timore – 308145309 – תימור איזנמן
offekgil – 308315092 – אופק גיל

מחלקת העל – WAVLTREE

שדות

private final	WAVLNode EXT	מצביע לצומת וירטואלי, אקסטרני (לשימוש פנימי בלבד). מאותחל סופית. מחוסר הורה וילדים, ובעל דרגה 1-.
Private	WAVLNode root	מצביע לשורש העץ
Private	WAVLNode max	מצביע לאיבר עם המפתח הקטן ביותר בעץ
Private	WAVLNode min	מצביע לאיבר עם המפתח הגדול ביותר בעץ

מתודות

בנאי ראשון

- קלט: אין.
- יוצר עץ ריק חדש, כאשר לאחר האתחול השורש הוא צומת אקסטרני (לשימוש עתידי).
- סיבוכיות: $O(1)$.

בנאי שני

- קלט: מפתח מסוג int, מידע מסוג string.
- מאתחל את השורש על ידי יצירת עצם מסוג WAVLNode כאשר:
  - מפתח מתקבל מהקלט.
  - מידע מתקבל מהקלט.
  - הורה – EXT, הצומת האקסטרני (לשימוש עתידי).
- משנה את המצביעים min,max לשורש (מהיותו היחידי בעץ).
- סיבוכיות: $O(1)$.

Empty

- קלט: אין.
- פלט: ערך בוליאני, האם העץ ריק או לא.
- שיטת פעולה:
  - במידה והשורש מצביע לצומת האקסטרני, EXT, אזי העץ ריק ומוחזר הערך true, על פי לוגיקת הבנאי הראשון.
  - אחרת, מוחזר false.
- סיבוכיות: $O(1)$.

Search

- קלט: מפתח מסוג int.
- פלט: מחרוזת המייצגת את המידע של הצומת עם המפתח k.
- שיטת פעולה:
  - במידה והעץ ריק, על ידי שימוש במתודה empty, מוחזר null.
  - במידה ולא ריק, מבצעת חיפוש בינארי עד אשר מגיעה לצומת עם מפתח הקלט, במידה ומצא – מחזיר את המידע ממנו.
  - אם הגענו עד לעלה חיצוני, EXT, סימן שהמפתח לא נמצא בעץ, ועל כן יוחזר null.
- סיבוכיות: $O(logn)$.

Min

- קלט: אין.
- פלט: מחרוזת המייצגת את המידע של המידע של הצומת עם המפתח המינימלי.
- שיטת פעולה:
  - במידה והעץ לא ריק, על ידי שימוש במתודה empty, פונה לצומת בעל המפתח המינימלי בעזרת השדה min, ומחזיר את המידע ממנו.
  - אחרת, מחזיר null.
- סיבוכיות: $O(1)$.

Max

- קלט: אין.
- פלט: מחרוזת המייצגת את המידע של המידע של הצומת עם המפתח המקסימלי.
- שיטת פעולה:
  - במידה והעץ לא ריק, על ידי שימוש במתודה empty, פונה לצומת בעל המפתח המקסימלי בעזרת השדה max, ומחזיר את המידע ממנו.
  - אחרת, מחזיר null.
- סיבוכיות: $O(1)$.

Size

- קלט: אין.
- פלט: מספר המייצג את כמות הצמתים בעץ = גודל העץ.
- שיטת פעולה:
  - מחזירה את הערך המספרי של גודל תת העץ (השדה subTreeSize) של העצם מסוג WAVLNode של השורש.
    - הפעולה מוגדרת היטב, מכיוון שביצירת הצומת האקסטרני, EXT, גודל העץ שלו מוגדר להיות 0.
- סיבוכיות: $O(1)$.

getRoot

- קלט: אין.
- פלט: WAVLNode המייצג את שורש העץ.
- שיטת פעולה:
  - בודקת האם השורש מצביע לצומת האקסטרני, EXT, ומחזירה null אם כן.

