**תיעוד מימוש עץ AVL – לירון כהן 207481268 ויובל מור 209011543**

**מתודות AVLTree**

**AVLTree()**

המתודה בונה עץ ריק, כלומר מאתחלת את שורש העץ להיות null ואת גודל העץ להיות 0.

סיבוכיות המתודה היא כיוון שהיא קובעת ערכים של שדות.

**Empty()**

המתודה בודקת האם שורש העץ הוא null. אם כן, מחזירה אמת ואחרת שקר.

סיבוכיות המתודה היא כיוון שהשוואת שורש העץ ל-null נעשית בזמן קבוע.

**getRank()**

המתודה מחזירה -1 אם העץ ריק. אחרת, מחזירה את דרגת שורש העץ.

סיבוכיות המתודה היא כיוון שהשוואת שורש העץ ל-null וקבלת הדרגה נעשית בזמן קבוע.

**Search(int k)**

המתודה מקבלת מספר שלם המייצג מפתח בעץ ומבצעת חיפוש בינארי על צמתי העץ.

בכל שלב, המתודה משווה את מפתח הצומת הנבדק לעומת המפתח שהתקבל כארגומנט.

נתחיל ממצביע לשורש, ונמשיך בחיפוש כל עוד לא הגענו לצומת שהיא null:

אם המפתחות שווים, מחזירה את ערך הצומת.

אם המפתח שהתקבל קטן ממפתח הצומת, מצביע הצומת יעבור לבן השמאלי.

אם המפתח שהתקבל גדול ממפתח הצומת, מצביע הצומת יעבור לבן הימני.

אם לא נמצא המפתח עד לשלב זה, המפתח אינו נמצא בעץ ומוחזר null.

סיבוכיות המתודה היא מכיוון שמדובר בעץ AVL (שהינו עץ חיפוש בינארי מאוזן), עומק העץ הוא ובמקרה הגרוע בכל איטרציה של לולאת ה-while לא מוצאים את המפתח ולכן יורדים רמה בעץ, עד להגעה לעלה וירטואלי.

**treePosition(int k) – פעולת עזר**

המתודה מקבלת מפתח להכנסה ומחזירה את צומת ההורה של המיקום המתאים להכנסה. אם המפתח קיים כבר בעץ, מחזירה את הצומת עם המפתח שהתקבל.

סיבוכיות המתודה היא מכיוון שמדובר בעץ AVL (שהינו עץ חיפוש בינארי מאוזן), עומק העץ הוא ובמקרה הגרוע בכל איטרציה של לולאת ה-while לא מוצאים את המפתח ולכן יורדים רמה בעץ, עד להגעה לעלה וירטואלי.

**insertBST(IAVLNode) – מתודת עזר**

המתודה מקבלת צומת ומכניסה אותה למקום המתאים בעץ לפי הגדרות עץ חיפוש בינארי.

המתודה משתמשת במתודה treePosition שסיבוכיותה .

לאחר מכן, המתודה בודקת האם מפתח הצומת להכנסה זהה למפתח שהתקבל מהפעולה, אם כן המתודה מחזירה -1.

אם לא, בודקת האם להכניס את הצומת כבן שמאלי או בן ימני, מעדכנת את ה-rank של הצומת החדשה ואת שדה ה-size של העץ, ולבסוף מחזירה 1.

מלבד הקריאה לפעולת העזר מתבצעות בדיקות תנאים ועדכוני שדות ולכן בסה"כ סיבוכיות המתודה היא .

**otherChild(AVLNode p, AVLNode c) – מתודת עזר**

המתודה מקבלת הורה ובן ומחזירה את הבן השני של ההורה.

סיבוכיות המתודה היא כיוון שמתבצעת בדיקת תנאי והחזרת שדות.

**rankDiff(AVLNode p, AVLNode c) – מתודת עזר**

המתודה מקבלת הורה ובן ומחזירה את הפרש ה-rank ביניהם.

סיבוכיות המתודה היא כיוון שמתבצעת פעולה אריתמטית והחזרת שדות.

**Promote(AVLNode n), Demote(AVLNode n) – מתודות עזר**

המתודות מקבלות ומוסיפות/מורידות 1 מה-rank שלה.

סיבוכיות המתודה היא כיוון שמתבצעת פעולה אריתמטית, עדכון והחזרת שדות.

**rightRotate(AVLNode z, AVLNode n), leftRotate(AVLNode z, AVLNode n) – מתודות עזר**

המתודות מקבלות הורה ובן (ביניהם הקשת שעלינו לסובב), מעדכנות את המצביעים הרלוונטיים ומחזירות 1.

סיבוכיות המתודות היא כיוון שמתבצעים עדכוני מצביעים ועדכוני שדות.

