Data Structures – Practical Assignment 1

Gal Oren (302378633, orengal1), Katz Saar (318966249, saarkatz)

# Class documentation:

|  |
| --- |
| WAVLTree |

Fields:

|  |  |  |
| --- | --- | --- |
| Type | Name | Description |
| WAVLNode | **root** | A reference to the root of the tree. Root is null if the tree is empty. |
| **int** | **size** | The number of nodes in the tree. |
| WAVLNode | **maxNode** | A reference to the maximum node of the tree. is null if the tree is empty. |
| WAVLNode | **minNode** | A reference to the minimum node of the tree. is null if the tree is empty. |

Public methods:

|  |  |  |  |
| --- | --- | --- | --- |
| Type | Signature | Description | Time complexity |
| **int** | delete(**int** k) | Deletes the node with the key from the tree.  Returns the number of rebalance operations required or if the node is not in the tree. | Amortized is also because the node is searched for first. Number of re-balance operations are amortized, along with insert, to |
| **boolean** | empty() | Returns if the tree is empty (), false otherwise. |  |
| String[] | infoToArray() | Returns an array of the information in the tree ordered by the keys. |  |
| **int** | insert(**int** k, String i) | Inserts a new node with the key with info into the tree.  Returns the number of rebalance operations required to balance the tree or if a node with key already exists within the tree | Amortized is also because the insertion place is searched for first. Number of re-balance operations are amortized, along with delete, to |
| **int**[] | keysToArray() | Returns an array of the key in the tree in order from smallest to largest. |  |
| String | max() | Returns the information corresponding to the largest key. |  |
| String | min() | Returns the information corresponding to the smallest key. |  |
| String | search(**int** k) | Returns the information corresponding to the key . |  |
| **int** | size() | Returns the size of the tree (The number of key info pairs). |  |
| String | toString() | Rreturns a string representing the tree. |  |

Private methods:

|  |  |  |  |
| --- | --- | --- | --- |
| Type | Signature | Description | Time complexity |
| **static void** | changeRank(WAVLNode node, **int** amount) | Changes the rank of a by , adjusting the differences to the parent and children. |  |
| **static int** | deletionCase(WAVLNode node) | Given a , determines what case of rebalance should be employed to balance the tree after deletion.  0 - No further changes should be made.  1 - Demotion required.  2/3 - Double demotion required, left/right.  4/5 - Rotation required, left/right.  6/7 - Double rotation required, right-left/ left-right. |  |
| **static void** | demote(WAVLNode node) | Changes the rank of by . |  |
| **static** WAVLNode | findInsertionPlace(WAVLNode node, **int** k) | Searches the subtree starting at for the node under which a node with key should be inserted to keep the order of keys within the tree correct. Returns if the key exists in the tree beforehand. |  |
| **static** WAVLNode | findKey(WAVLNode node, **int** k) | Searches for the a node with the key in the subtree starting at . Returns if no such node is found. |  |
| **static int** | insertionCase(WAVLNode node) | Given a , determines what case of rebalance should be employed to balance the tree after insertion.  0 - No further changes should be made.  1 - Promotion required.  2/3 - Rotation required, left/right.  4/5 - Double rotation required, left/right. |  |
| **static** WAVLNode | predecessor(WAVLNode node) | Return the node previous key wise to . Returns null if has the smallest key in the tree. | Amortized on series of predecesors: |
| **static void** | promote(WAVLNode node) | Changes the rank of by |  |
| **static** WAVLNode | rotateRight(WAVLNode node) | Rotates the with its left child. Returns the left child. |  |
| **static** WAVLNode | rotateLeft(WAVLNode node) | Rotates the with its right child. Returns the right child. |  |
| **static** String | subtreeToString(WAVLNode node) | Returns a string representation of the subtree starting at . |  |
| **static** WAVLNode | successor(WAVLNode node) | Return the node next key wise to . Returns null if has the smallest key in the tree. |  |

|  |
| --- |
| WAVLNode |

Fields:

|  |  |  |
| --- | --- | --- |
| Type | Name | Description |
| WAVLNode | **parent** | A reference to the parent. if node has no parent. |
| WAVLNode | **left** | A reference to the left child. if node has no left child. |
| WAVLNode | **right** | A reference to the right child. if node has no right child. |
| **int**[] | **differences** | An array of size with the differences to left and right children. Children which are are considered external leafs and therefore have rank of . |
| **int** | **key** | The key of the node. |
| String | **value** | The info of the node. |