timore – 308145309 – תימור איזנמן
offekgil – 308315092 – אופק גיל

	▪ יש לשים לב כי בעץ ריק השורש הוא הצומת האקסטרני. ○ אחרת, מחזירה את המצביע root באמצעות החזרת השדה התואם. • סיבוכיות: $O(1)$.
Select	• קלט: מספר של פיו נחפש את האיבר ה-i בגודלו בעץ. • פלט: מחרוזת המייצגת את המידע של הצומת ה-i בגודלו בעץ. • שיטת פעולה: ○ בדיקות התחלה: ▪ אם העץ ריק, על ידי המתודה empty, או שאינדקס הקלט גדול מכמות הצמתים בעץ, בעזרת המתודה size, מוחזר הערך null. ▪ אם מחפשים את האיבר הראשון, משמע הקטן ביותר, מחזיר את המידע מהצומת המינימלי, בעזרת המצביע min. ▪ אם מחפשים את האיבר האחרון, הבדיקה מתבצעת בעזרת המתודה size, מוחזר המידע מהצומת המקסימלי, בעזרת המצביע max. ○ במידה והמתודה לא נעצרה עד כה החיפוש נעשה בשיטה הבאה: ▪ בכל איטרציה, נתייחס לתת העץ שאנו נמצאים בו, כאילו הוא עץ חדש, ללא זיכרון למה היה בעבר. ▪ מתחילים מהשורש. ▪ אם גודל תת העץ השמאלי, משמע כמות המפתחות שקטנים ממני, ועוד אחד (אני) הוא האינדקס אותו חיפשנו, אזי שאני הצומת ה-i בגודלו בעץ. ▪ אחרת, עלינו להמשיך לחפש על פי: • אם האינדקס גדול או שווה מגודל תת העץ השמאלי, אזי סימן שהוא נמצא בתת העץ השמאלי, ונחזור על הפעולה עד אשר נמצא אותו. • אחרת, זה סימן שהוא נמצא בתת העץ הימני, אך הפעם, מכיוון ואנו מסתכלים על תת העץ הזה ללא זיכרון לעבר, עלינו למצוא כעת את האיבר ה-i פחות גודל העץ ועוד אחד, מכיוון ששללנו את יתר האיברים כבר, וכעת נחזור על הפעולה עד אשר נמצא אותו. ○ לבסוף, נחזיר את המידע של הצומת עם המפתח בגודל ה-i. • סיבוכיות: $O(logn)$.
Successor	• קלט: עצם מסוג WAVLNode. • פלט: WAVLNode המייצג את הצומת עם המפתח הבא בגודלו בעץ, אחרי צומת הקלט. • שיטת פעולה: ○ במידה והעצם המתקבל הוא המקסימלי בעץ, על ידי שימוש במצביע, אזי אין מפתח גדול ממנו, ויוחזר null. ○ אחרת, למדנו כי על מנת למצוא את המפתח הבא בגודלו עלינו לרדת לבן הימני, ואז שמאלה ברציפות ככל האפשר, והאחרון הוא היורש. או מנגד, במידה ואין בן ימני, כל עוד זה בן הימני של אביו, עלינו להמשיך לטפס עד אשר נמצא אב שאני אהיה הבן השמאלי שלו, והאחרון יהיה היורש. ○ לבסוף, מחזירה את היורש שמצאנו. • סיבוכיות: $O(logn)$.
Predecessor גרסא רגילה	• קלט: עצם מסוג WAVLNode. • פלט: WAVLNode המייצג את הצומת עם המפתח הקודם בגודלו בעץ, לפני צומת הקלט. • שיטת פעולה: ○ קריאה לפונקציה predecessor בגסראתי הבולאנית, עם הnode שנקלט והערך הבוליאני false. • סיבוכיות: $O(logn)$.
Predecessor **גרסא בולאנית**	• קלט: עצם מסוג WAVLNode, ערך בוליאני המייצג האם קראנו למתודה מתוך deleteBinary או לא. • פלט: WAVLNode המייצג את הצומת עם המפתח הקודם בגודלו בעץ, לפני צומת הקלט. • שיטת פעולה: ○ במידה והעצם המתקבל הוא המינימלי בעץ, על ידי שימוש במצביע, אזי אין מפתח קטן ממנו, ויוחזר null. ○ אחרת, למדנו כי על מנת למצוא את המפתח הקודם בגודלו עלינו לרדת לבן השמאלי, ואז ימינה ברציפות ככל האפשר, והאחרון הוא הקודם בגודלו. לאורך חיפוש זה, מתבצעת בדיקה האם המתודה נקראה מתוך deleteBinary. במידה וכן, עלינו להוריד את גודל תת העץ עבור צומת, מכיוון שאנו עתידים למחוק את הקודם בגודלו לאחר ההחלפה. או מנגד, במידה ואין בן שמאלי, כל עוד זה בן השמאלי של אביו, עלינו להמשיך לטפס עד אשר נמצא אב שמגיעים אליו מצד שמאל, והוא יהיה הקודם. ○ לבסוף, מחזירה את המפתח של הקודם שמצאנו. • סיבוכיות: $O(logn)$.
keysToArray	• קלט: אין. • פלט: מערך מספרים המכיל את כל המפתחות הקיימים בעץ. • שיטת פעולה: ○ מקבלת את הערך של גודל העץ מהפונקציה size(). ○ במידה וגודל העץ הוא 0, מוחזר מערך ריק.

