|  |  |  |  |
| --- | --- | --- | --- |
| **Function Signature** | **Worst Case** | **Best Case** | **AVL Case** |
| **public** Node insert(**int** key, Node node) | Ο(n) | Ο(1) | Ο(log2(n)) |
| **private int** minKey(Node node) | Ο(n) | Ο(1) | Ο(log2(n)) |
| **private int** maxKey(Node node) | Ο(n) | Ο(1) | Ο(log2(n)) |
| **private int** totalKeyNum(Node node) | Ο(n) | Ο(n) | Ο(n) |
| **private int** totalNodeNum(Node node) | Ο(n) | Ο(n) | Ο(n) |
| **public int** checkAVL(Node node) | Ο(n) | Ο(n) | Ο(n) |

n … Number of Nodes

***public Node insert(int key, Node node);***

***private int minKey(Node node);***

***private int maxKey(Node node):***

Im besten Fall ist ein einseitiger Baum vorhanden und das Einfügen findet auf der anderen Seite statt. D.h. nur 1 Funktionsaufruf und Nodezugriff (root). **→** Ο(1)

Im schlechtesten Fall muss der Knoten bei einem einseitigen Baum auf der längeren Seite nach allen anderen Knoten eingefügt werden. **→** Ο(n)

Im Fall eines AVL-Baumes sind beide Seiten gleich tief mit einer Verdopplung der Knoten per Ebene. **→** Ο(log2(n))

***private int totalKeyNum(Node node);***

***private int totalNodeNum(Node node;***

***public int checkAVL(Node node):***

Sowohl im besten, im schlechtesten, als auch im AVL Fall wird der ganze Baum traversiert. Dies geschieht rekursiv, sodass die die Funktionen per Knoten 1-Mal aufgerufen werden und die Rückgabewerte von den Blattknoten auf bis zur Wurzel übergeben werden. **→** Ο(n)