מחלקת **AVLTree**

**שדות:**

Root – שורש העץ

Min – מצביע לnode בעל הkey המינימלי בעץ

Max – מצביע לnode בעל הkey המקסימלי בעץ

אנו מתחזקים את השדות בפעולות השונות, כאשר נדרש לכך.

**בנאים:**

Public AVLTree() – בנאי של עץ ריק, מעדכן את השורש של המופע הנוכחי להיות מטיפוס AVLNode.

Public AVLTree(int k, String i) – בנאי המקבל מפתח (k) וערך (i), בו אנו מתחזקים את השדות השונים של העץ ושל הnode.

פעולות:

**Public Boolean empty()**

* מה עושה – בודקת אם המופע הנוכחי של המחלקה הוא עץ ריק
* כיצד פועלת – בודקת מהו גובה השורש. במימוש פעולת public AVLNode גובה השורש בעץ ריק הוגדר להיות -1, לכן אם הגובה הוא -1, הפעולה תחזיר שהעץ הוא ריק – כלומר true, ואחרת הוא לא ריק, כלומר תחזיר false.
* סיבוכיות זמן הריצה –
* קריאה לפונקציית עזר –

**Public String search(int k)**

* מה עושה –
* כיצד פועלת –
* סיבוכיות זמן הריצה –
* קריאה לפונקציית עזר –

**Private IAVLNode research(IAVLNode node, int key)**

* מה עושה –
* כיצד פועלת –
* סיבוכיות זמן הריצה –
* קריאה לפונקציית עזר -

**private IAVLNode TreePosition(int k,IAVLNode node)**

* מה עושה –
* כיצד פועלת –
* סיבוכיות זמן הריצה –
* קריאה לפונקציית עזר -

**Public int insert(int k, String i)**

* מה עושה –
* כיצד פועלת –
* סיבוכיות זמן הריצה –
* קריאה לפונקציית עזר -

**Private void UpdateSize(IAVLNode node)**

* מה עושה – הפונקציה מקבלתnode בעץ, ומעדכנת לו את הsize שלו ע"י סכימת גדלי הsize של הבן השמאלי והימני שלו, +1 (שכן צריך להוסיף גם אותו עצמו). כל עוד הnode לא מצביע לnull, נעלה במעלה העץ ונעדכן כך עד שנגיע לשורש.
* כיצד פועלת – הפונקציה מקבלת node בעץ, ופועלת בהנחה שכל הsize של הצמתים תחת הNode הזה מעודכנים. עבור הnode, היא מחשבת את הsize שלו ע"י סכימת גדלי הsize של שני בניו + 1 (מוסיפה אותו עצמו), וכך היא עולה עד לשורש, כך שבכל פעם הNode מעודכן להיות האב שלו, ומשיג את הsize שלי כפי שתיארנו בתחילת הפונקציה.
* סיבוכיות זמן הריצה –
* קריאה לפונקציית עזר -

