Homework Assignment #3 Report

20175095 Shin Jisu

1. Elaborating problem statement and requirements in points
   1) We need to find the dead-end signs on the graph and it should be not redundant.
   2) To do this, we should find the leaves that starts approaching to the end.
   3) Dead-end sign might locate between the leaf I mentioned above and the vertex of connected components.
   4) Thus, we should find leaves recursively until it reaches the connected components and compare those to get the dead-end sign.

2. Data structure used and why it has been used

I used some vectors to get the vertexes of the given input. It is useful to push the input value, for example, vector at the index v, we can put all the vertexes(w) that are connected to v.
Also I used pair to store the connected nodes where dead-end sign locates on the answer vector and print the output from there.

3. Algorithm used to solve the problem and why

I first declare the vector and put all leaves recursively on it until there exists no more leaves.
Additionally, the nodes like the tree's root (the tree here means the tree that gets separate from the SCC) are collected and use DFS(depth-first search) algorithm on it.
We do this because it is important in here to classify SCC and leaves which approaches to the end.
After classification, check where the dead-end sign should be set and put those vertexes on new vector.
Sort this vector in ascending order of v-locations and print the input.

4. Briefly explain an alternate solution, if there is

   1) We need to separate the graph to the connected components(SCC).
   2) If the graph is consisted of the tree, we can just set all the leaf to be dead-end.
   3) If not, we start from the leaf and tries to remove the leaf iteratively and repeats it until there does not exist any leaf in the graph.

4) When the graph is consisted of the tree(1-2), it is already done. If is not(1-3), dead-end will be placed on the edge which is consisted of the one vertex removed and the other vertex not.

5. Calculation of time complexity of the algorithm used

On the major algorithm, there exists for loop inside the for loop and when I check the loop, the time complexity will be O(n+V) and V in here is the number of the edges.