**rightLeftRotate(AVLNode z, AVLNode n), leftRightRotate(AVLNode z, AVLNode n) – מתודות עזר**

המתודות מקבלות הורה ובן (ביניהם הקשת הראשונה שעלינו לסובב), קוראות לפעולות הסיבוב המתאימות ומחזירות 2 (עבור שני הסיבובים).

סיבוכיות המתודות היא כיוון שמתבצעות קריאות לפעולות שסיבוכיותן .

**RebalanceInsert(AVLNode n) – מתודת עזר**

המתודה מקבלת צומת שהוכנס ומאזנת מחדש את העץ בהתאם למצב הצומת (אל מול ההורה והבנים שלו). המתודה קוראת לפעולות ה-promote וה-rotations אם יש צורך ומחזירה את מספר פעולות האיזון שנעשו.

סיבוכיות המתודה במקרה הגרוע היא . מלבד כאשר יש צורך ב-promote, כל הפעולות נעשות ב-. אם התבצע promote, ישנה אפשרות שבעיית האיזון "עלתה למעלה" רמה אחת בעץ ולכן התבצעה קריאה נוספת ל-rebalanceInsert. ראינו בכיתה שמספר פעולות ה-promote הוא לכל היותר גובה העץ, אשר כמעט מאוזן ולכן מדובר ב- בסך הכל.

**Insert(int k, String s)**

המתודה מקבלת מפתח וערך ויוצרת צומת חדש בהתאמה. אם העץ ריק, מכניסה את הצומת כשורש העץ ומחזירה 0 כיוון שלא בוצעו פעולות איזון. אם העץ לא ריק, היא מכניסה את הצומת לעץ לפי חוקי עץ חיפוש בינארי (באמצעות מתודת העזר insertBST). אם המפתח היה בעץ, מחזירה -1. אחרת, מאזנת מחדש את העץ (באמצעות מתודת העזר rebalanceInsert) ומחזירה את מספר פעולות האיזון שנעשו.

סיבוכיות המתודה היא כיוון ששתי פעולות העזר נקראות באופן טורי והן בסיבוכיות של . מלבד זאת מתבצעת יצירת צומת ובדיקות פשוטות שהן .

**Successor(AVLNode n) *– מתודת עזר***

*המתודה מקבלת צומת בעץ ומחזירה את הצומת העוקב שלו בהתאם לנלמד בכיתה.*

*סיבוכיות המתודה היא כיוון שבמקרה הגרוע המתודה צריכה "לעלות" את כל העץ כדי למצוא את הצומת העוקב, ומכיוון שמדובר בעץ* AVL *כמעט מאוזן, מדובר ב- רמות.*

**deleteBST(IAVLNode) – מתודת עזר**

המתודה מקבלת צומת ומוחקת אותה מהמקום המתאים בעץ לפי הגדרות עץ חיפוש בינארי. המתודה מחזירה את ההורה של הצומת שנמחק.

אם צריך, המתודה משתמשת במתודה successor שסיבוכיותה . בכל מקרה אחר המתודה משנה את המצביעים הרלוונטיים למחיקה ומקטינה את גודל העץ ב-1.

מלבד הקריאה לפעולת העזר מתבצעות בדיקות תנאים ועדכוני שדות ולכן בסה"כ סיבוכיות המתודה היא .

**RebalanceDelete(AVLNode n) – מתודת עזר**

המתודה מקבלת את ההורה של הצומת שנמחק ומאזנת מחדש את העץ בהתאם למצב הצומת (אל מול ההורה והבנים שלו). המתודה קוראת לפעולות ה-demote, promote וה-rotations אם יש צורך ומחזירה את מספר פעולות האיזון שנעשו.

סיבוכיות המתודה במקרה הגרוע היא . פעולות איזון טרמינליות נעשות בקריאה למתודות עזר שסיבוכיותן . אם בעיית האיזון "עלתה למעלה" רמה אחת בעץ והתבצעה קריאה נוספת ל-rebalanceDelete, מספר ה"עליות" בעץ הוא לכל היותר גובה העץ, אשר כמעט מאוזן ולכן מדובר ב- בסך הכל.

**Delete(int k)**

המתודה מקבלת מפתח למחיקה ומוצאת באמצעות treePosition את מיקום הצומת למחיקה. אם המפתח לא בעץ, מחזירה -1. אחרת, מוחקת את הצומת מהעץ לפי חוקי עץ חיפוש בינארי (באמצעות מתודת העזר deleteBST). לאחר מכן, המתודה מאזנת מחדש את העץ (באמצעות שליחת ההורה למתודת העזר rebalanceDelete) ומחזירה את מספר פעולות האיזון שנעשו.

נציין שאם הצומת למחיקה היא שורש העץ, המתודה מוחקת את השורש מהעץ כרגיל אך שולחת ל-rebalanceDelete את השורש ולא את ההורה שלו (שהוא null) כדי לאזן את שורש העץ אם צריך.

