Homework 4

Problem 1:

- **(b)** I couldn't write a certain modification, but I know that with a proper modification Asymptotic Time Complexity have to be **$O(n^k)$**. It is an expected result because we know that the time complexity of LCS for two sequences is $O(n^2)$.

Problem 2:

- **(a)** The problem is a minimum vertex cover problem:
  - Designing steps are:

```
1) Initialize the result as {}
2) Consider a set of all edges in given graph.  Let the set be E.
3) Do following while E is not empty
...a) Pick an arbitrary edge (u, v) from set E and add 'u' and 'v' to result
...b) Remove all edges from E which are either incident on u or v.
4) Return result

Initialize all vertices as not visited.
Consider all edges one by one
    An edge is only picked when both visited[u] and visited[v]
        Go through all adjacents of u and pick the first not yet visited vertex (We
are basically picking an edge (u, v) from remaining edges.
                Add the vertices (u, v) to the result set.
                We make the vertex u and v visited so that all edges from/to them
would be ignored
Print the vertex cover
```

- **(b)**
  - Average Case: **O(V+E)**, for C=Ø while there is uncovered edge (u, v) Operations:
    add vertex u into C
    add vertex v into C
    return C

  - Worst case: **$O(2^n)$**

Problem 3:

- **(a)**
  - Input: Graph G=(V, E), INT i
  - Output: Spanning tree of G at most i leaves

- **(b)** One of real-life application of MLST is network design. For example, if you have a delivery company, you want to lease cargo trucks to connect them up with each other and the leasing company charges different amounts of money to connect different pairs of cities. If you want a set of routes that connects all your offices with a minimum total cost. It should be a minimum leaf spanning tree.