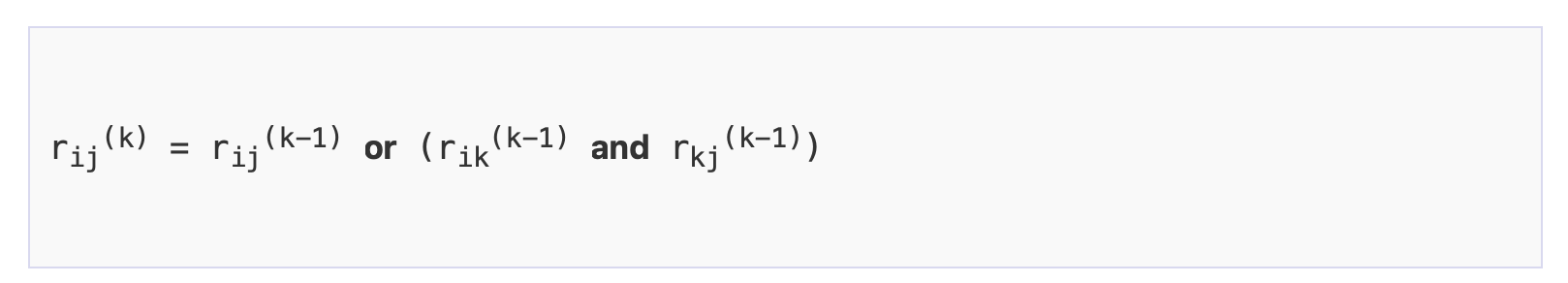
Warshal Algorithm ( transitive Closure)

k=1,2…….n where n is the number of node in a graph



R(0)

1 2 3 4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 0 | 1 | 0 | 0 |
| 2 | 0 | 0 | 1 | 0 |
| 3 | 1 | 0 | 0 | 1 |
| 4 | 0 | 0 | 0 | 0 |

k=1

rij(1)= rij(0) or [r i,1 (0) && r 1,j(0)]

i=1 and j=1

i=1 and j=3

i=1 and j=4

i=2 and j=1

i=3 and j=2

1 2 3 4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 0 | 1 | 0 | 0 |
| 2 | 0 | 0 | 1 | 0 |
| 3 | 1 | \*1 | 0 | 1 |
| 4 | 0 | 0 | 0 | 0 |

k=2

R(2)

rij(2)= rij(1) or [r i,2 (1) && r 2,j(1)]

i=3 j=3

i=1 and j=3 1 2 3 4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 0 | 1 | 1 | 0 |
| 2 | 0 | 0 | 1 | 0 |
| 3 | 1 | 1 | 1 | 1 |
| 4 | 0 | 0 | 0 | 0 |

R(3)

rij(3)= rij(2) or [r i,3 (2) && r 3,j(2)]

1 2 3 4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 1\* | 1 | 1 | 1\* |
| 2 | 1\* | 1\* | 1 | 1\* |
| 3 | 1 | 1 | 1 | 1 |
| 4 | 0 | 0 | 0 | 0 |

R(4)

rij(4)= rij(3) or [r i,4 (3) && r 4,j(3)]

1 2 3 4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 1 |
| 2 | 1 | 1 | 1 | 1 |
| 3 | 1 | 1 | 1 | 1 |
| 4 | 0 | 0 | 0 | 0 |

Djkstra’s Algorithm( Shortest Path Problem)

It is a solution to the single source shortest path problem in graph theory.

It works on both directed and undirected graphs. However all edges must have non-negative weights.

Greedy Algorithm:

A greedy algorithm always makes the choice that looks best at the moment.

2 5 4

4

3

4

2

6

5

We start from node A

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Selection | A | B | C | D | E |
| A | 0 | 2(A) | 5(A) | ∞ | 4(A) |
| A,B | 0 | 2(A) | 5(A) | 8(A,B) | 4(A) |
| A,B,E | 0 | 2(A) | 5(A) | 8(A,B) | 4(A) |
| A,B,E,C | 0 | 2(A) | 5(A) | 7(A,C) | 4(A) |
| A,B,E,C,D | 0 | 2(A) | 5(A) | 7(A,C) | 4(A) |

SELECTED NODE: A,B,E,C,D

2

7 9

2

1 4

10

3

8

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Selection | A | B | C | D | E |
| A | 0 | 10(A) | 3(A) | ∞ | ∞ |
| A,C | 0 | 7(A,C) | 3(A) | 11(A,C) | 5(A,C) |
| A,C,E | 0 | 7(A,C) | 3(A) | 11(A,C) | 5(A,C) |
| A,C,E,B | 0 | 7(A,C) | 3(A) | 9(A,C,B) | 5(A,C) |
| A,C,E,B,D | 0 | 7(A,C) | 3(A) | 9(A,C,B) | 5(A,C) |

3

4

2

2

Minimum Spanning Tree

1.Kruskal Algorithm

2. Prims Algorithm

We prevent cycles , edges having minimum weight connecting all the nodes.

