## Prim Algorithm

*Principle:*

*Just like the Kruskal Algorithm, Prim Algorithm is an special solution of the Minimum Spanning Tree Problem.* The basic principle of Prim Algorithm is just like the Shortest Route Algorithm as Dijkstra Algorithm. All edges in the collection A would consist one tree.

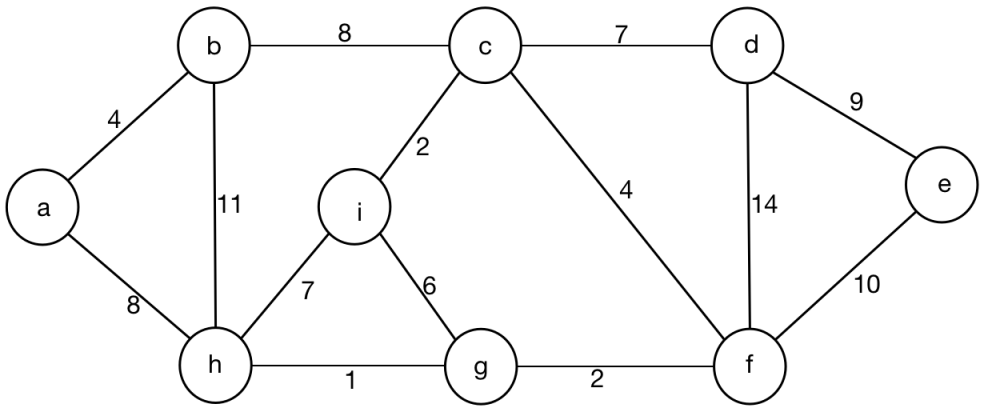
*Step:*

The tree starts from one random root node r, and grow to cover all nodes in collection V. Each step in the algorithm would choose one node outside the current collection V. The edge connects the node in the collection V with another node outside the collection V with the Minimum Total Weight. The added edge must be safe to the current collection V. Till the end of Algorithm, all edges in the Collection V would form the Minimum Spanning Tree.

*Supplement:*

Prim Algorithm belongs to the Greedy Algorithm, since each added edge must be one edge which constitutes the smallest weight.

*Example:*



*Initialization:*

A = Empty.

V - A = { a, b, c, d, e, f, g, h, i }.

Edge = Empty.

The information of all edges are collected in the array as below:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Node a | Node b | Node c | Node d | Node e | Node f | Node g | Node h | Node i |
| a, b, 4 | b, a, 4 | c, b, 8 | d, c, 7 | e, d, 9 | f, e, 10 | g, f, 2 | h, g, 1 | i, c, 2 |
| a, h, 8 | b, c, 8 | c, d, 7 | d, e, 9 | e, f, 10 | f, d, 14 | g, i, 6 | h, i, 7 | i, g, 6 |
|  | b, h, 11 | c, f, 4 | d, f, 14 |  | f, c, 4 | g, h, 1 | h, b, 11 | i, h, 7 |
|  |  | c, i, 2 |  |  | f, g, 2 |  | h, a, 8 |  |

Sort all edges according to the value of weight, according to the descending sequence.

|  |
| --- |
| ( h, g, 1 ) |
| ( g, f, 2 ) |
| ( i, c, 2 ) |
| ( a, b, 4 ) |
| ( c, f, 4 ) |
| ( i, g, 6 ) |
| ( i, h, 7 ) |
| ( c, d, 7 ) |
| ( a, h, 8 ) |
| ( b, c, 8 ) |
| ( d, e, 9 ) |
| ( e, f, 10 ) |
| ( b, h, 11 ) |
| ( d, f, 14 ) |

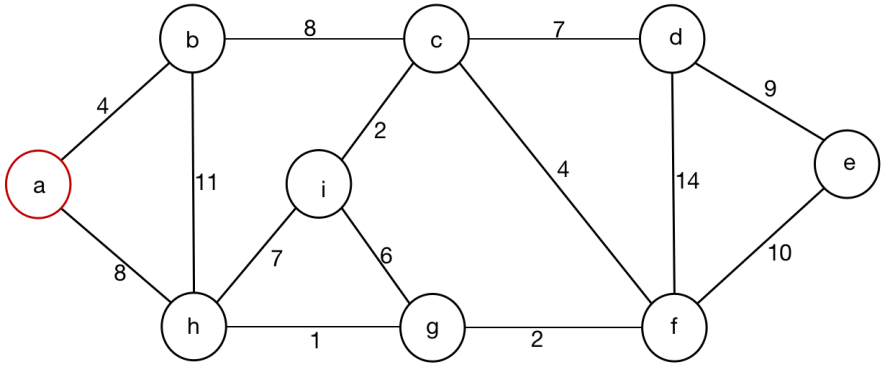
*First Step:*

Add node a into the collection A.