timore – 308145309 – תימור איזנמן
offekgil – 308315092 – אופק גיל

	○ אחרת, נוצר מערך חדש כגודל העץ שקיבלנו קודם. ○ כעת, מתבצעת לולאת for באורך גודל העץ של פעולות successor החל מהצומת המינימלי אשר מתקבל מהמצביע, ומכניסה למערך את המפתח של כל צומת. ○ לבסוף מוחזר המערך המלא. • סיבוכיות: $O(n)$.
infoToArray	• קלט: אין. • פלט: מערך מחרוזות המכיל את כל המידע הקיים בעץ. • שיטת פעולה: ○ מקבלת את הערך של גודל העץ מהפונקציה ()size. ○ במידה וגודל העץ הוא 0, מוחזר מערך ריק. ○ אחרת, נוצר מערך חדש כגודל העץ שקיבלנו קודם. ○ כעת, מתבצעת לולאת for באורך גודל העץ של פעולות successor החל מהצומת המינימלי אשר מתקבל מהמצביע, ומכניסה למערך את המידע של כל צומת. ○ לבסוף מוחזר המערך המלא. • סיבוכיות: $O(n)$.
Rebalance	• קלט: WAVLNode שהחל ממנו תתחיל הבדיקה לצורך באיזון. • פלט: מספר המייצג את כמות פעולות האיזון שנעשו. • שיטת פעולה: ○ המתודה תבדוק עבור כל צומת אב החל מהקלט ועד לשורש את הצורך באיזון לפי הפרמטר המוחזר מהמתודה deleteCases: ▪ אם מוחזר 0: אין צורך לבצע אף פעולת איזון וממשיכים. ▪ אם מוחזר 1: מבצעים פעולת demotion לצומת וממשיכים. ▪ אחרת, אנו צריכים לעקוב אחרי איזה בן לא מהווה בעיה מבחינת הפרשי דרגות, ועל כן נשמור אותו בצד, בנוסף אליו, נשמור במשתנה sonDir מאיזה צד הוא הבן, ובמשתנה direction את הכיוון ההפוך (לטובת שימוש עתידי ברוטציות). ▪ אם מוחזר 2: מבצעים פעולת double demotion לצומת ולבן שלו ששמרנו בשלב הקודם, וממשיכים. ▪ אם מוחזר 3: מדובר בפעולת רוטציה אחת, ועל כן משתמשים במתודה singleRotation על הבן ששמרנו עם הכיוון ההפוך מהצד בו הוא נמצא, וממשיכים. ▪ אם מוחזר 4: מדובר בפעולת רוטציה כפולה, נפעיל את המתודה singleRotation פעמיים על פי הבן ששמרנו: • אם הוא ימני – נפעיל על בנו השמאלי של הבן, קודם רוטציה ימינה ואז שמאלה, וממשיכים. • אחרת, נפעיל על בנו הימני של הבן, קודם רוטציה שמאלה ואז ימינה, וממשיכים. ○ לבסוף, מחזירים את כמות פעולות האיזון שנעשו. • סיבוכיות: $O(logn)$.
singleRotation	• קלט: WAVLNode המייצג את הצומת שיהפוך להיות השורש של תת העץ לאחר הרוטציה, מחרוזת המייצגת את הכיוון אליו נעשה את הרוטציה. • פלט: WAVLNode המייצג את הצומת שהיה השורש של תת העץ לפני הרוטציה. • שיטת פעולה: ○ בתחילה, אנו שומרים 3 משתנים: הצומת שקיבלנו, אביו וגודל תת העץ שהוא שורשו. ○ כעת, אנו רוצים לחבר בין האבא של השורש המקורי, לשורש החדש. נעשית בדיקה האם המקורי הוא שורש העץ הגדול, ואם כן אזי שאין לו אבא, אז צריך לשנות את מצביע השורש של המחלקה להצביע, אחרת נעשית בדיקה האם ההחלפה נעשית מימין או משמאל וההשמה בהתאם. ○ כעת, על סמך הכיוון שקיבלנו בקלט, מתבצעת סדרת פעולות של חיבור השורש הישן כבן של השורש החדש, ושל הבן הקודם של השורש החדש, לאביו החדש, שהוא השורש המקורי. ○ כעת, מתבצעת בדיקה האם השורש המקורי הוא כעת עלה. אם כן, הוא מקבל ערכי גודל תת עץ ודרגה של עלה. אחרת, מוריד את דרגתו ב-1. ○ לבסוף, מחזיר את השורש המקורי. • סיבוכיות: $O(1)$.
Insert	• קלט: int המייצג את המפתח ומחרוזת המייצג את המידע אשר רוצים להכניס לעץ יחד בתור איבר חדש. • פלט: מספר פעולות האיזון שהתרחשו בזמן הכנסת האיבר החדש לעץ. • שיטת פעולה: ○ ראשית נבדקת האפשרות שהעץ ריק בעזרת המתודה ()empty – אם כן, מייצרים WAVLNode חדש עם הערכים שנקלטו, ומשימים אותו בתור שדה השורש של העץ, כמו גם האיבר המקסימאלי ומינימאלי. מחזירים 0ת שכן לא בוצעו פעולות איזון. ○ במידה והעץ לא היה ריק, אנו עושים חיפוש בינארי על העץ, על מנת למצוא את המיקום בו צריך להכניס את האיבר החדש (אנו יודעים זאת על ידי הגעה לעלה אקסטרני במקום הנדרש), ולשם מחברים איבר חדש עם הערכים שנקלטו (חיבור: הגדרה לאבא את האיבר כבן בכיוון הרלוונטי, ולאיבר את האבא בתור הורה) . במידה ומגלים בחיפוש כי כבר קיים איבר עם המפתח שנקלט, מוחזר 1-. ○ במידה ואכן מצאנו כי ניתן להכניס את האיבר, עוברים בלולאה על כל הוריו של המקום הרצוי, ומגדילים להם את גודל הsubTreeSize. ○ במידה וההורה היה אונארי, אין צורך בפעולות איזון, ומוחזר 0.

