**עץ WAVL**

קארין בל פדר

תמי לביא

תיאור המחלקה

למחלקת WAVLTree ישנם 3 משתני מחלקה, שלושתם מטיפוס WAVLNode (יורחב בהמשך):

1. ***root*** המשמש כמצביע לשורש העץ הנוכחי.
2. ***max\_node*** המצביע לאיבר המקסימלי בעץ.
3. ***min\_node*** המצביע לאיבר המינימלי בעץ.
4. ***Size***int המחזיק את גודל העץ הנוכחי.

למחלקה שלוש מחלקות פנימיות:

1. ***AbsWAVLNode:*** מחלקה אבסטרקטית המייצגת קודקוד כלשהו בעץ. לכל קודקוד דרגה (rank) ואבא (parent) מסוג WAVLNode (מפורט בנקודה הבאה). זאת כיוון שקודקוד חיצוני לא יכול להיות אבא של קודקוד בעל מידע.
2. ***WAVLNode:*** מייצגת קודקוד בעל מידע בעץ, ועל כן יש לה מפתח (key) וערך (value). בנוסף מחזיק בן שמאלי וימני מסוג AbsWAVLNode. בנים אלו לעולם לא יהיו null - או שיהיו בעלי מידע – WAVLNode אחרים, או שיהיו מסוג WAVLExternalNode.

למחלקה מספר מתודות עזר לא טריוויאליות:

* *getRankDiff():* מחזיר את ההפרש בין הדרגה של הבן השמאלי לשל הבן הימני. משמש להחלטה לאיזה כיוון לאזן את העץ.
* *updateRank():* מעדכן את הדרגה בהתאם לילדיו. מחזיר Boolean האם הדרגה השתנתה.
* *isValidRankDiff():* האם הפרשי הדרגות של הקודקוד מול ילדיו חוקי, ובפרט אם הוא עלה פנימי האם דרגתו 0.

1. ***WAVLExternalNode:*** מחלקה המייצגת עלים חיצוניים. מכאן שדרגתם תמיד 1-, ואין להם בנים.

מימוש הפונקציות:

# פונקציות עזר כלליות:

## ***Private void rotateRightLeftToRightRight(WAVLNode source, WAVLNode child):***

מתודת עזר שמטרתה לבצע פעולת רוטציה ראשונה מ-Right-Left ל-Right-Right במידה ונדרש double rotation. O(1).

## ***private int rebalanceRightSide(WAVLNode source):***

מתודת עזר שמטרתה לאזן את צידו הימני של NODE, כולל להחליף את מקומו. במידה ונדרשת רוטציה כפולה תקרא ל-rotateRightLeftToRightRight. תחזיר 1 במידה ובוצעה רוטציה יחידה, 2 אם בוצעה רוטציה כפולה, כלומר נקראה המתודה הקודמת. סיבוכיות זמן ריצה .

## ***private void rotateLeftRightToLeftLeft(WAVLNode source, WAVLNode child):***

אחראית על ביצוע רוטציה ראשונה במידה ונדרש double rotration בצד השמאלי של העץ. תחליף מצב Left-Right למצב Left-Left. .

## ***private int rebalanceLeftSide(WAVLNode node):***

מתודת עזר לאיזון צידו הימני של ה-node, כולל החלפת מקומו. במידה ונדרשת רוטציה כפולה תקרא לפונקציה הקודמת. תחזיר 1 אם בוצעה רוטציה יחידה, 2 אם כפולה. סיבוכיות זמן ריצה .

## ***private rebalance(WAVLNode node, int count):***

מתודת עזר רקורסיבית לאיזון העץ. תפסיק או כאשר הגענו ל-root ואיזנו אותנו, או אם הקודקוד הנוכחי "ספג" את השינוי בעץ ולא נדרש עדכון ב-RANK שלו.

המתודה בודקת האם לאחר שינוי ה-RANK הקודקוד מאוזן. במידה ולא – נבדוק לאיזה כיוון חוסר האיזון נוטה, ובהתאם נקרא לאחת המתודות rebalanceLeftSide, rebalanceRightSide. במידה והקודקוד מאוזן (ואכן עודכן – אחרת היינו בתנאי עצירה) נאזן את אביו.

המתודה מחזירה מספר פעולות האיזון שבוצעו לפי הספירה הבאה:

* רק עדכון RANK (כלומר promotion\demotion) – 1
* רוטציה יחידה (כולל החלפת קודקודים ועדכון דרגה) – 1
* רוטציה כפולה (כולל 2 החלפות ועדכוני דרגה) – 2

סיבוכיות זמן ריצה: . במידה ונדרש לעדכן מעלה ועד השורש נגיד לסיבוכיות זמן ריצה זו (נלמד בשיעור).

## ***private WAVLNodefindClosestNode(int k):***

מתודת עזר שמטרתה למצוא את הקודקוד הקיים "הקרוב ביותר" למפתח k. המתודה פועלת מ-min כלפי מעלה, ומחפשת את האב הקדמון המשותף לmin ו-k. לאחר מכן תרד כלפי העלים.

הקודקוד שנחזיר יהיה אחד משלוש: קודקוד בעל מפתח k – כלומר מצאנו את הקודקוד עצמו. קודקוד בעל מפתח גדול מ-K שהבן השמאלי שלו הוא WAVLExternalNode (כלומר המקום החוקי להכניס קודקוד עם מפתח k משמאלו). קודקוד בעל מפתח קטן מ-K שהבן הימני שלו הוא WAVLExternalNode (כלומר המקום החוקי להכניס קודקוד עם מפתח k מימינו).

סיבוכיות זמן ריצה: .

# פונקציות מטרה:

## ***public int insert(int k, String i):***

מתודה שאחראית על הכנסת איבר חדש k במידה ואינו קיים בעץ. מחזירה את כמות פעולות האיזון השונות שנעשו בשביל לשמור על העץ.