**Private int Balance(IAVLNode node)**

* מה עושה –
* כיצד פועלת –
* סיבוכיות זמן הריצה –
* קריאה לפונקציית עזר –

**Private void rotateRight(IAVLNode node)**

* מה עושה –
* כיצד פועלת –
* סיבוכיות זמן הריצה –
* קריאה לפונקציית עזר -

**Private void rotateLeft(IAVLNode node)**

* מה עושה –
* כיצד פועלת –
* סיבוכיות זמן הריצה –
* קריאה לפונקציית עזר -

**Public int delete(int k)**

* מה עושה –
* כיצד פועלת –
* סיבוכיות זמן הריצה –
* קריאה לפונקציית עזר -

**Private int BalanceDelete(IAVLNode node)**

* מה עושה –
* כיצד פועלת –
* סיבוכיות זמן הריצה –
* קריאה לפונקציית עזר -

**Public String min()**

* מה עושה –
* כיצד פועלת –
* סיבוכיות זמן הריצה –
* קריאה לפונקציית עזר -

**Private IAVLNode minrec(IAVLNode node)**

* מה עושה –
* כיצד פועלת –
* סיבוכיות זמן הריצה –
* קריאה לפונקציית עזר -

**Public String max()**

* מה עושה –
* כיצד פועלת –
* סיבוכיות זמן הריצה –
* קריאה לפונקציית עזר -

**Private IAVLNode maxrec(IAVLNode node)**

* מה עושה –
* כיצד פועלת –
* סיבוכיות זמן הריצה –
* קריאה לפונקציית עזר -

**Public int[] keysToArray()**

* מה עושה –
* כיצד פועלת –
* סיבוכיות זמן הריצה –
* קריאה לפונקציית עזר -

**void KeysToArrayRec(int [] keys,IAVLNode node,int [] index)**

* מה עושה –
* כיצד פועלת –
* סיבוכיות זמן הריצה –
* קריאה לפונקציית עזר -

**public String[] infoToArray()**

* מה עושה –
* כיצד פועלת –
* סיבוכיות זמן הריצה –
* קריאה לפונקציית עזר -

**Private void infoToArrayRec(String [] infos,IAVLNode node,int [] index)**

* מה עושה –
* כיצד פועלת –
* סיבוכיות זמן הריצה –
* קריאה לפונקציית עזר -

**Public int size()**

* מה עושה – מחזירה את size העץ
* כיצד פועלת – מחזירה את הsize של שורש המופע.
* סיבוכיות זמן הריצה –
* קריאה לפונקציית עזר – שימוש בפונקציה getSize() של המחלקה AVLNode – אשר מחזירה את גודל הsize של Node.

**Private IAVLNode successor(IAVLNode node)**

* מה עושה –
* כיצד פועלת –
* סיבוכיות זמן הריצה –
* קריאה לפונקציית עזר -

**Private IAVLnode predecessor(IAVLNode node)**

* מה עושה –
* כיצד פועלת –
* סיבוכיות זמן הריצה –
* קריאה לפונקציית עזר -

**Public IAVLNode getRoot()**

* מה עושה – במידה ועץ המופע לא ריק, מחזירה את הnode שהוא שורש העץ.
* כיצד פועלת – הפונקציה קוראת לפונקציה **isEmpty**, ובעזרתה בודקת אם עץ המופע הוא עץ ריק – אם הוא ריק – נחזיר false, ואחרת נחזיר את הnode שהוא שורש העץ.
* סיבוכיות זמן הריצה –
* קריאה לפונקציית עזר -

**public AVLTree[] split(int x)**

* מה עושה – הפונקציה מקבלת מספר, אשר יהווה מפתח לאחד הצמתים בעץ, והפונקציה תפצל את העץ שהוא המופע הנוכחי של המחלקה, לשני עצים, כאשר אחד מהם הוא עץ בו כל המפתחות קטנים מx, ובשני כל המפתחות גדולים מx. הפונקציה שומרת על העצים תקינים.
* כיצד פועלת – נתחזק 4 עצי AVL, שניים שבסוף נחזיר (t1 מפתחות קטנים, t2 מפתחות גדולים), ושניים שימשו אותנו לתחזוקה שלהם (left למפתחות קטנים, right למפתחות גדולים). כעת נביט **בnode** שערך המפתח שלו הוא **x**. תחילה נביט בבניו – אם יש לו בן שמאלי, נקח את הבן השמאלי להיות השורש של העץ בעל המפתחות הקטנים אשר בסוף נחזיר (t1). כנל עבור הבן הימני והעץ בעל המפתחות הגדולים שבסוף נחזיר (t2). מן הנקודה הזאת נעלה במעלה העץ – **אם אנחנו בן ימני לאב שלנו**, נקח את הבן השמאלי של האב להיות השורש של העץ (left) שעוזר לנו לתחזוקה. ונבצע join עבור האב שלנו, העץ left שאנו מתחזקים, עם העץ t1 שנחזיר לבסוף. **אם אנחנו בן שמאלי לאב שלנו**, נקח את הבן הימני של האב להיות השורש של העץ (right) שעוזר לנו לתחזוקה. ונבצע join עבור האב שלנו, העץ right שאנו מתחזקים, עם העץ t2 שנחזיר בסוף.

