Chapter-6

Solving Spanning Tree Problems

In this chapter, two new algorithms namely, spread search algorithm and neighbour search algorithm are proposed for finding a maximum spanning tree of a given weighted connected simple graph and fuzzy graph respectively. The proposed algorithms differ from the existing algorithms with time complexity of $O((m-k)^2)$ where $m$ is the number of edges and $k$ is the number of pendant edges in the given graph. Numerical examples are presented for illustrating the solution procedure of the algorithms.

6.1 INTRODUCTION

Graph theory (Bondy and Murty, 1976; Vasudev, 2006) is simply to say that it is a study of the graph that began with Leonhard Euler in his study of the Bridges of Konigsberg problem. Network which is a connected graph is one more popular area in graph theory. Many researchers have contributed more related to connectivity problems in the sense of theoretical as well as in computational. Minimum spanning tree problem in a connected graph is one of most important in combinatorial optimization and has important applications in transportation, communications, distribution systems, etc. Many authors have studied the minimum spanning tree problem and presented numerous efficient algorithms (Dijkstra, 1959; Kruskal, 1956; Prim, 1957; Gabow et al. 1986; Bondy and Murty, 1976; Christofides, 1975). Among these, the two basic algorithms namely, Kruskal’s algorithm and Prim’s algorithm are greedy algorithms which are more popular for solving minimum (maximum) spanning tree in a connected graph. The modified Prim’s algorithm used maximum spanning tree to solve maximum capacity root problem (Prim, 1957). Fuzzy graph theory (Mordeson and Nair, 2000) is a combined theory of graph theory (Bondy and Murty, 1976) and fuzzy set theory (Zadeh, 1965). In 1975, Rosenfeld (1975) and Yeh and Bang (1975) introduced fuzzy graphs independently. Rosenfeld (1975) established several basic fuzzy graph-theoretic concepts like trees, bridges, cycles, paths and connectedness and developed some of their
properties. An algorithm was presented to find the connectivity of a pair of nodes in a fuzzy graph by Bhattacharya and Suraweera (1991). Some connectivity concepts regarding fuzzy bridges and fuzzy cut nodes were established by Mathew and Sunitha (2010, 2009) and Bhattacharya (1987). Sunitha and Vijayakumar (1999) studied the some properties of fuzzy bridges and fuzzy cut nodes using these concepts they characterized the fuzzy trees. The strong arcs of a fuzzy tree was introduced and studied by Bhutani and Rosenfeld (2003). The idea of strong arc in maximum spanning trees and its applications in neural networks and cluster analysis were studied by Sameena and Sunitha (2006,2008, 2012). Bhattacharya and Suraweera (1991) proposed an algorithm for finding a maximum spanning tree of a connected fuzzy graph using a depth first search. In fuzzy graphs, the existing minimum spanning tree algorithms in crisp graph are used to construct a maximum spanning tree of a fuzzy graph.

6.2 A WEIGHTED CONNECTED SIMPLE GRAPH

A graph G consists of a pair \((V, E)\), where \(V\) is a non empty finite set whose elements are called vertices and \(E\) is its edges are pairs of distinct vertices of \(V\) are called edges of the graph \(G\).

An edge in a graph \(G\) is said to be a loop if it connects a vertex to itself. A set of edges in a graph \(G\) is said to be multiple edges if they have the same start and end vertices. A simple graph is a graph which has neither loops nor multiple edges. A graph \(G\) in which every edge is assigned a real number is called a weighted graph. A graph \(H=(V_1, X_1)\) is called a subgraph of \(G=(V, X)\) if \(V_1 \subseteq V\) and \(X_1 \subseteq X\).

A connected acyclic graph is called a tree. A spanning tree of a graph \(G\) is a tree which has all the vertices of \(G\).

A weighted graph is a graph \(G\) in which each edge \(e\) has been assigned a non-negative number \(w(e)\), called the weight of \(e\). The weight of a spanning tree, \(T\) is the sum of the weights of the edges in the tree \(T\). A maximum (minimum) spanning tree of \(G\) is a spanning tree of \(G\) with maximum (minimum) weight.
Now, define the following terms in a weighted graph $G = (V, E)$.

Let $v_1$ and $v_2$ be two nodes in $G$. The node $v_1$ is said to be a neighbour of the node $v_2$ if $w(v_1, v_2) \neq 0$.

