[DOCUMENT TITLE]

CS 234

JIADONG MAI

20557203

Assignment 4

Q3

a)

```
              2
          /       \
       7             4
    /    \         /    \
 9         12    10      8
```

b)

```
              12
           /       \
        9             10
    /       \       /     \
 4            8   7        2
```

c)

```
              12
           /       \
          9          10
      /     \       /    \
     4       8     7      2
```

d)

```
              8
           /     \
        7           9
      /               \
    4                   12
  /                    /
2                    10
```

e)

```
              8
            /   \
          4       10
        /   \    /   \
      2      7  9     12
```

f)

if inserting [2,7,4,0,1,3] into an empty AVL Tree:

```
              2
            /   \
          1       3
        /        /  \
      0         4    7
```

If inserting [2,7,4,0,1,3] into an AVL Tree in e:

```
                  4
              /       \
           2             8
         /   \         /    \
       0      2      7       10
        \      \    /  \    /  \
         1      3  4    7  9    12
```

Q4

a)

```
def bi_partition (graph):
    vertices = graph[V]
    edges = graph[E]
    v = {}
    v1 = []
    v2 = []
    v1.append(vertices[0])
    while ((len(v) >=  len(vertices))
        for point in v1:
            if v[point] == ‘green’:
                return false
            else:
                v[point] = ‘red’
            for k in edges[point]:
                if k not in v2:
                    v2.append(edges[point])
        for point in v2:
            if v[point] == ‘red’:
                return false
            else:
                v[point] = ‘green
            for k in edges[point]:
                if k not in v1:
                    v1.append(edges[point])
    return True
```

b)

the algorithm is O(n + m), n is the number of vertices and m is the number of edges.

I create a dictionary v, and two empty list called v1 and v2.

I assume the input graph is look like this;

```
a   b   c
|  x  |  x  |
d   e   f
```

First for loop:

Start from the first vertices, ‘a’, put ‘a’ in v1, and mark the ‘a’ into red in dictionary v, v == {‘a’: ‘red’}, and then put the other vertices collected to ‘a’ based on the edges, and put these vertices in v2 list, v2 == [‘d’, ‘e’].

And then second for loop:

For each vertices in v2, and mark it as green in the dictionary v, v == {‘a’: ‘red’, ‘d’: ‘green’, ‘e’: ‘green’}. and add all the vertices collected the v2 vertices into v1 list, v1 == [‘a’, ‘b’, ‘c’].

Keep the while loop, for each vertices in v1, and mark it as red in the dictionary v, v == {‘a’: ‘red’, ‘b’: ‘red’, ‘c’: ‘red’, ‘d’: ‘green’, ‘e’: ‘green’} and add all the vertices collected the v1 vertices into v2 list, v2 == [‘d’,’e’,’f’].

Continue, for each vertices in v2, and mark it as green in the dictionary v, v == {‘a’: ‘red’, ‘b’: ‘red’, ‘c’: ‘red’, ‘d’: ‘green’, ‘e’: ‘green’, ‘f’: ‘green’}. and add all the vertices collected the v2 vertices into v1 list, v1 == [‘a’, ‘b’, ‘c’].

And then, the len of dictionary v is equal or bigger to the number of vertices in the graph, return true.

If there have any vertice in already mark color, and the color covert to is list color, v1 == red or v2 == green. Then there is no bipartite, return false.

Q5

a)

Vertex of graph G: {‘A’, ‘B’,’C’,’D’,’E’,’F’,’G’}

Edge of graph G: {(‘A’,’B’), (‘B’,’C’), (‘C’,’A’), (‘C’,’D’), (‘C’,’G’), (‘D’,’E’), (‘D’,’G’), (‘E’,’F’), (‘F’,’E’), (‘F’,’G’), (‘G’,’B’)}

b)

adjacency list:

```
   G = {‘A’: [‘B’],
          ‘B’: [‘C’],
          ‘C’: [‘A’, ‘D’, ‘G’],
          ‘D’: [‘E’, ‘G’],
          ‘E’: [‘F’],
          ‘F’: [‘E’, ‘G’],
          ‘G’: [‘B’]}
```

c)

adjacency matrix:

```
  M = [[0,1,0,0,0,0,0],
         [0,0,1,0,0,0,0],
         [1,0,0,1,0,0,1],
         [0,0,0,0,1,0,1],
         [0,0,0,0,0,1,0],
         [0,0,0,0,1,0,1],
         [0,1,0,0,0,0,0]]
```