timore – 308145309 – תימור איזנמן
offekgil – 308315092 – אופק גיל

	○ נשמר משתנה countBalance שיספור את פעולות האיזון שמתבצעות. במידה וההורה היה עלה, נגדיל את הrank של ההורה ב-1, ונבדוק את שלושת מצבי האיזון עבור ההורה של ההורה (parent) ועבור כל האיברים ממנו ועד השורש (או עד שמגיעים לרוטציה): ▪ מקרה 1: אם מגלים כי ההפרש הדרגות בין parent לבין בנו האחר הינו 1, ניתן להעלות בדרגה את parent, ולהעלות בבדיקה לאיבר הבא. ▪ מקרה 2: אם הפרשו של parent מבנו האחר הינו 2, נבדוק את הפרש הדרגות של בנו המקורי (זה שהעלנו במקור את דרגתו) מהנכד של parent אשר פונה "פנימה" - לכיוון בו נמצא parent. אם ההפרש הינו 2, מדובר במקרה 2, מבצעים רוטציה בודדת (אשר כוללת גם 2 שינויי דרגה), ומחזירים את סך פעולות האיזון. ▪ מקרה 3: אם ההפרש של הבן המקורי מהנכד שפונה "פנימה" הינו 1, יש צורך ברוטציה כפולה (שכוללת 4 שינויי דרגה) ומחזירים את סך פעולות האיזון. ▪ כל עוד לא מגיעים לפעולת רוטציה, או לשורש, ממשיכים בבדיקה זו עבור ההורה של ההורה הנוכחי. ○ לבסוף – אם לא בוצעו כלל פעולות רוטציה, מחזירים את המשתנה שספר את מספר פעולות האיזון (אשר במקרה זה הינן רק פעולות העלאה בדרגה). • סיבוכיות: O(logn)
Delete	• קלט: מספר המייצג את המפתח של הצומת אותו עלינו למחוק. • פלט: מספר המייצג את כמות פעולות האיזון שביצענו. • שיטת פעולה: ○ תחילה, נבדק האם העץ ריק בשימוש המתודה empty, או האם המפתח לא קיים בעץ בשימוש המתודה search. אם אחד התנאים מתקיים אזי נחזיר 1-. ○ לאחר מכן, מתבצעת בדיקה האם הצומת היחידי בעץ הוא השורש, והוא גם המפתח הנדרש למחיקה, אם כן אנו מוחקים אותו על ידי הצבעה לצומת האקסטרני כעל השורש, ומחזירים כי ביצענו 0 פעולות איזון. ○ כעת, אם לא נעצרנו עדיין, אנו מבצעים חיפוש בינארי עד אשר נמצא את הצומת הדרוש למחיקה, על פי המפתח. ○ כאשר מצאנו את הצומת, אנו בודקים אם הוא הצומת המינימלי או המקסימלי בעץ. במידה וכן, אנו מעדכנים את המצביע המתאים לצומת הבא/הקודם. ○ כעת, על פי סוג הצומת עלה/אונארי/בינארי, הוא נשלח למתודת המחיקה המתאימה. ○ לאחר מכן, מבצעים פעולות rebalance במידת הצורך. ○ לבסוף, מחזירים את כמות פעולות האיזון הכוללות שעשינו. • סיבוכיות: $O(logn)$.