סיבוכיות המתודה היא כיוון ששלוש פעולות העזר נקראות באופן טורי והן בסיבוכיות של . מלבד זאת מתבצעות בדיקות פשוטות שהן .

**Min()**

המתודה מתחילה עם מצביע לשורש העץ ויורדת עם המצביע לבן השמאלי כל עוד הבן השמאלי קיים (והוא לא עלה וירטואלי, כדי להגיע לעלה האמיתי הנדרש). המתודה מחזירה את ערך העלה השמאלי ביותר בעץ.

סיבוכיות המתודה היא מכיוון שמדובר בעץ AVL (שהינו עץ חיפוש בינארי מאוזן), עומק העץ הוא וכדי להגיע לעלה השמאלי ביותר בעץ (בעל הערך המינימלי) נצטרך לרדת בלולאה פעמים.

**Max()**

המתודה מתחילה עם מצביע לשורש העץ ויורדת עם המצביע לבן הימני כל עוד הבן הימני קיים (והוא לא עלה וירטואלי, כדי להגיע לעלה האמיתי הנדרש). המתודה מחזירה את ערך העלה הימני ביותר בעץ.

סיבוכיות המתודה היא מכיוון שמדובר בעץ AVL (שהינו עץ חיפוש בינארי מאוזן), עומק העץ הוא וכדי להגיע לעלה הימני ביותר בעץ (בעל הערך המקסימלי) נצטרך לרדת בלולאה פעמים.

**nodesToArray() – מתודת עזר**

המתודה מחזירה מערך צמתים שמכיל את צמתי העץ מסודרים לפי מפתחות בסדר עולה.

המתודה מגדירה מחסנית ששומרת את כל הצמתים שהמצביע עבר בהם.

כל עוד המצביע לא הגיע לעלה וירטואלי או שיש עוד צמתים במחסנית, נכניס את הצומת הנוכחי למחסנית ונזוז לבן השמאלי. כשנגיע לעלה וירטואלי נכניס למערך את הצומת העליון במחסנית (המינימום באותו הרגע) ונזוז לבן הימני של הצומת שהכנסנו למערך.

סיבוכיות המתודה היא מכיוון שנבקר בכל צומת לכל היותר 3 פעמים (בדרך לבן השמאלי, בחזרה מהבן השמאלי אל הימני ובחזרה מהימני להורה), לכן נקבל ביקורים. פעולות המחסנית, כמו גם בדיקות והחזרת שדות נעשות ב- ולכן נקבל סיבוכיות לינארית.

**KeysToArray()**

המתודה מחזירה מערך שלמים שמכיל את מפתחות העץ בסדר עולה.

המתודה משתמשת במתודת העזר nodesToArray() ומקבלת ממנה מערך צמתים מסודר לפי מפתחות בסדר עולה. המתודה עוברת על מערך הצמתים ומכניסה למערך שלמים את מפתחות הצמתים לפי הסדר.

סיבוכיות המתודה היא מכיוון שמתודת העזר פועלת ב- ומעבר על המערך נעשה גם כן ב-. שאר הפעולות נעשות בזמן קבוע.

**infoToArray()**

המתודה מחזירה מערך מחרוזות שמכיל את ערכי העץ מסודרים לפי מפתחות בסדר עולה.

המתודה משתמשת במתודת העזר nodesToArray() ומקבלת ממנה מערך צמתים מסודר לפי מפתחות בסדר עולה. המתודה עוברת על מערך הצמתים ומכניסה למערך מחרוזות את ערכי הצמתים לפי הסדר.

סיבוכיות המתודה היא מכיוון שמתודת העזר פועלת ב- ומעבר על המערך נעשה גם כן ב-. שאר הפעולות נעשות בזמן קבוע.

**Size()**

המתודה מחזירה את ערך השדה size של העץ.

סיבוכיות המתודה היא כיוון שהיא מחזירה ערך של שדה.

**Split(int x)**

* לקרוא את הפוסט בפורום בכותרת " שינוי חתימת המחלקה AVLNode" ואת "פונקציית split"

**findRankEquiv(AVLTree tree, int rank) – מתודת עזר**

המתודה מקבלת עץ ומספר המייצג דרגה ומחזירה את הצומת על השדרה השמאלית של עץ שדרגתה היא המקסימלית שקטנה או שווה ל-rank (עבור המתודה join).

סיבוכיות המתודה היא מכיוון שמדובר בעץ AVL (שהינו עץ חיפוש בינארי מאוזן), עומק העץ הוא ובמקרה הגרוע נצטרך לרשת עד לעלה השמאלי ביותר בעץ, כלומר הלולאה תרוץ פעמים.

