## Minimum Spanning Tree

*Usage Scenario:*

During Electronic Circuit Design, stitches with multiply components need to be connected together. When we want to connect with n stitches, n – 1 lines can be used to connect each stitch pair. Apparently, the total length of all lines should be as long as possible.

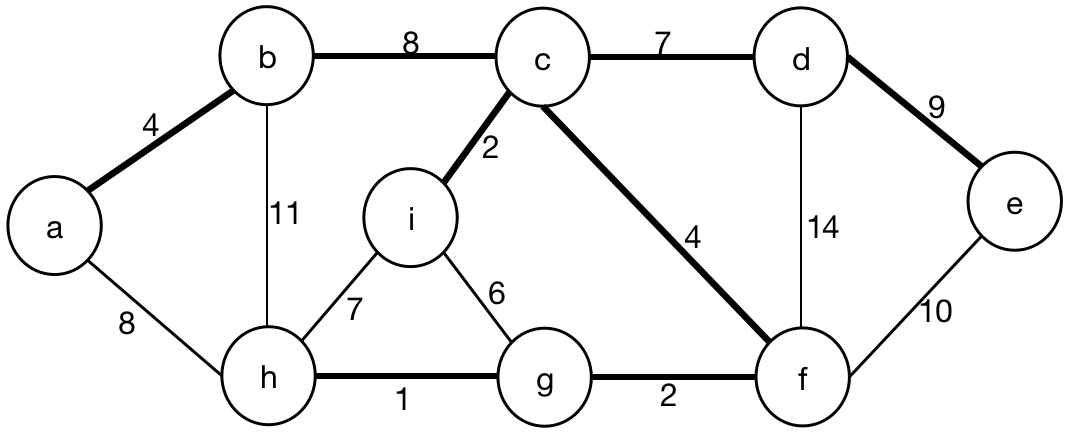
*Description:*

Using one *Connected Undirected Graph G = (V, E)* to represent, and V is the collection of all stitches, while *E* is the collection of all possible connection between *the stitches u and v*. For each *edge (u, v)* which belongs to E, we assign the *weight w(u, v)* as the cost of connection between the stitch u and v.

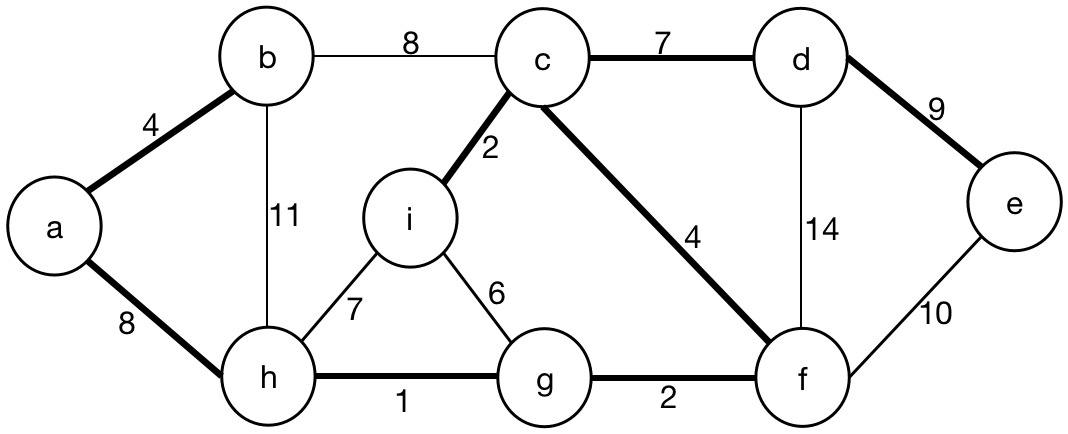
We hope to find an *Acyclic Sub – Graph Collection T* belongs to E, which can be used to connect all stitches, and has the least cost, which means that *w(T) = Total-Sum w(u, v) (u, v)* belongs to E.

Since T is the Acyclic Graph which connects all nodes, therefore, T can be treated as one *Generated Tree*. Of course, we can call T the *Minimum Spanning Tree*. Such problem can be called as the *Minimum Spanning Tree Problem*.

*For Example:*

**

One of the Minimum Spanning Tree in the Graph above is G = { V, E }, V = { a, b, c, d, e, i, f, g, h }, E = { (a, b), (b, c), (c, d), (d, e), (c, f), (c, i), (f, g), (g, h) }. The total cost in the Graph above equals to ~~4~~ + ~~8~~ + 7 + ~~9~~ + 4 + ~~2~~ + 2 + ~~1~~ = ~~4~~ + ~~10~~ + ~~10~~ + ~~13~~ = 20 + 17 = 37.



However, another Minimum Spanning Tree in the Graph above is G = { V, E }, V = { a, b, c, d, e, i, f, g, h }, E = { (a, b), (a, h), (c, d), (d, e), (c, f), (c, i), (f, g), (g, h) }. The total cost in the Graph above equals to 4 + 8 + 7 + 9 + 4 + 2 + 2 + 1 = 4 + 10 + 10 + 13 = 20 + 17 = 37.

*Introduction:*

In this Chapter, we would like to introduce two algorithms which can be used to solve *Minimum Spanning Tree Problem*, which are *Kruskal Algorithm* and *Prim Algorithm*. These two algorithms are all *Greedy Algorithm*.