deleteLeaf	• קלט: WAVLNode המייצג את הצומת שעלינו למחוק. • פלט: מספר המייצג את כמות פעולות האיזון שביצענו. • שיטת פעולה: ○ תחילה, אנו בודקים האם האב של הצומת הנמחק הוא אונארי, אם כן, סימן שיהפוך להיות עלה, ועל כן מקבל את השמה של כל הערכים המתאימים לעלה, ומחזיר 1, מכיוון ששינינו את דרגתו. ○ אחרת, על סמך בדיקה של איזה צד הצומת הנמחק הוא בן של אביו, מתבצעת פעולה המחיקה. ○ לאחר מחיקת העלה, מתקיימת בדיקה האם הבן השני של האבא הוא עלה, אם כן סימן שצריך רק לעשות בדיקות rebalance ולהחזיר את כמות פעולות האיזון שנעשו. ○ אחרת, יהיה צורך ברוטציה, ולאחר מכן לבצע בדיקות rebalance, ולהחזיר את כמות פעולות האיזון שנעשו + 3, מכיוון שביצענו ברוטציה פעולת promotion, rotation, demotion. • סיבוכיות: $O(logn)$.
deleteUnary	• קלט: WAVLNode המייצג את הצומת שעלינו למחוק. • פלט: WAVLNode המייצג את אביו של הנמחק. • שיטת פעולה: ○ המתודה למעשה מקפיצה את האבא של הנמחק להכיר את הבן של הנמחק. ○ בתחילה אנו מגלים מאיזה צד הצומת הנמחק הוא אונארי, ועל פי זה מבצעים את יתר הפעולות. ○ נעשית בדיקה האם האב הוא עלה אקסטרני, ואם כן זה אומר שהנמחק הוא השורש, לכן צריך להשים את הבן של הנמחק להיות השורש. ○ אם הוא לא השורש, אז על פי איזה צד הצומת הנמחק הוא בן של אביו, מתבצעת ההתאמה בין אב הצומת לילד הצומת. ○ לבסוף, מוחקים את הצומת, על ידי השמת אבא וילדים null, ומחזירים את צומת האב. • סיבוכיות: $O(1)$.
deleteBinary	• קלט: WAVLNode המייצג את הצומת שעלינו למחוק. • פלט: מספר המייצג את כמות פעולות האיזון שביצענו. • שיטת פעולה: ○ תחילה, אנו שומרים על מספר משתנים: ▪ צומת ה-predecessor על ידי קריאה למתודה עם הערך הבוליאני true. ▪ מפתח ה-predecessor. ▪ מידע ה-predecessor. ○ כעת, אנו מבצעים פעולות מחיקה של ה-predecessor בהתאם להם הוא עלה או צומת אונארי. ○ לבסוף, מבצעים החלפת ערכים של הצומת שעלינו למחוק, על ידי החלפת המפתח והמידע שלו בזה של ה-predecessor שלו, ובכך למעשה מבצעים את מחיקתו. ○ לבסוף מחזירים את כמות פעולות האיזון שעשינו.