The information of collection A and collection of V - A are as below:

1. A = { a }
2. V - A = { b, c, d, e, f, g, h, i }.
3. Edge = Empty.

The Minimum Spanning Tree is just as below:



*Second Step:*

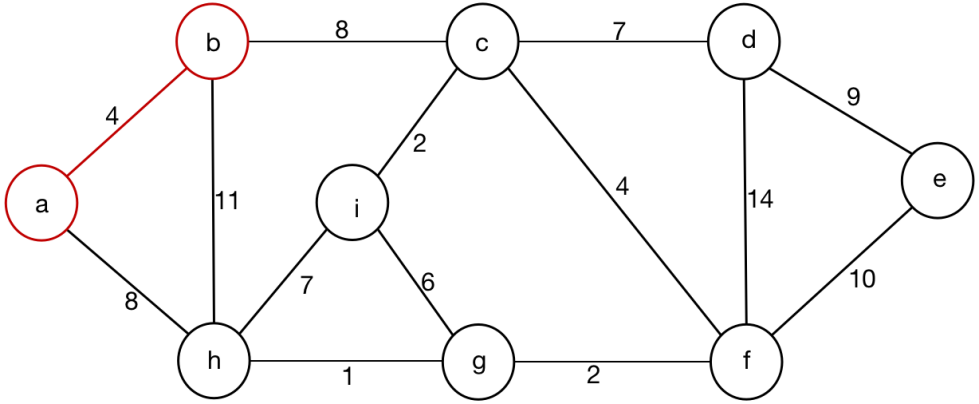
Select one node from the collection A, and select another node from the collection V - A, which makes the Minimum Spanning Tree with the Minimum Weight.

Choose the node a from the collection A, and choose another node b from V - A, with the weight 4 among all left edges, which makes the Minimum Spanning Tree with the Minimum Total Weight.

The information of collection A, collection V - A, and collection Edge are as below:

1. A = { a, b }.
2. V - A = { c, d, e, f, g, h, i }.
3. Edge = { ( a, b, 4 ) }.

The Minimum Spanning Tree is just as below:



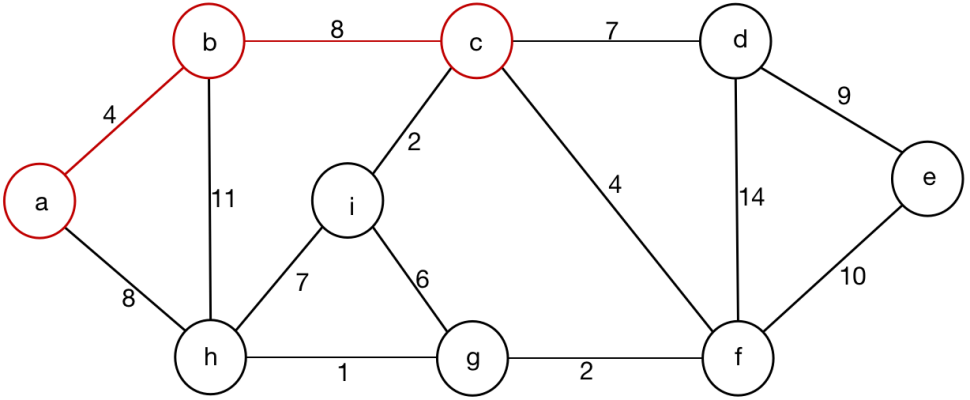
*Third Step:*

Choose the node b from the collection A, and choose another node c from V - A, with the weight 8 among all left edges, which makes the Minimum Spanning Tree with the Minimum Total Weight.

