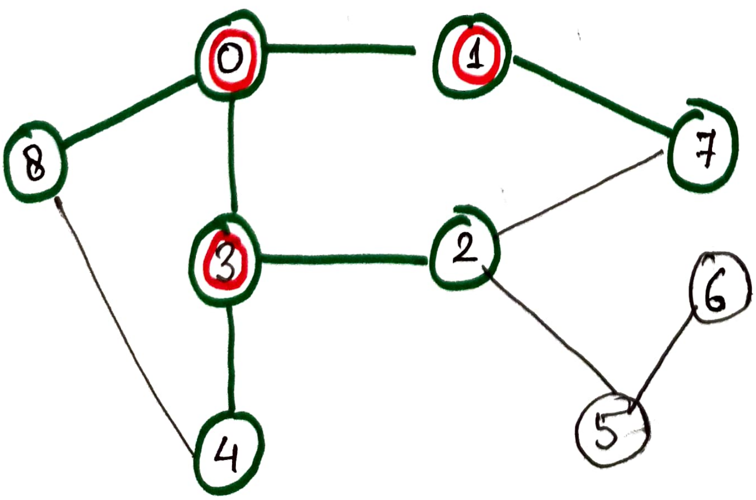
Step 3:- $R = \{0, 1\}$, $S = \{3, 8, 7\}$, $d(0,1) = 1$,
1 is visited

3	8	7



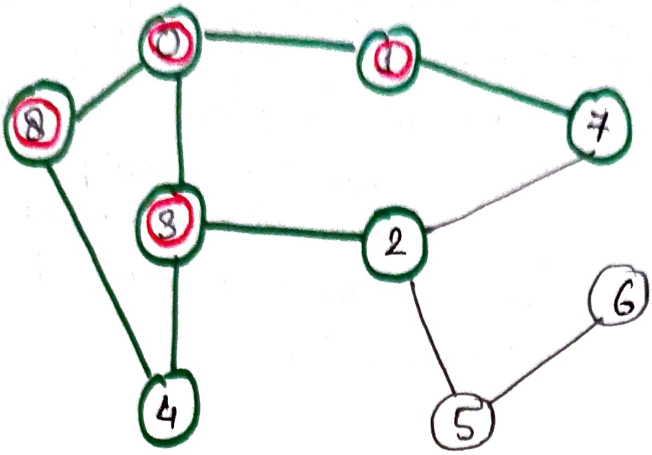
Step 4:- $R = \{0, 1, 3\}$, $S = \{8, 7, 2, 4\}$ $d = (0, 3) = 1$
3 visited

8	7	2	4



Step 5 :- $R=\{0,1,3,8\}$, $S=\{7,2,4\}$, $d(0,8)=1$
8 is visited

| 7 | 2 | 4 |



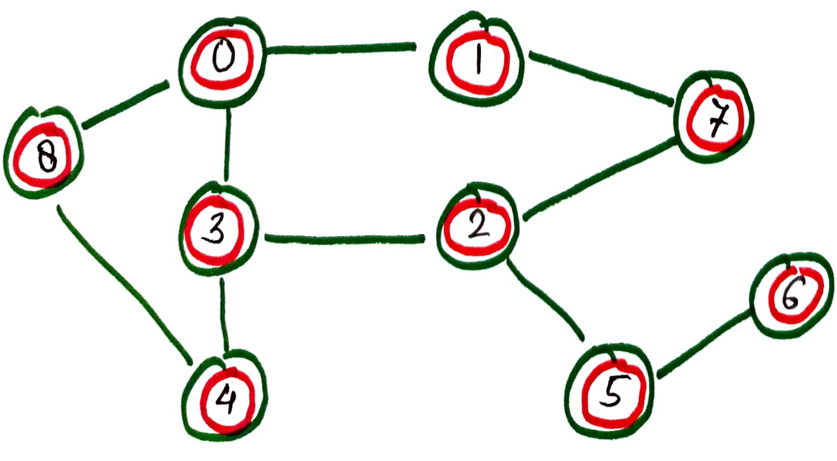
Step 6 → $R=\{0,1,3,8,7\}$,
$S=\{2,4\}$, $d(0,7)=2$, 7 is visited

Step 7 → $R=\{0,1,3,8,7,2\}$
$S=\{4,5\}$, $d(0,2)=2$, 2 visited

Step 8 → $R=\{0,1,3,8,7,2,4\}$
$S=\{5\}$, $d(0,4)=2$, 4 visited

Step 9 → $R=\{0,1,3,8,7,2,4,5\}$
$S=\{6\}$, $d(0,5)=3$, 5 visted

Step 10 → $R=\{0,1,3,8,7,2,4,5,6\}$
$S=\{\}$, $d(0,6)=4$, 6 visited.



