Princeton University

COS 217: Introduction to Programming Systems

DT Algorithms

1. int DT_insertPath(char* path);

Step 1). If the DT is not initialized, return INITIALIZATION_ERROR.

Step 2) Search for farthest Node reachable from the root following the given path.

    If Node is NULL, but the root is not NULL, returns CONFLICTING_PATH.

    If Node representing path already exists, returns ALREADY_IN_TREE.

    If the root is NULL, set Node to NULL.

    Let Node be the Parent of the child node that to be added in the DT.

Step 3) For the next node on the rest of the path (excluding the prefix from Step 2.)
    Create a new node, if there is an allocation error in creating any of the new node or their fields, returns MEMORY_ERROR; set the parent link of the new node to Parent.
    If Parent is NULL, set the root to the new node.
    If Parent is not NULL, add the new node to the children of Parent and keep the children in increasing order by path; If there is an error occurs, returns PARENT_CHILD_ERROR

    Increment the count of the DT by 1.
    Set Parent to the new node, repeat Step 3 if there are additional nodes on the path.

Step 4) Returns SUCCESS.

2. int DT_rmPath(char* path);

Step 1). If the DT is not initialized, return INITIALIZATION_ERROR.

Step 2). Search for farthest Node reachable from the root following the given path. If Node is NULL, returns NO_SUCH_PATH.

Step 3) If Node's parent is NULL (the path is the root in this case), set root to NULL; if Node's parent is not NULL, remove Node from the children of its parent. (This will disconnect Node from the DT.)

Step 4) (Recursively destroy the entire hierarchy of nodes rooted at Node, including Node itself.)

```
For each child node in the children of Node,
    If it has more children, set Node to be the child node; repeat Step 4.
Free its children, free its path, free the child node.
```

Step 5) Decrement count of the DT by the number of nodes removed and Returns SUCCESS.