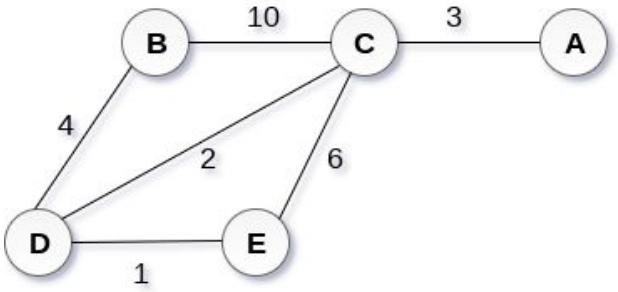
Prim’s Algorithm

Prim's Algorithm is used to find the minimum spanning tree from a graph. Prim's algorithm finds the subset of edges that includes every vertex of the graph such that the sum of the weights of the edges can be minimized.

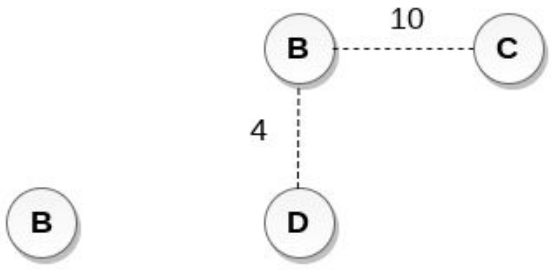
Prim's algorithm starts with the single node and explore all the adjacent nodes with all the connecting edges at every step. The edges with the minimal weights causing no cycles in the graph got selected.

Example

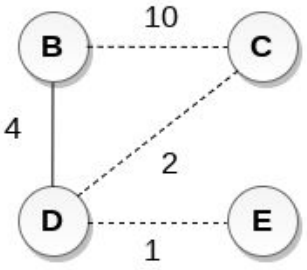
Lets consider start node is B



Solution:

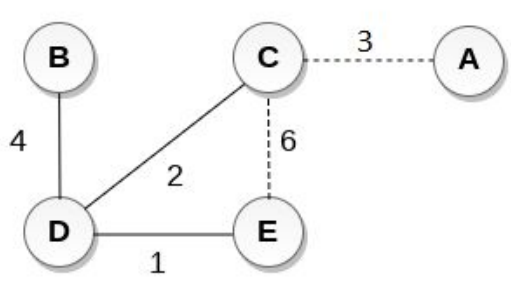


Step 2

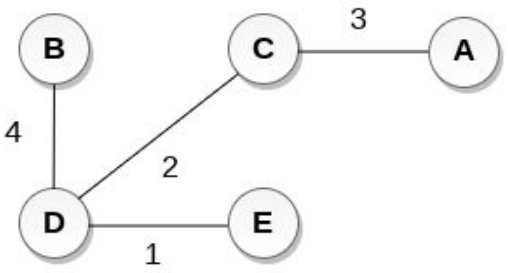


Step 3

Step 1



Step 4



Step 5

Algorithm

Step1:

Create a Min Heap of size V where V is the number of vertices in the given graph. Every node of min heap contains vertex number and key value of the vertex.

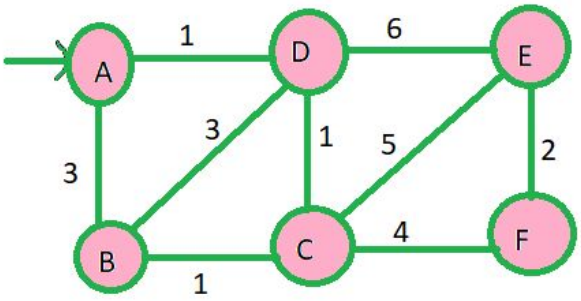
Step2:

Initialize Min Heap with first vertex as root (the key value assigned to first vertex is 0). The key value assigned to all other vertices is INF (infinite).

Step3: While Min Heap is not empty, do following
…..a) Extract the min value node from Min Heap. Let the extracted vertex be u.
…..b) For every adjacent vertex v of u, check if v is in Min Heap (not yet included in MST). If v is in Min Heap and its key value is more than weight of u-v, then update the key value of v as weight of u-v.