timore – 308145309 – תימור איזנמן
offekgil – 308315092 – אופק גיל

	• סיבוכיות: $O(logn)$.
deleteCases	• קלט: WAVLNode המייצג את הצומת עליו אנו מבצעים את הבדיקה איזה תצורת איזון נדרשת לאחר המחיקה שביצענו. • פלט: מספר המייצג את סוג תצורת המחיקה שעלינו לעשות. • שיטת פעולה: ○ המתודה למעשה מחזירה איזה מבין ארבעת המקרים האפשריים של הפרת חוקי עץ WAVL לאחר מחיקה, אם בכלל יש הפרה, קיימת עבור הצומת אותו קיבלנו. ○ תחילה, נבדקת האפשרות שאין הפרה כלל של החוקים, והמתודה תחזיר 0 המייצג שאין בעיה. ○ לאחר מכן, בהתאם לאיזה צד ההפרה הקורית מתבצעות הבדיקות הנוספות. ○ אם ההפרש לבן השני הוא 2, אזי ניתן לבצע פעולת demotion בלבד, לכן נחזיר 1. ○ אם ההפרש לבן השני הוא 1, תחילה נשמור את הבן השני במשתנה ואז: ▪ אם ההפרש מהבן לשני בניו הוא 2, אזי double demotion יהיה מספיק, ויוחזר הערך 2. ▪ אם ההפרש מהבן ה"חיצוני" הוא 1, אזי נצטרך לבצע רוטציה, ויוחזר הערך 3. ▪ אחרת, נצטרך לבצע רוטציה כפולה, ויוחזר הערך 4. • סיבוכיות: $O(1)$.

timore – 308145309 – תימור איזנמן
offekgil – 308315092 – אופק גיל

מחלקה מקוננת – WAVLNode

שדות			
	private	int key	מכיל את מפתח ה-Node, כלומר הערך המספרי החח"ע אשר מעיד (וקובע) את מיקום האיבר בעץ, ולפיו נבצע את המתודות השונות של עץ הWAVL. ערך זה מאותחל ע"י קלט המשתמש, בעת יצירת איבר חדש.
	Private	String value	מכיל את הערך שהמשתמש הגדיר עבור הnode, בעת אתחולו. ערך זה אינו חח"ע.
	Public	int rank	קובע את דרגתו של הnode, בהתאם לכללי עץ WAVL, כלומר השדה תמיד יהיה בין 0 לבין פעמיים גובה העץ. בעת הכנסת איבר חדש לעץ, השדה של האיבר החדש מאותחל להיות 0, שכן הוא יהיה עלה (אשר דרגתו תמיד 0). שדה זה משמש את פעולות ההכנסה ומחיקה של העץ, על מנת לקבוע את הסיטואציה, וכן השדה מעודכן על פי הסיטואציה המתאימה.
	Public	WAVLNode parent	מפנה אל הnode אשר מהווה הורה של הnode הנוכחי. השדה מאותחל באמצעות עם בניית האיבר, ע"י קלט רלוונטי. נשים לב כי ההורה של שורש העץ הינו null.
	Public	WAVLNode left	מפנה אל בנו השמאלי של הnode (אשר גם הוא node). בזמן בניית הnode, השדה מאותחל להיות עלה אקסטרני EXT, שכן האיבר המדובר הינו עלה.
	Public	WAVLNode right	בדומה לשדה left, אך מצביע על הבן הימני.
	Public	int subTreeSize	שומר את גודל תת העץ (מספר האיברים) הקיים מתחת לאיבר הספציפי, כולל הוא עצמו. מתוחזק בזמן פעולות ההכנסה ומחיקה של העץ.