נמשיך כך עד שנגיע לשורש.

**לבסוף, נחזיר מערך בגודל 2,** כאשר במקום ה0 יהיה העץ בעל המפתחות הקטנים, ובמקום ה1 העץ בעל המפתחות הגדולים, וכך נסיים.

* סיבוכיות זמן הריצה –
* קריאה לפונקציית עזר -

**Public int join(IAVLNode x, AVLTree t)**

* מה עושה – הפונקציה מקבלת node מסוג IAVLNode, ועץ מסוג AVLTree, ומאחדתם אותם עם המופע הנוכחי של המחלקה.
* כיצד פועלת –
* סיבוכיות זמן הריצה –
* קריאה לפונקציית עזר -

**Private IAVLNode findMax(AVLTree tree)**

* מה עושה – מחזירה את הnode בעל המפתח המקסימלי מבין המפתחות בעץ.
* כיצד פועלת – הפונקציה מתבוננת בשורש העץ, כל עוד הבן הימני שלו הוא node אמיתי (בודקים בעזרת הפונקציה isRealNode) אנו מביטים בבן הימני של הצומת, עד שנחזיר עד הבן הימני ביותר.
* סיבוכיות זמן הריצה –
* קריאה לפונקציית עזר - אין

**Private void updateMax(AVLTree tree)**

* מה עושה – הפונקציה מקבלת עץ AVL ומעדכנת את המצביע max שלו בעזרת פונקציית העזר findMax.
* כיצד פועלת – הפונקציה מעדכנת את השדה max של העץ, עי קריאה לפונקציה findMax
* סיבוכיות זמן הריצה –
* קריאה לפונקציית עזר – נעזרנו בפונקציית העזר findMax, עליה נרחיב במקום המתאים.

**Private IAVLNode findMin(AVLTree tree)**

* מה עושה – הפונקציה מחזירה את הnode בעל המפתח המינימלי מבין המפתחות בעץ.
* כיצד פועלת – הפונקציה מתבוננת בשורש העץ, כל עוד הבן השמאלי שלו הוא node אמיתי (בודקים בעזרת הפונקציה isRealNode) אנו מביטים בבן השמאלי של הצומת, עד שנחזיר עד הבן השמאלי ביותר.
* סיבוכיות זמן הריצה –
* קריאה לפונקציית עזר - אין

**Private void updateMin(AVLTree tree)**

* מה עושה – הפונקציה מקבלת עץ AVL ומעדכנת את המצביע min שלו בעזרת פונקציית העזר findMin.
* כיצד פועלת – הפונקציה מעדכנת את השדה min של העץ, עי קריאה לפונקציה findMin
* סיבוכיות זמן הריצה –
* קריאה לפונקציית עזר – נעזרנו בפונקציית העזר findMin, עליה נרחיב במקום המתאים.

**Public void treePrinter()**

* מה עושה –
* כיצד פועלת –
* סיבוכיות זמן הריצה –
* קריאה לפונקציית עזר -

**Public Boolean checkB()**

* מה עושה –
* כיצד פועלת –
* סיבוכיות זמן הריצה –
* קריאה לפונקציית עזר -

**Public Boolean checkButil(IAVLNode node)**

* מה עושה –
* כיצד פועלת –
* סיבוכיות זמן הריצה –
* קריאה לפונקציית עזר -

**Public Boolean checkSize()**

* מה עושה –
* כיצד פועלת –
* סיבוכיות זמן הריצה –
* קריאה לפונקציית עזר -

**Public Boolean checkMaxMin()**

* מה עושה –
* כיצד פועלת –
* סיבוכיות זמן הריצה –
* קריאה לפונקציית עזר -