Answer to the question no 4

Time complexity of BFS using adjency list:

```
visited = [] , queue []                          <-- O(1)

BFS (visited, graph, node, endpoint):
    output = "Places: "                          <-- O(1)
    visited <- append (node)                     <-- O(1)
    queue <- append (node)                       <-- O(1)
    while queue not empty                        <-- O(X)
        m <- que.pop(0)                          <-- O(1)
        print m                                  <-- O(1)
        output += m + " "                        <-- O(1)
        if m = endpoint break                    <-- O(1)
                                                 } O(C1)
        for each i of m in graph                 <-- O(Y)
            if i not in visited                  <-- O(1)
                visited <- append (i)            <-- O(1)
                queue <- append (i)              <-- O(1)
                                                 } O(C2)
```

Analysis:

Let, V = number of vertex in the graph

E = number of edges in the graph

We know that, while doing BFS for adjacency lists, the inner loop [O(Y)] will discover all its neighbors by traversing its adjency list twice (because given graph in undirected) in linear time. So, For worse case scenario, the time complexity of inner loop will be,

$$O(Y) = O(2E)$$
$$= O(E)$$

Now, each node can have different number of edge. The outer while loop will run as long as the queue remain full. So, the while loop will run V times.

Now we will calculate the total time complexity.

Let,

$$N = C_1 + \frac{C_2 E}{V}$$

This N is the average time complexity of task done inside the while loop in each eteration. This N will be done V times. So time complexity of outer while loop will be,

$$O(x) = O\left(V \cdot \left(C_1 + \frac{C_2 E}{V}\right)\right)$$

$$= O\left(C_1 V + C_2 E\right)$$

Therefore,

the total time complexity of my BFS using adjacency list is,

$$O(1) + O(1) + O(1) + O(1) + \quad O\left(C_1 V + C_2 E\right)$$

Discarding all the constant, Time complexity $= O(V+E)$

Time complexity of BFS using adjacency Matrix:

In BFS, for each node, we will have to traverse entire row of length V. each row will also have V elements. So, traversing through the entire matrix will take $(V \times V)$ time.

So the time complexity of BFS for matrix is $O(V \times V) = O(V^2)$

Time complexity of DFS for adjacency list:

In worse case scenario, DFS will have a similar time complexity as BFS.

At first dfs (graph, endpoint) will be called. This function has a for loop which will run for each node in graph. Assumming V = Total vertices of graph and E = total edges of the graph, we can say that

the for loop will run V times inside dfs function.

Now, dfs_visit (graph, s) is a recursive function which will traverse through the edges. Since this is an undirected graph, each

edge will appear twice in adjacency list.

So, the total time complexity of DFS for adjacency list will be,

$$O(V) + O(2E)$$
$$= O(V) + O(E)$$
$$= O(V+E)$$

Time complexity of DFS using adjacency matrix;

The adjacency matrix is a $V \times V$ matrix

In worst case scenario, the algorithm will have to traverse through entire $V \times V$ matrix.

So the time complexity of DFS using matrix will be, $O(V \times V)$

$$= O(V^2)$$

Here, the rival will get to the victory road first.

Because, Ash will go to victory road like following,

$1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 7 \rightarrow 11 \rightarrow 6 \rightarrow 12$

↑ Destination

He will travel 7 different cities before reaching to victory road.

On the other hand, the rival will use DFS Algorithm. He will go to victory road like following

$1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 7 \rightarrow 11 \rightarrow 12$

He will travel 5 different cities before reaching to victory road.

Since victory road in this graph has a higher depth from pallet town, DFS algorithm will be better in this case.