Dijkstra's Algorithm — for weighted graph {only positive weight}

Input :- A graph with non-negative edge weights & starting vertex U. The weight of edge xy is $w(x,y)$; let $w(x,y) = \infty$, if xy is not an edge (or not participating)

Idea → Maintain the set S of vertices to which the shortest path from u is known, enlarging S to include all vertices. To do this, maintain tentive distance $t(z)$ from u to each $z \notin S$, being the length of the shortest u, z-path.

Steps →

① Mark ~~sele~~ selected initial node with a current distance of 0 & rest with infinity.

② Set the non-visited node with smallest current distance as current node C.

③ for each neighbor N of current node C: add the current distance of C with weight of the edge connecting C-N. If it is smaller than the current distance of N, set it as the new current distance of N.

④ Mark the current node c is visited.

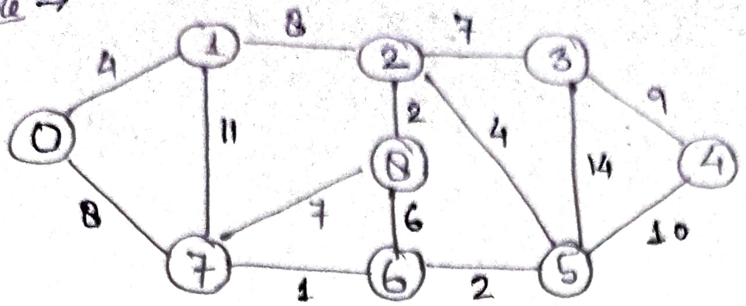
⑤ Repeat the process from step 2.

In Dijkstra's Algo we are using priority queue.