Let $v_1$ and $v_2$ be two nodes in $G$. The node $v_1$ is said to be a strong neighbour of the node $v_2$ if $w(v_1, v_2)$ is the maximum of $\{ w(v_1, v_2) : v_2 \in V \text{ is a neighbour of } v_1 \}$.

Let $v_1$ and $v_2$ be two nodes in $G$. The node $v_1$ is said to be a weak neighbour of the node $v_2$ if $w(v_1, v_2)$ is the minimum of $\{ w(v_1, v_2) : v_2 \in V \text{ is a neighbour of } v_1 \}$.

6.3 SPREAD SEARCH ALGORITHM

We suggest the following new algorithm namely, spread search algorithm for finding a maximum spanning tree of a given weighted connected simple graph.

The proposed algorithm proceeds as follows:

ALGORITHM:

Let $G = (V, E)$ be a connected weighted simple graph with $n$ vertices and $m$ edges.

STEP 1: Collect all the pendent edges of $G$ and form a set. Let it be $S$.

STEP 2: If $|S| = n - 1$, stop the computation and $T_0 = (V, S)$ is a spanning tree. If not, move to the Step 3.

STEP 3: Find an edge $e_1 = (u_1, u_2)$ in $G_1 = G - S$ such that $w(e_1) = w(u_1, u_2)$ is maximum. If more than one occur, select any one edge.

STEP 4: Construct $T_1 = T_0 + e_1 = (V, S_1)$ where $S_1 = S \cup e_1$.

If $|S_1| = n - 1$, stop the computation and $T_1$ is a spanning tree.

If not, move to the Step 5.
STEP 5: Find an edge $e_2$ whose one end vertex is an end vertex of the edge $e_1$ in the graph $G_2 = G_1 - e_1$ such that $w(e_2)$ is the maximum of \{ $w(u, u_r), \mu_r$ is a strong neighbour of a non-isolated end vertex $u$ of the edge $e_1$ in the graph $G_2$ \}.

STEP 6: Construct $T_2 = T_1 + e_2 = (V, S_2)$ where $S_2 = S_1 \cup e_2$.

If $|S_2| = n - 1$, stop the computation and $T_2$ is a spanning tree.

If not, move to the Step 7.

STEP 7: Find an edge $e_3$ whose one end vertex is an end vertex of the edge $e_1$ or $e_2$ in the graph $G_3 = G_2 - e_2$ such that $w(e_3)$ is the maximum of \{ $w(u, u_r), \mu_r$ is a strong neighbour of a non-isolated end vertex $u$ of the edge $e_1$ or $e_2$ in the graph $G_3$ \} with $T_2 + e_3$ not containing a cycle. If $e_3$ forms a cycle delete edge from $G_3$ continue the step 7.

STEP 8: Construct $T_3 = T_2 + e_3 = (V, S_3)$ where $S_3 = S_2 \cup e_3$.

If $|S_3| = n - 1$, stop the computation and $T_3$ is a spanning tree.

If not, move to the Step 7.


Now, the following theorem is established to show the maximality of the spanning tree obtained by the proposed method.

**Theorem 6.3.1** In a connected weighted simple graph $G$, the spread search algorithm provides a maximum weighted spanning tree of $G$.

**Proof**: Let $T_0$ be a graph obtained by the spread search algorithm.

According to the proposed algorithm, $T_0$ is a spanning tree of $G$. 

80
CLAIM: To prove that $T_0$ is a maximum spanning tree of $G$.

That is, $W(T_0) \geq W(T)$, for all spanning tree $T$ of $G$.

Let $T$ be a spanning tree of $G$

Assume that $W(T_0) < W(T)$.

This implies that there exist an edge $e = (u, v)$ in $T$, but $e \notin T_0$ such that the weight of the edge $e = (u, v)$ is greater than the weight of the path connecting the vertices $u$ and $v$ in $T_0$.

Now, since $e \notin T_0$, $u$ is not a strong neighbour of $v$ in $G$ and there exist strong neighbours for $u$ and $v$ in $G$, $u_1$ and $v_1$ respectively such that $(u, u_1)$ and $(v, v_1)$ are in $T_0$.

Since the strong neighbour for $u$ in $G$ is $u_1$ and the strong neighbour for $v$ in $G$ is $v_1$, we have $W(u, u_1) > W(u, v)$ and $W(v, v_1) > W(u, v)$.

