Projekt 2

Celem projektu jest implementacja oraz porównanie drzew binarnych typu BST oraz Splay.

Poniżej znajduje się moja wersja drzewa Splay

```
import random

class Node:
    def __init__(self, data=None, par=None):
        self.data = data
        self.left = self.right = None
        self.parent = par

class Tree:
    def __init__(self):
        self.dummy = Node('u')
        self.root = None

    def Rotate(self, B):
        if B == self.dummy or B is None or B == self.root:
            return

        A = B.parent
        P = A.parent

        if A.left == B:
            B.parent = P
            if P:
                if P.left == A:
                    P.left = B
                else:
                    P.right = B

            Beta = B.right

            A.parent = B
            B.right = A

            A.left = Beta
```

```
        A.left = Beta
        if Beta:
            Beta.parent = A

    else:
        B.parent = P
        if P:
            if P.left == A:
                P.left = B
            else:
                P.right = B

        Beta = B.left

        A.parent = B
        B.left = A

        A.right = Beta
        if Beta:
            Beta.parent = A

    if P is None:
        self.root = B
```

```
def splay(self, node):
    while node.parent is not self.dummy:
        if node.parent.parent is self.dummy:

            self.Rotate(node)
        else:
            parent = node.parent
            grandparent = parent.parent
            if node == parent.left:
                if parent == grandparent.left:

                    self.Rotate(parent)
                    self.Rotate(node)
                else:

                    self.Rotate(node)
                    self.Rotate(node)
            else:
                if parent == grandparent.left:

                    self.Rotate(node)
                    self.Rotate(node)
                else:

                    self.Rotate(parent)
                    self.Rotate(node)
    self.root = node
```

```
def find(self, node, value):
    if node is None:
        return None, False

    if value == node.data:
        return node, True

    if value < node.data:
        if node.left:
            return self.find(node.left, value)
    else:
        if node.right:
            return self.find(node.right, value)

    return node, False

def splay_find(self, value):
    x, z = self.find(self.root, value)
    if x is not None:
        self.splay(x)
    return x, z
```

```
def append(self, value):
    if self.root is None:
        self.root = Node(value, self.dummy)
        self.dummy.right = self.root
        return

    node, found = self.splay_find(value)

    if found:
        return

    new_node = Node(value)
    if value < node.data:
        new_node.left = node.left
        new_node.right = node
        node.left = None
    else:
        new_node.right = node.right
        new_node.left = node
        node.right = None

    if new_node.left:
        new_node.left.parent = new_node
    if new_node.right:
        new_node.right.parent = new_node

    new_node.parent = self.dummy
    self.root = new_node
    self.dummy.right = new_node
```

Teraz porównamy to drzewo z drzewem BST. Zestawię ze sobą czasy tworzenia oraz wyszukiwania. Zmienne będą dodawane na dwa sposoby, raz prawdopodobieństwo wylosowania liczb będzie miało rozkład normalny, a w drugiej wersji jednostajny. Drzewa będą miały po 10 000 węzłów. Poniżej prezentuję te czasy.

Dla rozkładu jednostajnego:

Tworzenie:

Drzewo BST: `0.0338691711797311`

Drzewo SPLAY: `0.17256608241377`

Wyszukiwanie:

Drzewo BST: `2.5546696472168726e-06`

Drzewo SPLAY: `3.38762556829834e-06`

Dla rozkładu normalnego:

Tworzenie:

Drzewo BST: `0.026815298726456531`

Drzewo SPLAY: `0.1361738257865215`

Wyszukiwanie:

Drzewo BST: `3.3976519025635208e-06`

Drzewo SPLAY: `1.98725361876419e-06`

Jak widać dla rozkładu jednostajnego szybsze jest drzewo BST, natomiast dla rozkładu normalnego szybsze jest drzewo SPLAY. Dzieje się tak ponieważ drzewie Splay dzięki operacji splay(self,node) wartości najczęściej wyszukiwane znajdują się bliżej korzenia drzewa.