HW2 Report
Student/ID: 鄭凱翔/0746007

I. Environment
OS: macOS 10.14.1
Language: Python3.6
Editor: Visual Studio Code

II. Method

- **Parse string from input.txt and do relevant operation in main function**
  In order to parse Input.txt, the script reads the first line of the text file, getting how many operations, either insertion or deletion, to be executed. The script then instantiate a BRTree class object, which are going to detail below, and loop through each operation, reading data in and doing insertion or deletion.

- **BRTree class**
  The logic behind the black-red tree algorithm is complicated, so fig 1 shows the abstract logic within functions and the relationship between each function for better understanding.
  - Insert
    - Initiate a new node, set y to nil and x to root node. x is going to be pioneer for y to see if x reach the leaf node when finding the parent for new node
    - In a while loop, if x is not nil node, we assign current x to the y, then we determine x to go right or left, according to the value of new node is greater or lower than current node. If greater than x go right, else go left till x is the nil node. And we link y and new node.
    - if the tree we insert is an empty tree, the new node is the root of this tree. New node is right child of y if its value smaller than y, else is left child of y
    - We finally need to fix up for violating tree properties, and following fix-up procedures just show the case when parent node is at left of its parent
    - In a while loop, if parent's color is red, we keep doing follow operation:
      - get the uncle node
      - fix up depend on uncle's color
      - uncle is red [case 1]
        set both parent and uncle's color to black. parent's parents to red
      - uncle is black [case 2 case 3]
        [case 2] if current node is at the right of its parent, let parent to be current node and do the left rotation, which turn case 2 to case 3
        [case 3] if current node is at the left of its parent, color parent node black and parent's parent red and do right rotate
  - Delete
    - If the node to be deleted has less than two child, just transplant it with child node

- If the node to be deleted has two child, we have to find minimum node y within the right subtree of the current node to transplant current node
- transplant y with y child
- if y is not child of current node, having y's child transplant y first, then link y and current node 's child with each other
- transplant current node with y
- link left child of current node and y with each other
- set y color equal current node's color
- fix up if origin color is black
- We finally need to fix up for violating tree properties, and following fix-up procedures just show the case that current node is at left of its parent
- [case1] if sibling's color is red, color it black, and color parent red, then do the left rotation on parent.
- [case 2] if sibling's left and right are black, then color sibling red. set x to parent
- [case 3] sibling's right is black and left is red, color left black, and do right rotation on sibling. set sibling to parent right. After this operation, case 3 will turn into case 4
- [case 4] assign parent's color to sibling's, color parent and sibling right black. Then do left rotation on parent. set x to root
- In the end, color x black

III. Results

Please refer to the output file.

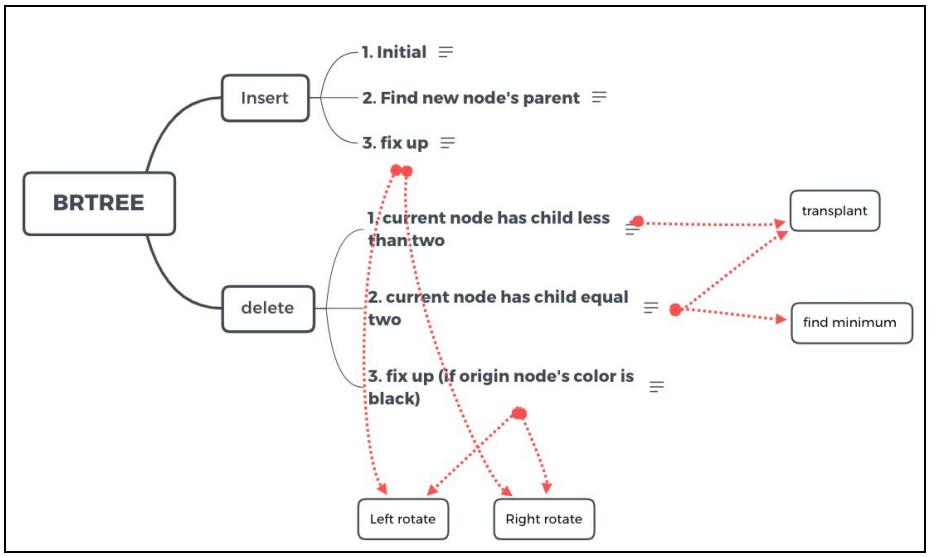

Fig 1. Hierarchy of BRTree class