Zadanie 8. (1,5pkt)

Zaproponuj strukturę danych do pamiętania zbioru liczbowego i wykonywania na nim operacji: insert, delete, mindiff. Ostatnia z tych operacji zwraca jako wynik najmniejszą różnicę między dwoma elementami zbioru.

Drzewo AVL z dodatkowymi polami w każdym wierzchołku:
1. minDiff – najmniejsza różnica 2 liczb tego poddrzewa,
2. minVal – najmniejsza wartość poddrzewa,
3. maxVal – największa wartość poddrzewa.

Po każdej operacji Insert/Delete aktualizujemy najpierw min/maxVal na podstawie min/maxVal dzieci, a następnie minDiff korzystając ze zaktualizowanych wartości min/maxVal oraz wartości minDiff dzieci.

```
// aktualizowanie wartości minimalnej drzewa
UpdateMinVal(T)
if (T.left == null)
        T.minVal = T.key
else
        T.minVal = T.left.minVal

// aktualizowanie wartości maksymalnej drzewa
UpdateMaxVal(T)
if (T.right == null)
        T.maxVal = T.key
else
        T.maxVal = T.right.maxVal

// aktualizowanie minimalnej różnicy drzewa
UpdateMinDiff(T)
if (T == null or (T.left == null and T.right == null))
        T.minDiff = +inf
else if (T.left == null)
        T.minDiff = min(T.right.minDiff, T.right.minVal – T.value)
else if (T.right == null)
        T.minDiff = min(T.left.minDiff, T.value – T.left.maxVal)
else
        T.minDiff = min(T.left.minDiff, T.right.minDiff,
```

T.value – T.left.maxVal, T.right.minVal - T.value)

```
// aktualizuje min/max wartości oraz minDiff po rotacji w prawo
// zakładamy, że wywołuje się w ostatniej linijce standardowego rightRotate
UpdateRightRotate(T)
	UpdateMinVal(T.right) // update y
	T.maxVal = T.right.maxVal // update x
	UpdateMinDiff(T.right) // update y
	UpdateMinDiff(T) // update x

// aktualizuje min/max wartości oraz minDiff po rotacji w lewo
// zakładamy, że wywołuje się w ostatniej linijce standardowego leftRotate
UpdateLeftRotate(T)
	UpdateMaxVal(T.left) // update x
	T.minVal = T.left.minVal // update y
	UpdateMinDiff(T.left) // update x
	UpdateMinDiff(T) // update y

// Balansowanie drzewa AVL
Balance(T)
balance = GetBalance(T)
if (balance > 1 and GetBalance(T.left)>=0) // Left Left
	return rightRotate(T)
else if (balance < -1 and GetBalance(T.left)<=0) // Right Right
	return leftRotate(T)
else if (balance > 1) // Left Right
	T.left = leftRotate(T.left)
	return rightRotate(T)
else if (balance < -1) // Right Left
	T.right = rightRotate(T.right)
	return leftRotate(T)
return T

// Zaktualizuj wartości min/max/minDiff po Insert/Delete
UpdateValues(T)
	UpdateMinVal(T)
	UpdateMaxVal(T)
	UpdateMinDiff(T)
```

```
// Wstawianie elementu do drzewa AVL
Insert(T, key)
if (T == null) // znaleźliśmy liść do którego trzeba wstawić klucz
        return newNode(key)
if (key < T.value) // wstawianie do lewego poddrzewa
        T.left = Insert(T.left, key)
else // wstawianie do prawego poddrzewa
        T.right = Insert(T.right, key)
UpdateValues(T) // zaktualizuj min/maxVal, minDiff
UpdateHeight(T)
return Balance(T)

// Usuwanie elementu z drzewa AVL
Delete(T, key)
if (T == null) // nie znaleźliśmy elementu do usunięcia
        return null
if (key < T.value) // usuwanie z lewego poddrzewa
        T.left = Delete(T.left, key)
else if(key > T.value) // usuwanie z prawego poddrzewa
        T.right = Delete(T.right, key)
else // usuwanie aktualnego elementu
        if (T.left == null)
                return T.right
        else if (T.right == null)
                return T.left
        T.value = T.right.minVal
        T.right = Delete(T.right, T.value)
if (T == null)
        return null
UpdateValues(T) // zaktualizuj min/maxVal, minDiff
UpdateHeight(T)
return Balance(T)

// Tworzenie nowego liścia
```

```
newNode(key)
node.height = 1
node.left = null
node.right = null
node.minVal = key // minimalna wartość drzewa
node.maxVal = key // maksymalna wartość drzewa
node.value = key // wartość korzenia drzewa
node.mindiff = +inf // minimalna różnica drzewa
return node
```