Algorithm 8.2 (Depth-First Search)

1. Let $S = \{v\}$ [v is the root], $T = \emptyset$ [initially there are no edges in the tree]; $b^* = v$ [our initial branch vertex]; $p^* = v$ $l(v) = 0$ [label root as vertex 0]; $i = 1$ [initializes variable i used to help label the vertices]; $U = V(G) - \{v\}$ [keeps track of unlabeled vertices].
2. While $N(b^*) \cap U \neq \emptyset$ [b^* has unlabeled neighbors] do
   begin
   label the next unlabeled neighbor w of b^* by i; place b^*w in T; remove w from U; let $p^* = b^*$ [helps in backtracking]; $b^* = w$ [new branch vertex]; $i = i + 1$ [increment i for future labeling].
   end.
3. $b^* = p^*$[backtrack].
4. If $b^* = v$ and $N(b^*) \cap U = \emptyset$ [v has no unlabeled neighbors], stop. A spanning tree for the component containing the root v has been found. Otherwise, repeat step 2. If $U = \emptyset$, then the tree found is a spanning tree of all of G. If $U \neq \emptyset$, then G is disconnected.

```
#Depth First Search
def DFS(self,v):
    if v in self.Graph:     #check that vertex v is in the graph
        #step 1
        graph = copy.deepcopy(self.Graph)       #create copy of self.Graph
        T = []                                  #empty list of edges
        bstar = v                               #initial branch vertex
        pstar = v
        history = [v]                           #history of vertices that are used as bstar
        label = []                              #list to hold labels
        label.append(v)                         #label v as 0
        U1 = graph.keys()                       #keys of dict
        U = []                                  #list of unlabeled vertices

        for item in U1:
            U.append(item)
        U.remove(v)         #remove v since it is labeled
        U.sort()

        neighbors = graph[v]        #list of vertices adjacent to v
        neighbors.sort()
        intersection = list(set(U) & set(neighbors))        #U intersect neighbors
        intersection.sort()                                 #finds unlabeled niehgbors

        #step 2
        while (True):
            while (intersection):
                w = intersection[0]
                label.append(w)          #label neighbor of bstar
                T.append((bstar,w))      #add edge to T
                U.remove(w)              #remove labeled vertex from unlabeled list
                pstar = bstar            #help for backtracking
                bstar = w
                history.append(bstar)       #update history list
                neighbors = graph[bstar]    #get neighbors
                neighbors.sort()
                intersection = list(set(U) & set(neighbors))    #get intersect of U and neighbors
                intersection.sort()

            neighbors = graph[bstar]            #update neighbors
            neighbors.sort()
            intersection = list(set(U) & set(neighbors))     #used to check for halting condition
            intersection.sort()

            if (len(intersection) == 0):    #if U intersect neighbors = []
                num = history.index(bstar)
                bstar = history[num-1]      #bstar = pstar

            if (bstar == v and intersection == [] and len(U) == 0): #halting condition
                return T        #return spanning tree

            if (intersection != []):
                history.append(bstar)       #update history list
```