מתודות	
בנאי	• קלט: מפתח מסוג int, ערך מסוג string, איבר מסוג WAVLNode • שיטת פעולה: הבנאי נקרא בכל פעם שמאותחל node חדש. המפתח והערך מושמים בתור מפתח וער האיבר. WAVLNode parent מושם בתור ההפניה להורה האיבר. כמו כן, הrank של האיבר מאותחל להיות 0 (בהתאם לכך שהינו עלה), subTreeSize מאותחל להיות 1 ושדות הבנים מאותחלים להיות EXT. נשים לב כי עבור עלה שהינו EXT ישנו אתחול מיוחד (המזוהה באמצעותה כנסת מפתח 1-.) • סיבוכיות: O(1).
gets	• לכל שדה ישנה מתודת get, אשר מחזירה את ערך השדה בסיבוכיות O(1). נשים לב כי תחזוק השגות באופן חוקי קורה במהלך ביצוע מתודות WAVLTree. • קלט: אין • פלט: השדה הרלוונטי לאותה מתודה • שיטת פעולה: מחזיר את השדה השמור • סיבוכיות: O(1)
isInnerNode	• קלט: אין (מתודת מופע על איבר ספציפי) • פלט: ערך בולאני • שיטת פעולה: המתודה בודקת אם שדה ה-rank האיבר גדול או שווה לאפס. אם כן, סימן שמדובר באיבר פנימי, והמתודה מחזירה true. אחרת, סימן שמדובר בעלה אקסטרני, והמתודה מחזירה false. • סיבוכיות: O(1)

timore – 308145309 – תימור איזנמן
offekgil – 308315092 – אופק גיל

מדידות:

מספר סידורי	מספר פעולות	מספר פעולות האיזון הממוצע לפעולת insert	מספר פעולות האיזון המקסימלי לפעולת insert	מספר פעולות האיזון לממוצע לפעולת delete	מספר פעולות האיזון המקסימלי לפעולת delete
1	10,000	3.4042	15	2.0339	9
2	20,000	3.38685	18	2.02275	10
3	30,000	3.43103	18	2.02296	10
4	40,000	3.41205	19	2.020375	11
5	50,000	3.40918	18	2.03228	10
6	60,000	3.41686	18	2.025916	11
7	70,000	3.40737	18	2.03111	11
8	80,000	3.41678	19	2.033925	12
9	90,000	3.40765	19	2.031966	12
10	100,00	3.40261	19	2.04219	12

- ציפיות על בסיס רקע תיאורטי:
  - מספר פעולות האיזון הממוצע לפעולת insert ולפעולת delete צפוי להיות מספר קבוע - $O(1)$ ללא תלות בגדול העץ או בגובהו.
  - מספר פעולות האיזון המקסימלי לפעולת insert או delete צפוי להיות מסדר הגודל של גובה העץ-שכידוע חסום מלמעלה ע"י $2\log n$ – כאשר n הוא מספר האיברים בעץ. לכן, נצפה שמספר זה יהיה מסדר הגודל של log(n), ויגדל במתינות עם גידול כמות האיברים בעץ.
- תוצאות המדידות:
  - מספר פעולות האיזון הממוצע לפעולת insert או delete בכל אחת מהמדידות היה **קבוע** (עם סטיות זניחות), ולא השתנה כפונקציה של מספר האיברים בעץ. אם כך, ניתן להסיק שתוצאות אלה עולות בקנה אחד עם ניתוח ה-amortized התיאורטי של מספר פעולות האיזון עבור פעולת הכנסה ומחיקה שהתבצע בהרצאה.
  - מספר פעולות האיזון המקסימלי הן עבור פעולת insert בודדת והן עבור פעולת delete בודדת היה מסדר הגודל של $logn$ - כאשר n הוא מספר האיברים בעץ. מספר זה גדל במתינות ככל שהגדלנו את כמות האיברים בעץ. ניתן להסיק כי גם תוצאות אלו תואמות את הניתוח התיאורטי שבוצע בכיתה.