**Join(IAVLNode x, AVLtree t)**

המתודה מקבלת עץ וצומת, ובהתאם לאלגוריתם שהצגנו בכיתה מאחדת את העץ הנוכחי לעץ הנתון כארגומנט בהוספת הצומת הנתונה.

אם אחד העצים ריק או ששניהם ריק, קוראת למתודה insert ומעדכנת את שורש העץ הנוכחי במידת הצורך.

אם שניהם לא ריקים, מבצעת השוואת מפתחות ודרגות כדי לקבוע את תצורת האיחוד, מוצאת בעץ בעל הדרגה הגבוהה את הצומת המקביל (מבחינת דרגה) לשורש העץ בעל הדרגה הנמוכה (באמצעות מתודת העזר findRankEquiv), משנה את המצביעים הרלוונטיים ומעדכנת את שורש העץ במידת הצורך.

לאחר מכן, המתודה קוראת למתודת העזר rebalanceInsert על מנת לאזן את העץ במידת הצורך.

סיבוכיות המתודה היא מכיוון שמתודות העזר כולן בסיבוכיות , הן פועלות בצורה טורית וביניהן מתרחשות בדיקות ושינויי מצביעים הנעשים ב-.

* לקרוא בפורום
* סרטון של שירי מהשעת קבלה?

**getRoot()**

המתודה מחזירה את ערך השדה root של העץ.

סיבוכיות המתודה היא כיוון שהיא מחזירה ערך של שדה.

**מתודות AVLNode**

* לזכור להשתמש ב-casting אם צריך

**AVLNode(int key, String info)**

המתודה בונה עצם מסוג AVLNode, מאתחלת את המפתח והערך שלו למפתח והערך שהתקבלו כארגומנטים.

בנוסף, אם המפתח שהתקבל אינו -1 (כלומר צומת אמיתי), משנה את ערך ה-isReal לאמת (כיוון שבאופן דיפולטי משתנים בוליאניים מאותחלים לשקר) ומגדירה את שני הבנים של הצומת לצמתים וירטואליים חדשים.

אם המפתח שהתקבל הוא -1, מגדירה את דרגתו כ-1.

סיבוכיות המתודה היא כיוון שהיא מבצעת השמות ובדיקות שנעשות בזמן קבוע.

**getKey()**

המתודה מחזירה את המפתח של הצומת.

סיבוכיות המתודה היא כיוון שהיא מחזירה ערך של שדה.

**getValue()**

המתודה מחזירה את הערך של הצומת.

סיבוכיות המתודה היא כיוון שהיא מחזירה ערך של שדה.

**setLeft(IAVLNode node)**

המתודה מגדירה את שדה הבן השמאלי של הצומת לצומת שהתקבל כארגומנט.

סיבוכיות המתודה היא כיוון שהיא קובעת ערך של שדה.

**getLeft()**

המתודה מחזירה את צומת הבן השמאלי של הצומת.

סיבוכיות המתודה היא כיוון שהיא מחזירה ערך של שדה.

**setRight(IAVLNode node)**

המתודה מגדירה את שדה הבן הימני של הצומת לצומת שהתקבל כארגומנט.

סיבוכיות המתודה היא כיוון שהיא קובעת ערך של שדה.

**getRight()**

המתודה מחזירה את צומת הבן הימני של הצומת.

סיבוכיות המתודה היא כיוון שהיא מחזירה ערך של שדה.

**setParent(IAVLNode node)**

המתודה מגדירה את שדה ההורה של הצומת לצומת שהתקבל כארגומנט.

סיבוכיות המתודה היא כיוון שהיא קובעת ערך של שדה.

**getParent()**

המתודה מחזירה את צומת ההורה של הצומת.

סיבוכיות המתודה היא כיוון שהיא מחזירה ערך של שדה.

**isRealNode()**

המתודה מחזירה אמת אם הצומת אמיתית ושקר אם היא עלה וירטואלי.

סיבוכיות המתודה היא כיוון שהיא מחזירה ערך של שדה.

**setHeight(int height)**

המתודה מגדירה את שדה הגובה של הצומת לערך שהתקבל כארגומנט.

סיבוכיות המתודה היא כיוון שהיא קובעת ערך של שדה.

**getHeight()**

המתודה מחזירה את גובה הצומת.

סיבוכיות המתודה היא כיוון שהיא מחזירה ערך של שדה.

**setRank(int rank)**

המתודה מגדירה את שדה ה-rank של הצומת לערך שהתקבל כארגומנט.

סיבוכיות המתודה היא כיוון שהיא קובעת ערך של שדה.

**getRank()**

המתודה מחזירה את ה-rank של הצומת.

סיבוכיות המתודה היא כיוון שהיא מחזירה ערך של שדה.

**שאלות נוספות**

* לקרוא את שרשור finger search כיצד למקם איברים חדשים