Now, $W(u, u_1) > W(u, v)$ and $W(v, v_1) > W(u, v)$, the weight of the path connecting $u$ and $v$ containing $(u, u_1)$ or $(v, v_1)$ is greater than the weight of the edge $(u, v)$ which is contradicts to the hypothesis $W(T_0) < W(T)$. Therefore, $W(T_0) \geq W(T)$, for all spanning tree $T$ of $G$.

Thus, $T_0$ is a maximum spanning tree of $G$.

This completes the theorem.

Remark 6.3.2 The proposed algorithm can be also, used to determining a minimum spanning tree of a weighted connected simple graph by considering in STEP 3. Find minimum weight in the given graph and the weak neighbour instead of strong neighbour in the algorithm.
Remark 6.3.3 If the graph $G$ has $k$-pendent edges, then the spread search algorithm has $m-k+1$ iterations takes to complete because one edge is added at each iteration. Therefore, the time complexity of the proposed algorithm is $O((m-k)^2)$, but the time complexity of the Prim’s algorithm is $O(n^2)$ where $n$ is the number of vertices and the time complexity of the Kruskal’s algorithm is $O(m \log n)$ where $n$ is the number of vertices and $m$ is the number of edges.

Now, we present the following numerical examples for understanding the solution procedure of the spread search algorithm.

Example 6.3.1 Consider the following weighted connected simple graph.

![Figure 6.3.1 Weighted graph](image)

82
CLAIM: To determine a maximum spanning tree of the given graph.
Now, the number of edges = 26 and the number of vertices = 12.
Now, since by the Step 1. and Step 2., we have
\[ S = \{(v_1, v_7), (v_2, v_{12}), (v_3, v_{11}), (v_4, v_9), (v_5, v_8), (v_6, v_{10})\} \quad T_0 = (V, S). \]
Now, since \( w(v_3, v_4) \) is maximum in the reduced graph
\[ G - \{(v_1, v_7), (v_2, v_{12}), (v_3, v_{11}), (v_4, v_9), (v_5, v_8), (v_6, v_{10})\} = (v_3, v_4) \]
and by Step 3. and Step 4., we have \( T_1 = T_0 + (v_3, v_4). \)
Now, by the Step 5. and the Step 6., we have \( T_2 = T_1 + (v_3, v_5). \)
Now, using the Step 9., we have \( T_3 = T_2 + (v_3, v_1). \)
Now, using the Step 9., we have \( T_4 = T_3 + (v_1, v_2). \)
Now, using the Step 9., we have \( T_5 = T_4 + (v_3, v_6). \)
and we stop the computation since the number of edges in \( T_5 \) is 11.
By Step 6., \( T_5 \) is a maximum spanning tree of the given graph and its total weight is 53.

Figure 6.3.2 Weighted maximum spanning tree
Example 6.3.2 Consider the following weighted connected simple graph.

CLAIM: To determine a minimum spanning tree of the given graph

Now, the number of edges = 18 and the number of vertices = 13.

Now, since by the Step 1. and Step 2., we have

\[ S = \{(v_1, v_8), (v_2, v_{13}), (v_3, v_4), (v_4, v_{11}), (v_5, v_7), (v_6, v_9)\} \]

\[ T_e = (V, S). \]

![Figure 6.3.3 Weighted Graph](image)

Now, a minimum spanning tree for the given graph based on the Remark 6.3.2 is determined.

Now, since \( w(v_8, v_9) \) is minimum in the reduced graph

\[ G - \{(v_1, v_8), (v_2, v_{13}), (v_3, v_4), (v_4, v_{11}), (v_5, v_7), (v_6, v_9)\} = (v_8, v_9) \]

and by Step 3. and Step 4., we have \( T_1 = T_0 + (v_8, v_9). \)

Now, by the Step 5. and the Step 6., we have \( T_2 = T_1 + (v_8, v_7). \)

Now, using the Step 9., we have \( T_3 = T_2 + (v_7, v_{10}). \)

Now, using the Step 9., we have \( T_4 = T_3 + (v_{10}, v_{11}). \)
Now, using the Step 9., we have \( T_5 = T_4 + (v_{11}, v_{12}) \).

Now, using the Step 9., we have \( T_6 = T_5 + (v_7, v_{13}) \).

and we stop the computation since the number of edges in \( T_6 \) is 12.