The information of collection A, collection V - A, and the collection Edge are as below:

1. A = { a, b, c }.
2. V - A = { d, e, f, g, h, i }.
3. Edge = { ( a, b, 4 ), ( b, c, 8 ) }.

The Minimum Spanning Tree is just as below:



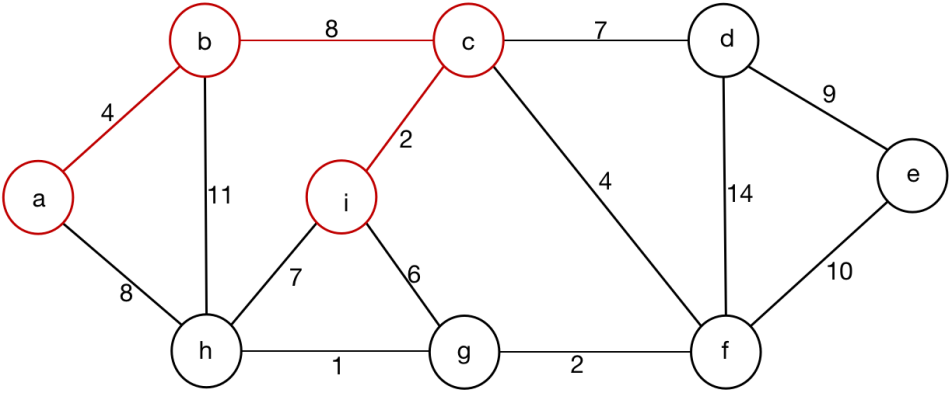
*Forth Step:*

Choose the node c from the collection A, and choose another node i from V - A, with the weight 2 among all left edges, which makes the Minimum Spanning Tree with the Minimum Total Weight.

The information of the collection A, collection V - A, and the collection Edge are as below:

1. A = { a, b, c, i }.
2. V - A = { d, e, f, g, h }.
3. Edge = { (a, b, 4), (b, c, 8), (c, i, 2) }.

The Minimum Spanning Tree is just as below:



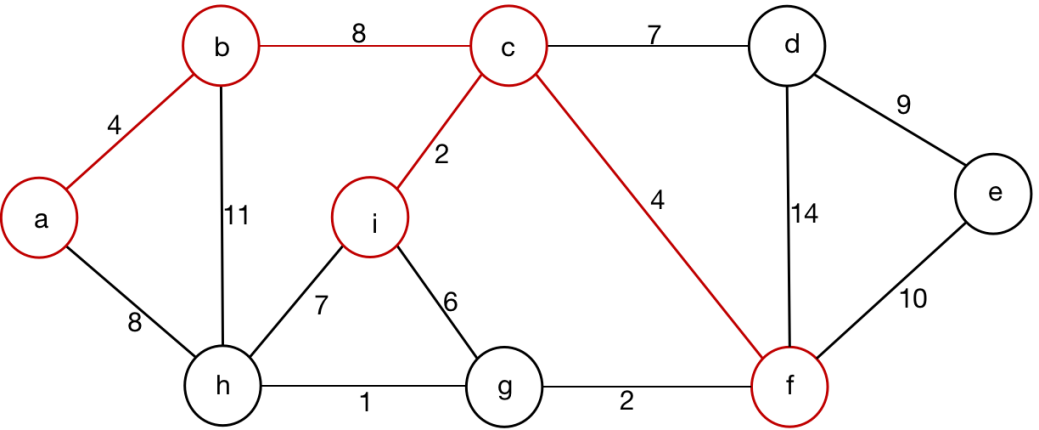
*Fifth Step:*

Choose the node c from the collection A, and choose another node f from V - A, with the weight 4 among all left edges, which makes the Minimum Spanning Tree with the Minimum Total Weight.

The information of the collection A, collection V - A, and the collection Edge are as below:

1. A = { a, b, c, i, f }.
2. V - A = { d, e, g, h }.
3. Edge = { (a, b, 4), (b, c, 8), (c, i, 2), (c, f, 4) }.

The Minimum Spanning Tree is just as below:



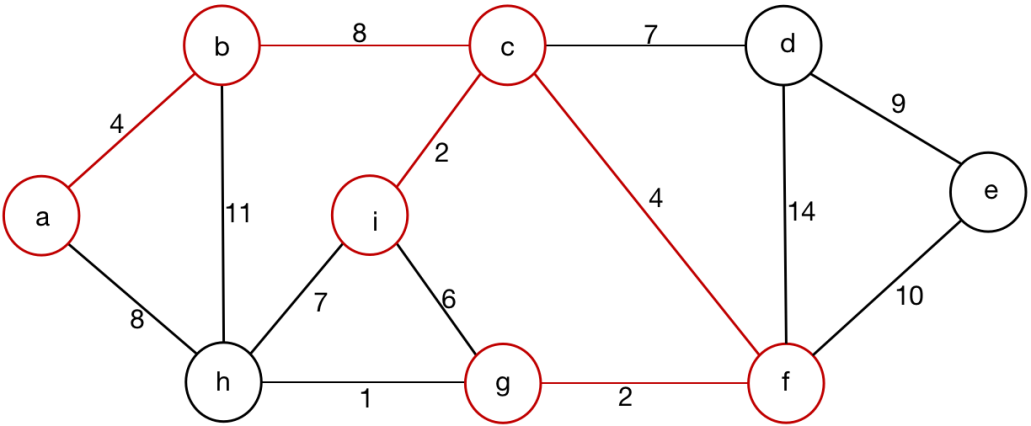
*Sixth Step:*

Choose the node f from the collection A, and choose another node g from the collection V - A, with the weight 2 among all left edges, which makes the Minimum Spanning Tree with the Minimum Total Weight.

The information of the collection A, collection V - A, and the collection Edge are as below:

1. A = { a, b, c, i, f, g }.
2. V - A = { d, e, h }.
3. Edge = { (a, b, 4), (b, c, 8), (c, i, 2), (c, f, 4), (f, g, 2) }.

The Minimum Spanning Tree is just as below:



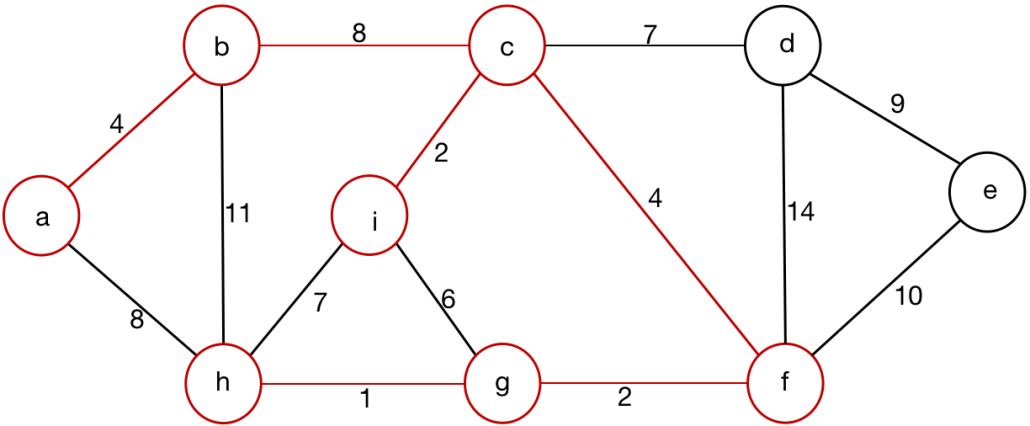
*Seventh Step:*

Choose the node g from the collection A, and choose another node h from the collection V - A, with the weight 1 among all left edges, which makes the Minimum Spanning Tree with the Minimum Total Weight.

The information of the collection A, collection V - A, and the collection Edge are as below:

1. A = { a, b, c, i, f, g, h }.
2. V - A = { d, e }.
3. Edge = { (a, b, 4), (b, c, 8), (c, i, 2), (c, f, 4), (f, g, 2), (g, h, 1) }.

The Minimum Spanning Tree is just as below:



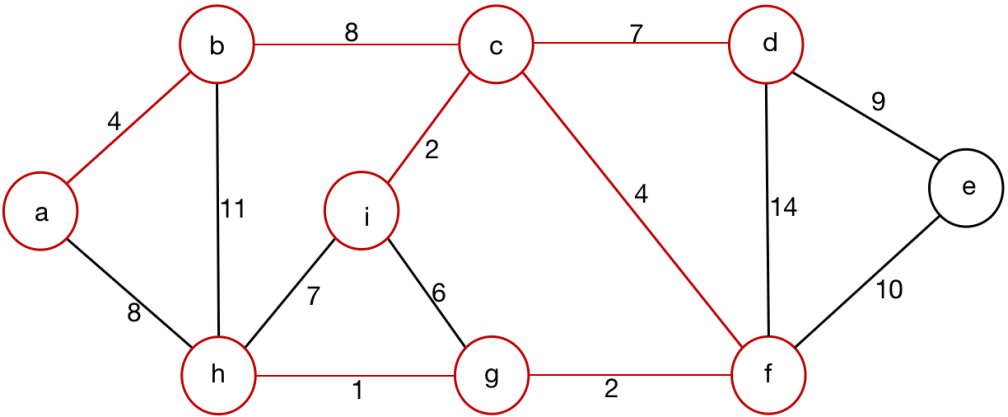
*Eighth Step:*

Choose the node c from the collection A, and choose another node d from the collection V - A, with the weight 7 among all left edges, which makes the Minimum Spanning Tree with the Minimum Total Weight.