Example

Lets consider start vertex is A



Step1:

Start vertex A deleted and adjacent of A will be added

V	E	Heap+ Map	V	Weight
			A	0*
			B	∞
			C	∞
			D	∞
			E	∞
			F	∞

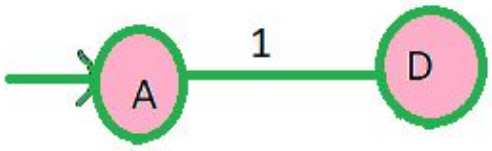
Step2:

Vertex D deleted, {A, D} minimum edge selected, adjacent of D will be added

V	E
B	{A, B}
D	{A, D}+

Heap+ Map

V	Weight
B	3
C	∞
D	1*
E	∞
F	∞



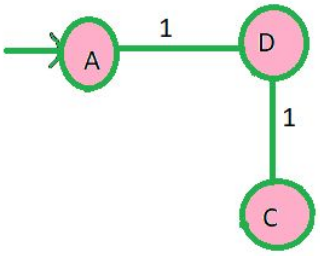
Step3:

Vertex C deleted, {D, C} minimum edge selected, adjacent of C will be added

V	E
B	{A, B}
D	{A, D}
C	{D, C}+
E	{D, E}

Heap+ Map

V	Weight
B	3
C	1*
E	6
F	∞



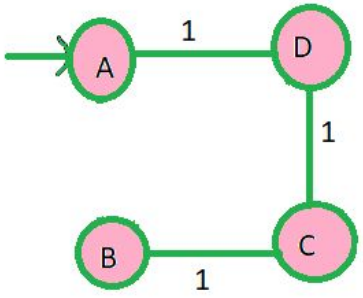
Step4:

Vertex B deleted, {B, C} minimum edge selected, adjacent of B will be added

V	E
B	{B, C}+
D	{A, D}
C	{D, C}
E	{C, E}
F	{C, F}

Heap+ Map

V	Weight
B	1*
E	5
F	4



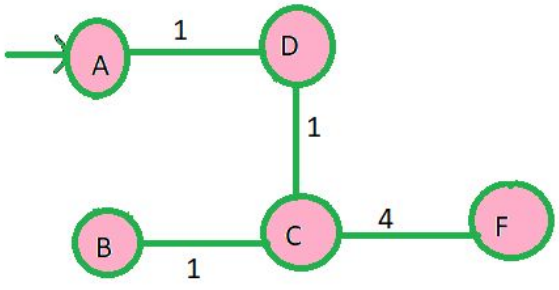
Step5:

Vertex F deleted, {C, F} minimum edge selected, adjacent of F will be added

V	E
B	{B, C}
D	{A, D}
C	{D, C}
E	{C, E}
F	{C, F}+

Heap+ Map

V	Weight
E	5
F	4*



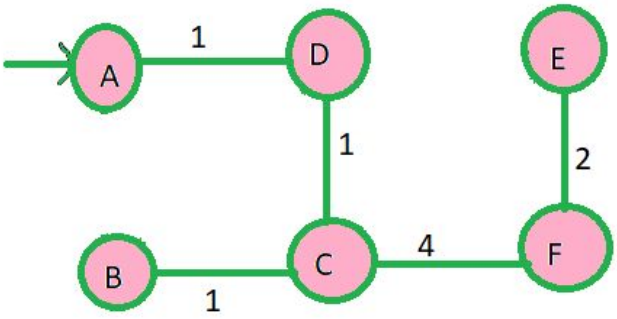
Step6:

Vertex E deleted, {F, E} minimum edge selected, adjacent of E will be added

V	E
B	{B, C}
D	{A, D}
C	{D, C}
E	{F, E}+
F	{C, F}

Heap+ Map

V	Weight
E	5*



Step7: Stop(Heap+ Map empty)

Cost: 1 + 1 + 1 + 4 + 2 = 9

Time Complexity

If adjacency list is used to represent the graph, then using breadth first search, all the vertices can be traversed in O(V + E) time.

- We traverse all the vertices of graph using breadth first search and use a min heap for storing the vertices not yet included in the MST.
- To get the minimum weight edge, we use min heap as a priority queue.
- Min heap operations like extracting minimum element and decreasing key value takes O(logV) time.

So, overall time complexity

= O(E + V) x O(logV)

= O((E + V)logV)

= O(ElogV)

This time complexity can be improved and reduced to O(E + VlogV) using Fibonacci heap.