Methods: Time complexity of all methods is

|  |  |  |
| --- | --- | --- |
| Type | Signature | Description |
| WAVLNode | getParent() | Returns the parent. |
| WAVLNode | getLeft() | Returns the left child. |
| WAVLNode | getRight() | Returns the right child. |
| **int** | getLeftDifference() | Returns the left difference. |
| **int** | getRightDifference() | Returns the right difference. |
| String | getValue() | Returns the info of the node. |
| **int** | getKey() | Returns the key of the node. |
| **int** | getDifference(**int** side) | Returns the difference corresponding to . . |
| WAVLNode | getChild(**int** side) | Returns the child corresponding to . . |
| **void** | setParent(WAVLNode parent) | Sets parent to be . |
| **void** | setLeft(WAVLNode left) | Sets the left child to be . |
| **void** | setRight(WAVLNode right) | Sets the right child to be . |
| **void** | setRightDifference(**int** leftDifference) | Sets the difference with the left child to be . |
| **void** | setLeftDifference(**int** rightDifference) | Sets the difference with the right child to be . |
| **void** | setKey(**int** key) | Sets the key to be . |
| **void** | setValue(String value) | Sets the info to be . |
| **void** | setDifference(**int** side, **int** difference) | Sets the difference corresponding to to be . . |
| **void** | setChild(**int** side, WAVLNode child) | Sets the child corresponding to to be . . |
| **boolean** | hasLeftChild() | Returns if the left child isn’t , otherwise. |
| **Boolean** | hasRightChild() | Returns if the right child isn’t , otherwise. |
| **boolean** | hasParent() | Returns if the parent isn’t , otherwise. |
| **boolean** | isLeaf() | Returns if both children are , otherwise. |
| **int** | relationWithParent() | Returns if the node is its parent’s left child, if it is its parent’s right child and if parent is |
| **int** | relationWithChild(WAVLNode node) | Returns if is this node’s left child, if is right child and otherwise. |
| String | toString() | Returns a string representation of the node. |

# Benchmark measurements:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| No. | No. operations | Avg. insert | Avg. delete | Max insert | Max delete |  |
| 1 | 10000 | 2.959000 | 1.611900 | 15 | 9 | 13.28771 |
| 2 | 20000 | 2.958750 | 1.594200 | 16 | 8 | 14.28771 |
| 3 | 30000 | 2.960667 | 1.593267 | 17 | 9 | 14.87267 |
| 4 | 40000 | 2.953925 | 1.605800 | 17 | 8 | 15.28771 |
| 5 | 50000 | 2.946940 | 1.593060 | 18 | 10 | 15.60964 |
| 6 | 60000 | 2.945217 | 1.592300 | 18 | 9 | 15.87267 |
| 7 | 70000 | 2.951514 | 1.596214 | 18 | 9 | 16.09507 |
| 8 | 80000 | 2.946375 | 1.595400 | 19 | 10 | 16.28771 |
| 9 | 90000 | 2.947589 | 1.595844 | 19 | 10 | 16.45764 |
| 10 | 100000 | 2.944130 | 1.595240 | 19 | 11 | 16.60964 |

אנו יודעים כי על פי ההסבר התאורטי סיבוכיות פעולות האיזון ה של ו- היא בעוד שהמקרה הגרוע הוא כאשר הוא גודל העץ. כפי שניתן לראות התוצאות התאורטיות אכן מתקיימות. מספר פעולות האיזון אכן קבוע בגודל העץ בעוד ההכנסה והמחיקה המקסימליות הולכות כמו עד כדי הבדל שנובע מהרנדומליות שבסדר הכנסת האיברים.