By Step 6., \( T_6 \) is a minimum spanning tree of the given graph and its total weight or membership value is 56.

Figure 6.3.4 Weighted minimum spanning tree

6.4 FUZZY GRAPHS

Let \( G \) be a graph with node set \( V \) and arc set \( E \subseteq V \times V \). A fuzzy graph is a pair \( G : (\sigma, \mu) \) where \( \sigma : V \rightarrow [0,1] \) and \( \mu : V \times V \rightarrow [0,1] \) such that \( \mu(u, v) \leq \sigma(u) \land \sigma(v) \), for all \( u, v \in V \) where \( V \) is a non-empty finite set called a set of vertices of the graph \( G \). If \( \mu(x, y) > 0 \), we say that \( x \) and \( y \) are neighbors in \( G : (\sigma, \mu) \).

A vertex in a fuzzy graph having only one neighbour is called a pendent vertex. An edge in a fuzzy graph incident with a pendent vertex is called a pendent edge.

The underlying crisp graph of a fuzzy graph \( G : (\sigma, \mu) \) is denoted by \( G^* = (\sigma^*, \mu^*) \), where \( \sigma^* = \{ u \in V / \sigma(u) > 0 \} \) and \( \mu^* = \{ (u, v) \in V \times V / \mu(u, v) > 0 \} \).
A path $P$ in a fuzzy $G : (\sigma, \mu)$ graph is a sequence of distinct vertices $u_0, u_1, u_2, \ldots, u_n$ such that $\mu(u_{i-1}, u_i) > 0$, $i = 1, 2, 3 \ldots n$. The weight of the path $P$ in a fuzzy graph $G$, $w(P)$ is defined by $w(P) = \sum_{i=1}^{n} \mu(u_{i-1}, u_i)$. If $u_n = u_0$ and $n \geq 3$ then $P$ is called cycle.

The strength of a path $P$ is defined as the minimum of $\mu(u_{i-1}, u_i)$, $i = 1, 2, 3 \ldots n$ and it is denoted by $S(P)$.

Let $G : (\sigma, \mu)$ be a fuzzy graph. The fuzzy graph $G : (\sigma, \mu) - (x, y)$ is obtained from $G : (\sigma, \mu)$ by replacing $\mu(x, y)$ by 0. That is, we remove the edge $(x, y)$ in $G : (\sigma, \mu)$.

A fuzzy graph $H : (\tau, \rho)$ is said to be a fuzzy subgraph of $G : (\sigma, \mu)$ if $\tau(u) \leq \sigma(u)$ for all $u \in V$ and $\rho(u, v) \leq \mu(u, v)$, for all $u, v \in V$.

A fuzzy subgraph $H : (\tau, \rho)$ is said to be a spanning fuzzy subgraph of $G : (\sigma, \mu)$ if $\tau(u) = \sigma(u)$, for all $u \in V$.

A fuzzy graph $G : (\sigma, \mu)$ is called fuzzy forest if it has a fuzzy spanning sub graph $F : (\sigma, \nu)$ which is a forest, if $(x, y) \in G$ but $\notin F$, there is a path $Q$ in $F : (\sigma, \nu)$ $F$ between $x$ and $y$ such that $s(Q) > \mu(x, y)$. A connected fuzzy forest is called a fuzzy tree.

A maximum spanning tree of a connected fuzzy graph $G : (\sigma, \mu)$ is a fuzzy spanning subgraph $T : (\sigma, \nu)$ such that its underlying crisp graph $T^*$ is a tree, and for which $\sum_{u \neq v} \nu(u, v)$ is maximum.

Now, we need the following terms in $G : (\sigma, \mu)$ which are used in the proposed algorithm:

Let $u$ and $x$ be two nodes in $G$. The node $u$ is said to be a strong neighbour of the node $x$ if $\mu(x, u)$ is the maximum of $\{ \mu(x, u) : u \in V is a neighbour of x \}$.

Let $u$ and $x$ be two nodes in $G$. The node $u$ is said to be a weak neighbour of the node $x$ if $\mu(x, u)$ is the minimum of $\{ \mu(x, u) : u \in V is a neighbour of x \}$.
6.5 MAXIMUM SPANNING TREE ALGORITHM

Now, we propose the following new algorithm namely, neighbour search algorithm for finding a maximum spanning tree of a given fuzzy graph.