The information of the collection A, collection V - A, and the collection Edge are as below:

1. A = { a, b, c, i, f, g, h, d }.
2. V - A = { e }.
3. Edge = { (a, b, 4), (b, c, 8), (c, i, 2), (c, f, 4), (f, g, 2), (g, h, 1), (c, d, 7) }.

The Minimum Spanning Tree is just as below:

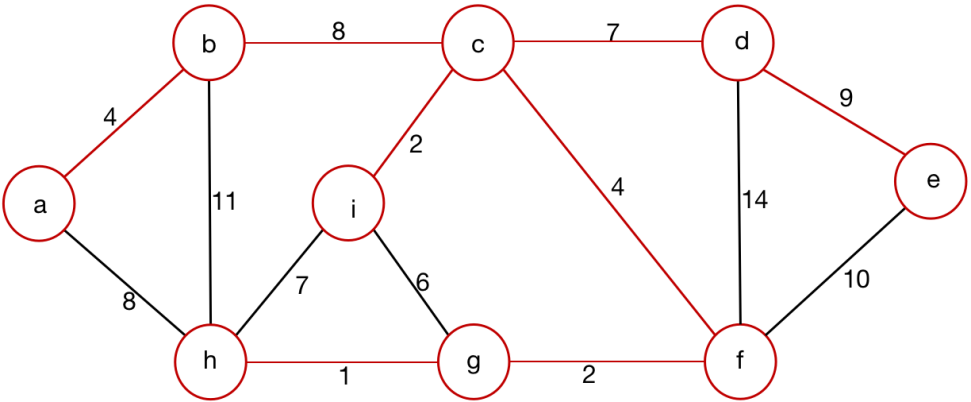


*Ninth Step:*

Choose the node d among the collection A, and choose another node e from the collection V - A, with the weight 9 among all left edges, which makes the Minimum Spanning Tree with the Minimum Total Weight.

1. A = { a, b, c, i, f, g, h, d, e }.
2. V - A = Empty.
3. Edge = { (a, b, 4), (b, c, 8), (c, i, 2), (c, f, 4), (f, g, 2), (g, h, 1), (c, d, 7), (d, e, 9) }.

The Minimum Spanning Tree is just as below:



*The Final Result would be:*

1. *A = { a, b, c, i, f, g, h, d, e }.*
2. *V - A = Empty.*
3. *Edge = { (a, b, 4), (b, c, 8), (c, i, 2), (c, f, 4), (f, g, 2), (g, h, 1), (c, d, 7), (d, e, 9) }.*

*Pseudo Code:*

*Analysis:*

*Data Structure of Graph G:*

* *Graph G includes node and vertex information, which can be represented by using G.V and G.E.*
* *Graph G is represented by the Adjacent List.*

*Detail information among Adjacent List:*

* *In each Adjacent List, there also keep other information to finish the following function, including the minimum weight value which stands for minimum weight among all adjacent edges.*
* *Except for the current node, the pioneer node is also needed to be recorded the current edge with minimum weight value.*

*Node Structure:*

*Node node {  
 int value; // The value of the current node;*

*Node \* relative\_node; // The relative node of the current node.*

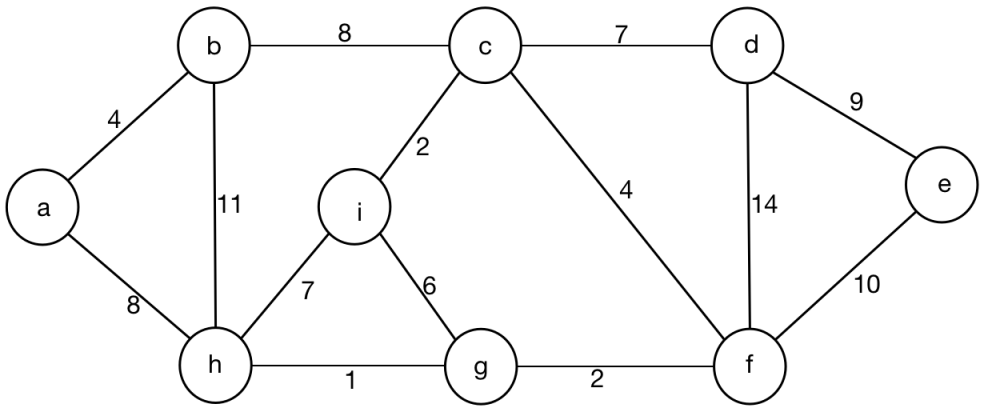
*int weight; // The minimum weight of the current node.*

