SUBSET SUM PROBLEM

IMPLEMENTATION USING BFS

In [6]:

```python
def SUBSETSUMBFS( graph, target ):
    queue = [ i for i in graph ] #creating a queue using the keys of the dictionary 'graph'
    visited = set()

    while queue: #while queue not empty
        currentnode = queue.pop(0)
        visited.add(currentnode)
        print("----------------------------------")
        print(f"Current node is { currentnode } ")

        for neighbour in graph[currentnode]: #neighbouring nodes of current node
            if neighbour not in visited: #checks if neighbouring nodes are visited or not
                print(f"Checking if { currentnode } + { neighbour } equals to { target }")
                if currentnode + neighbour == target: #checks if currentnode + neighbour node equals target
                        print(f"{ currentnode } + { neighbour } = { target }")
                        result.append(( currentnode, neighbour )) #return the pairs
                queue.append( neighbour ) #add neighbour to the queue if not already visited

    return None

graph = { 1: [ 2, 3 ], 2: [ 5 ], 3: [ 7, 6 ] , 4: [ 5 ] , 5: [ 2, 4 ], 6: [ 3 ] , 7: [ 3 ] }
target = 9
result = []
SUBSETSUMBFS( graph, target ) #function call
print( result )
```

```
----------------------------------
Current node is 1
Checking if 1 + 2 equals to 9
Checking if 1 + 3 equals to 9
----------------------------------
Current node is 2
Checking if 2 + 5 equals to 9
----------------------------------
Current node is 3
Checking if 3 + 7 equals to 9
Checking if 3 + 6 equals to 9
3 + 6 = 9
----------------------------------
Current node is 4
Checking if 4 + 5 equals to 9
4 + 5 = 9
----------------------------------
Current node is 5
----------------------------------
Current node is 6
----------------------------------
Current node is 7
----------------------------------
Current node is 2
----------------------------------
Current node is 3
----------------------------------
Current node is 5
----------------------------------
Current node is 7
----------------------------------
Current node is 6
----------------------------------
Current node is 5
[(3, 6), (4, 5)]
```

IMPLEMENTATION USING DFS

In [9]:

```python
result = []

def DFS( node, graph, target, visited, result ):
    if node in visited: #checks if node is visited or not
        return result

    visited.add(node) #add node to visited

    for neighbour in graph[node]: #traverse neighbouring nodes of node
        if neighbour not in visited: #checks if neighbour is visited node or not
            print(f"Checking if { node } + { neighbour } equals to { target }")
            if node + neighbour == target: #checks if node + neighbour = target
                print(f"{ node } + { neighbour } = { target }")
                result.append(( node, neighbour )) #returns the pair

            DFS( neighbour, graph, target, visited, result ) #function call

def SUBSETSUMDFS( graph, target ):
    visited = set()
    result = []
    for node in graph: #for keys in the graph
        print("-------------------------------------")
        print(f"Current node is { node }")
        print("-------------------------------------")
        if node not in visited: #checks if node is not in visited
            DFS( node , graph, target, visited, result )
    return result

graph = { 1: [ 2, 3 ] , 2: [ 5 ], 3: [ 7, 6 ], 4:[ 5 ], 5: [ 2, 4 ] , 6:[ 3 ], 7 : [ 3 ] }
target = 9
result = SUBSETSUMDFS( graph, target ) #function call
print( result )
```

```
-------------------------------------
Current node is 1
-------------------------------------
Checking if 1 + 2 equals to 9
Checking if 2 + 5 equals to 9
Checking if 5 + 4 equals to 9
5 + 4 = 9
Checking if 1 + 3 equals to 9
Checking if 3 + 7 equals to 9
Checking if 3 + 6 equals to 9
3 + 6 = 9
-------------------------------------
Current node is 2
-------------------------------------
-------------------------------------
Current node is 3
-------------------------------------
-------------------------------------
Current node is 4
-------------------------------------
-------------------------------------
Current node is 5
-------------------------------------
-------------------------------------
Current node is 6
-------------------------------------
-------------------------------------
Current node is 7
-------------------------------------
[(5, 4), (3, 6)]
```

IMPLEMENTATION USING SET

In [12]:

```python
set1 = { 1, 2, 3, 5, 6, 7 }
target = 9

def SUBSETSUM( set1, target):
    HashMap = {}
    for i in set1:
        if i in HashMap:
            result.append((HashMap.get(i), i)) #append ( i, complement ) to the result
        else:
            value = target - i
            HashMap[ value ] = i
            print(f"HashMap = { HashMap } ")
            print("-----------------------------")

result = []
SUBSETSUM( set1, target )
print(f"Final Result = { result }")
```

```
HashMap = {8: 1}
-----------------------------
HashMap = {8: 1, 7: 2}
-----------------------------
```

```
HashMap = {8: 1, 7: 2, 6: 3}
-----------------------------
HashMap = {8: 1, 7: 2, 6: 3, 4: 5}
-----------------------------
Final Result = [(3, 6), (2, 7)]
```