The proposed algorithm proceeds as follows:

ALGORITHM

Let \( G: (\sigma, \mu) \) be a fuzzy graph with \( n \) vertices and \( m \)-edges

STEP 1: Collect all the pendent edges of \( G: (\sigma, \mu) \) and form a set. Let it be \( S \).

STEP 2: If \( |S| = n - 1 \), stop the computation and \( T_\circ: (\sigma, \mu) \) where \( T_\circ = (V, S) \) is a fuzzy spanning tree of \( G: (\sigma, \mu) \). If not, move to the Step 3.

STEP 3: Find an edge \( e_1 = (u_1, u_2) \) in \( G_1: (\sigma, \mu) = G: (\sigma, \mu) - S \) such that \( \mu(e_1) = \mu(u_1, u_2) \) is maximum. If more than one occur, select any one edge.

STEP 4: Construct \( T_1 = T_\circ + e_1 = (V, S_1) \) where \( S_1 = S \cup e_1 \).

If \( |S_1| = n - 1 \), stop the computation and \( T_1: (\sigma, \mu) \) is a fuzzy spanning tree of \( G: (\sigma, \mu) \). If not, move to the Step 5.

STEP 5: Find an edge \( e_2 \) whose one end vertex is an end vertex of the edge \( e_1 \) in the graph \( G_2: (\sigma, \mu) = G_1: (\sigma, \mu) - e_1 \) such that \( w(e_2) \) is the maximum of \( \{ \mu(u, u_r), \mu_r \) is a strong neighbour of a non-isolated end vertex \( u \) of the edge \( e_1 \) in the graph \( G_2: (\sigma, \mu) \} \) with \( T_1 + e_2 \) not containing a cycle.

STEP 6: Construct \( T_2 = T_1 + e_2 = (V, S_2) \) where \( S_2 = S_1 \cup e_2 \).

If \( |S_2| = n - 1 \), stop the computation and \( T_2: (\sigma, \mu) \) is a fuzzy spanning tree of \( G: (\sigma, \mu) \). If not, move to the Step 7.
STEP 7: Find an edge $e_3$ whose one end vertex is an end vertex of the edge $e_1$ or $e_2$ in the graph $G_3 : (\sigma, \mu) = G_2 : (\sigma, \mu) - e_2$ such that $w(e_3)$ is the maximum of $\{ \mu(u, u_r), \mu_r \}$ is a strong neighbour of a non-isolated end vertex $u$ of the edge $e_1$ or $e_2$ in the graph $G_3 : (\sigma, \mu)$ with $T_2 + e_3$ not containing a cycle.

STEP 8: Construct $T_3 = T_2 + e_3 = (V, S_3)$ where $S_3 = S_2 \cup e_3$.

If $|S_3| = n - 1$, stop the computation and $T_3 : (\sigma, \mu)$ is a fuzzy spanning tree of $G : (\sigma, \mu)$. If not, move to the Step 7.

STEP 9: Continue the Step 7 and the Step 8 for $G_4 : (\sigma, \mu) = G_3 : (\sigma, \mu) - e_3$ and its reduced fuzzy graphs.

Now, we prove the maximality of a fuzzy spanning tree of a fuzzy graph $G : (\sigma, \mu)$ obtained by the proposed algorithm.

**Theorem 6.5.1** A fuzzy spanning tree obtained from a connected fuzzy graph $G : (\sigma, \mu)$ by the neighbour search algorithm is a maximum spanning tree.

**Proof**: Let $U : (\sigma, \mu)$ be a fuzzy spanning tree obtained from $G : (\sigma, \mu)$ by the neighbour search algorithm.

Let $T : (\sigma, \mu)$ be a fuzzy spanning tree of $G : (\sigma, \mu)$.

Assume that $w(U : (\sigma, \mu)) < w(T : (\sigma, \mu))$.

This implies that there exist an edge $e = (u, v)$ in $T$, but $e \notin U$ such that $\mu(e) = \mu(u, v)$ is greater than for the weight of the path connecting the vertices $u$ and $v$ in $U : (\sigma, \mu)$.