Real life application :

1

4 6 5 3

2 4

Edges in priority queue

|  |  |
| --- | --- |
| (1,2) | 1 |
| (3,5) | 2 |
| (2,4) | 3 |
| (4,5) | 4 |
| (1,3) | 4 |
| (2,5) | 5 |
| (1,5) | 6 |

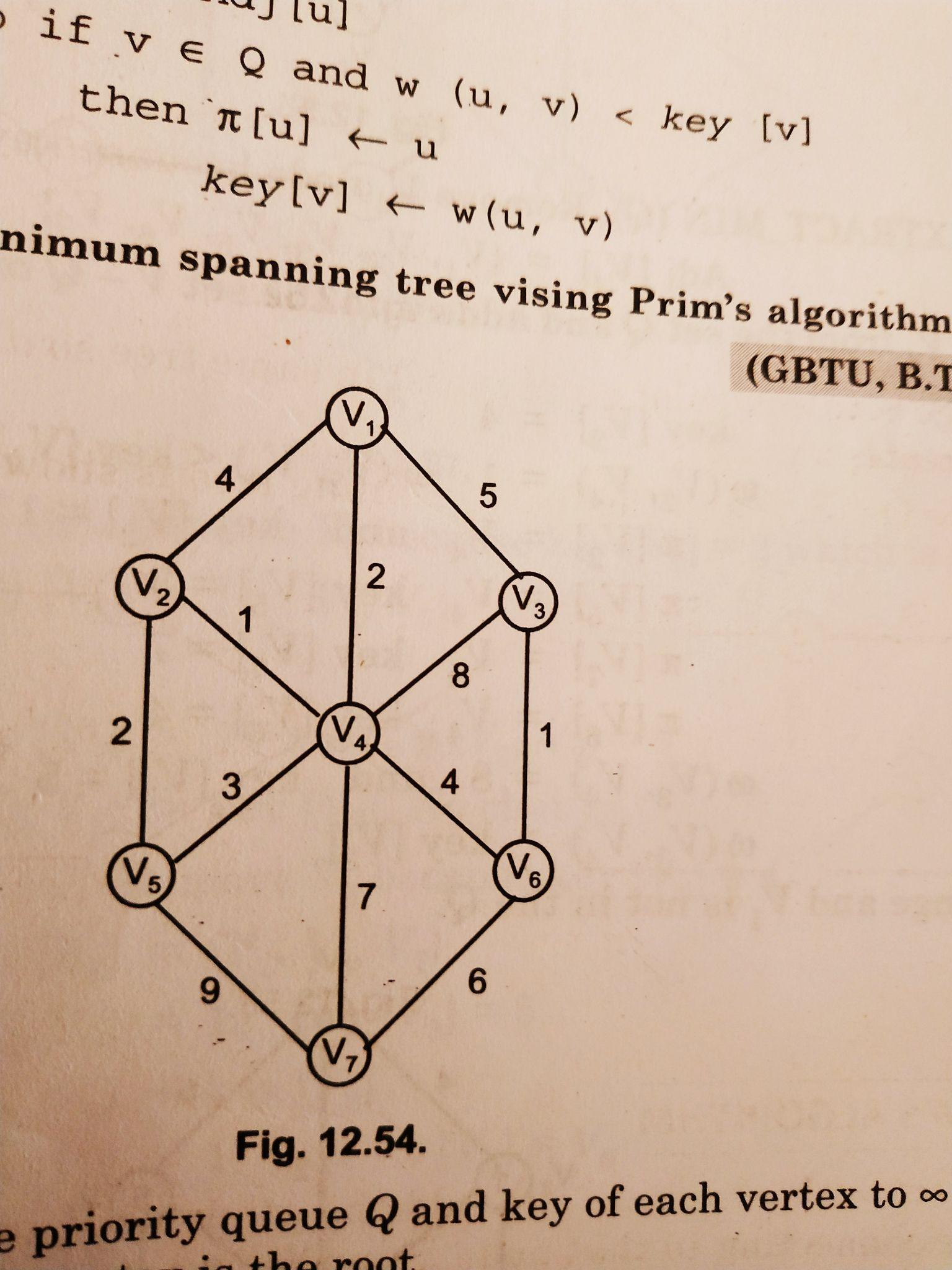
1

2

3

4

Prims Algorithm



Active node List =empty

Active node list: v1,v4,v2,v5,v6,v3,V7

2

1

2

6

1

4

Edges that belongs to MST are

(v1,v4),(v4,v2),(v2,v5),(v4,v6) (v6,v3),(v6,v7)

Weight of MST= 2+1+2+4+1+6=16

Algorithm

1. Select and arbitrary node as a tree node from graph.
2. Node of that graph added to the tree at each step is that node adjacent to a node of the tree by an arc of minimum weight
3. Repeat step 2 until all nodes of the graph are included in tree.