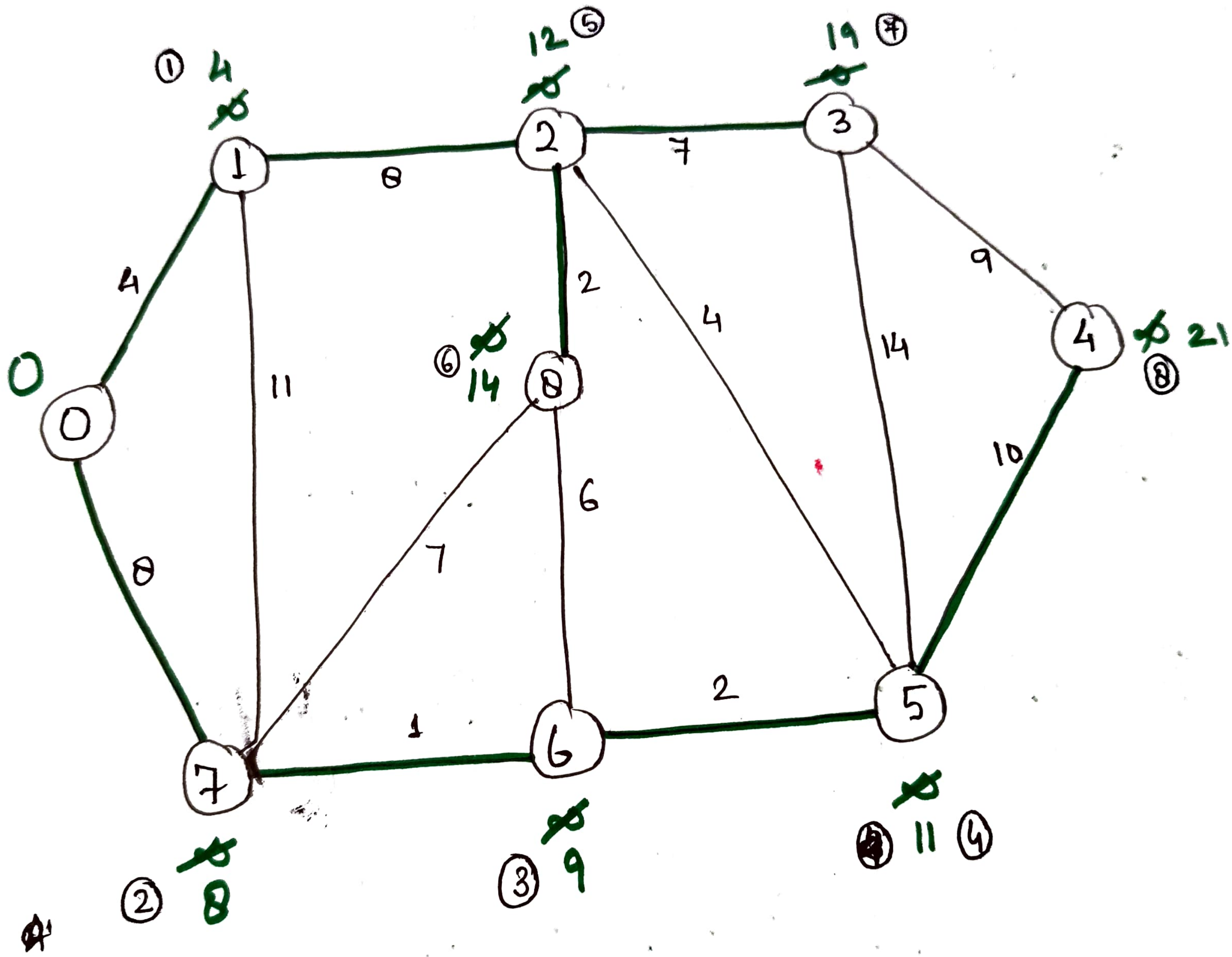
~~i = EXTRACTMIN; mark i;~~

⑤

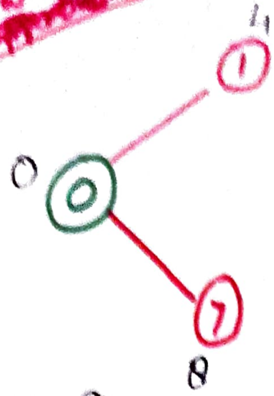
Example →



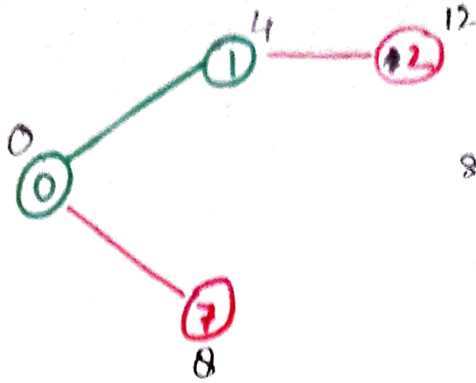
Solution → Dijkstra Algo.



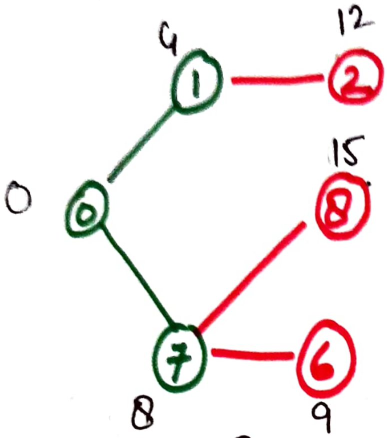
II Approach



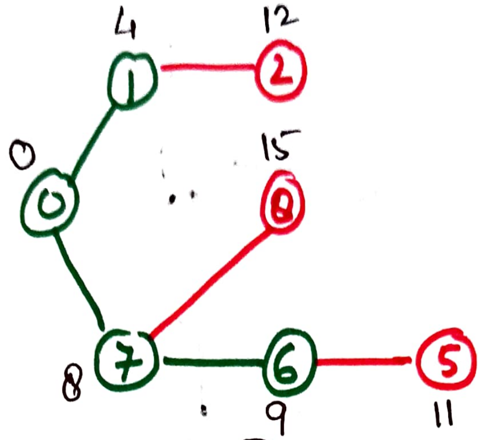
step(1)
minpath.



step(2)



step 3



step 4