Now, since $e \notin U : (\sigma, \mu) , u$ is not a strong neighbour of $v$ in $G : (\sigma, \mu)$ and there exist strong neighbours for $u$ and $v$ in $G : (\sigma, \mu)$, $u_1$ and $v_1$ respectively such that $(u, u_1)$ and $(v, v_1)$ are in $U : (\sigma, \mu)$.
Since strong neighbour for u in $G:(\sigma, \mu)$ is $u_1$ and the strong neighbour for v in $G:(\sigma, \mu)$ is $v_1$, such that $\mu(u, u_1) > \mu(u, v)$ and $\mu(v, v_1) > \mu(u, v)$, the weight of the path connecting u and v containing $(u, u_1)$ or $(v, v_1)$ is greater than $\mu(u, v)$ which is contradiction to the assumption.

Therefore, $w(U:(\sigma, \mu)) \geq w(T(\sigma, \mu))$, for all fuzzy spanning tree $T:(\sigma, \mu)$ of $G:(\sigma, \mu)$.

This implies $U:(\sigma, \mu)$ is a maximum spanning tree of $G:(\sigma, \mu)$.

This completes the theorem.

**Remark 6.5.1** Because one edge is added at each iteration, the neighbour search algorithm completes in $m-k+1$ iterations if the fuzzy graph $G:(\sigma, \mu)$ has k-dependent edges. Therefore, the time complexity of the algorithm is $O((m-k)^2)$. Note that the time complexity of the Prim’s algorithm is $O(n^2)$ where n is the number of vertices and the time complexity of the Kruskal’s algorithm is $O(m \log n)$ where n is the number of vertices and m is the number of edges.

**Example 6.5.1** Consider the following fuzzy graph $G:(\sigma, \mu)$

CLAIM: To determine a maximum spanning tree of the given graph.

Now, the number of edges = 32 and the number of vertices = 23.

Now, since and by the Step 1. and the Step 2., we have

$$S = \{(v_3, v_{18}), (v_6, v_{19}), (v_{10}, v_{20}), (v_{13}, v_{16}), (v_{15}, v_{21}), (v_{14}, v_{22}), (v_{11}, v_{23}), (v_{14}, v_{17})\}$$

and $T_o = (V, S)$.

Now, since $\mu(v_4, v_8)$ is maximum in the reduced graph

$G-\{(v_3, v_{18}), (v_6, v_{19}), (v_{10}, v_{20}), (v_{13}, v_{16}), (v_{15}, v_{21}), (v_{14}, v_{22}), (v_{11}, v_{23}), (v_{14}, v_{17})\} = (v_4, v_8)$

and by the Step 3. and the Step 4., we have $T_1 = T_0 + (v_4, v_8)$. 

89
Now, by the Step 5. and the Step 6., we have \( T_2 = T_1 + (v_3, v_4) \).

Now, using the Step 9., we have \( T_3 = T_2 + (v_3, v_6) \).

Now, by the Step 9., we have \( T_4 = T_3 + (v_1, v_3) \).

Now, using the Step 9., we have \( T_5 = T_4 + (v_5, v_8) \).

Now, by the Step 9., we have \( T_6 = T_5 + (v_5, v_9) \).

Now, using the Step 9., we have \( T_7 = T_6 + (v_9, v_{12}) \).

Now, using the Step 9., we have \( T_8 = T_7 + (v_2, v_5) \).
Now, using the Step 9., we have $T_9 = T_8 + (v_4, v_7)$

Now, using the Step 9., we have $T_{10} = T_9 + (v_6, v_{10})$

Now, using the Step 9., we have $T_{11} = T_{10} + (v_{12}, v_{15})$

Now, using the Step 9., we have $T_{12} = T_{11} + (v_{10}, v_{13})$

Now, using the Step 9., we have $T_{13} = T_{12} + (v_7, v_{11})$

Now, using the Step 9., we have $T_{14} = T_{13} + (v_{12}, v_{14})$

and we stop the computation since the number of edges in $T_{14}$ is 22.

Now, by the Step 6., $T_{14}$ is a maximum spanning tree of the given fuzzy graph and its total weight or membership value is 3.85.

Figure 6.5.2 Fuzzy maximum spanning tree
6.6 CONCLUSION

A weighted connected simple graph and a fuzzy graph are considered in this chapter. Spread search algorithm and neighbour search algorithm are presented for determining maximum spanning tree problems in weighted connected simple graph and fuzzy graph respectively. The proposed algorithm is based on the concept namely, strong neighbour and weak neighbour of a vertex in the graph and, it totally differs all other existing algorithms. The time complexity of this two algorithms is $O((m-k)^2)$ where $m$ is the number of edges and $k$ is the number of pendent edges.