CS 4820, Spring 2018 **Homework 2, Problem 2**

Name: Rongguang Wang
NetID: rw564
Collaborators: Siyuao Liu (sl2928); Yihao Chen (yc2288)

(2) *(10 points)* Suppose we are given a tree, T, and a set of paths in T. Two paths are called *compatible* if they have no vertices in common. Design an algorithm to select a maximum cardinality subset of the paths, such that every two paths in the subset are compatible.

Solution:

The input is a tree and a set of paths in the tree. The problem is aiming to find a subset that contains the maximum number of non-conflicting paths. The tree can be treated as many intervals starting from the root and end at the leaves. So, the root of the tree can be the start point. To cover most of the vertices, pick the vertex that furthest to the root as the end of the path in each iteration. Delete any paths that intersect with the closest vertex to the root in the selected path, which avoid the conflicting problem in following path choosing. Then, do the pick up and delete work for each iteration.

Basically, the algorithm works as following:
1) Find the path that contains the furthest vertex from the root in the tree.
2) Delete any path that intersects with the closest vertex to the root in the selected path.
3) Repeat the step 1 and step 2 until there is only one vertex in the tree.

Algorithm's Correctness:
Basically, there are two points to prove for the algorithm. Firstly, any path selected using the algorithm is not conflict with the other paths in the subset. In another word, no two paths in the selected subset are conflict, which means they have no-common vertex. Assume two paths were selected using the algorithm. After the fist path was picked, the paths that intersect with the closest vertex to the root were deleted from the path set. In this case, the second selection wont have chance to pick up a vertex that inside the first path because previous selection chose the furthest point from the root. Therefore, no two paths in the selected subset are conflict.

Secondly, the subset of paths selected using the algorithm is the maximum subset. Assume that there is an optimal selected subset of paths S and a subset S' selected by using this algorithm. Every path in S can be mapped to the minimum nodes in S'. The minimum nodes are the closest vertices to the root in S'. Therefore, the subset S' contains the maximum non-conflicting paths.

Running Time Analysis:
Pre-processing:
To find all nodes (vertices) in the paths provided in the given set, a stack can be implemented using linked list which records the ancestor of the end nodes (furthest point from the root) in the paths. This procedure will take $O(n)$ time. Additionally, the tree will be traversed for each given path in the set. The worst case is that n paths were given and the time complexity is $O(n^2)$.

Path selection:
The path can be selected by using Breadth First Search (BFS) which takes $O(n)$ time.

Path deletion:
The common vertex searching takes linear time in linked list which is $O(n)$.
Therefore, the time complexity to implement the algorithm is $O(n^2)$.