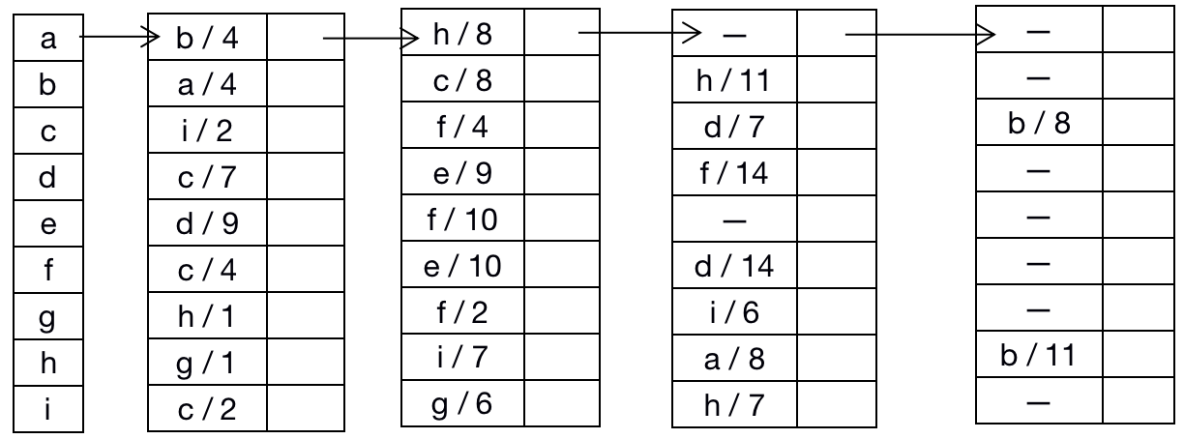
*Node \* adjacent; // The adjacent node of current with minimum weight.*

*}*

*Image:*

*Graph Image and Adjacent List are as below:*



**

*Code:*

*Node node {  
 int value; // The value of the current node;*

*Node \* relative\_node; // The relative node of the current node.*

*int weight; // The minimum weight of current node.*

*Node \* adjacent\_node; // The adjacent node of current with minimum weight.*

*}*

*MST - PRIM ( G, root, total\_weight )*

*// Graph G is used to record all nodes and edges of the original graph.*

*Graph G;*

*// Pointer node root is used to present root node of Graph G.*

*Node \* root;*

*//Variable total\_weight is used to present total weight of Minimum Spanning Tree.*

*int total\_weight; {*

*// Initialize all nodes information excluding all values and relative\_node;*

*for ( int i = 0; i < G.size(); i ++ ) {*

*G[ i ]->weight = 0;*

*G[ i ]->relative\_node = NULL;*

*G[ i ]->adjacent\_node = NULL;*

*}*

*Store the first node G[ 0 ] into the collection V;*

*G[ 0 ]->weight = 0;*

*G[ 0 ]->adjacent\_node = NULL;*

*Set the collection G - V as the next node select from;*

*Node \* adjacent;*

*Node \* node;*

*int weight;*

*while ( G - V ! = Empty ) {*

*// Select the edge consists of (node, adjacent) with the minimum weight;*

*(node, adjacent, weight) = Minimum\_Select (V, G - V);*

*node->weight = weight;*

*node->adjacent\_node = adjacent;*

*Store the node into the collection V;*

*Set the collection G - V as the next node select from;*

*Continue and go to the next round;*

*}*

*return collection V as the final result;*

*}*

*Instruction:*

Procedure *Minimum\_Select (V, G - V)* is used to select one edge with the minimum weight. During the procedure, one node is in the collection V, while the other node is in the collection G - V. Under such procedure, this edge can be ensured as one safety edge.

*Minimum\_Select (V, G) {*

*For each node in collection V, check all edges in the collection G:*

*Select the minimum weight for each node;*

*Check whether another node of edge is in collection G - V;*

*( Exclude the edge that another node is also in collection V; )*

*Find the edge with minimum weight, node and adjacent;*

*return (node, adjacent, weight);*

*}*