**תיעוד סיבוכיות זמן הריצה**

**balance(self, node)** – הפעולה מבצעת את ה"תיקונים" הנדרשים החל מ-node ועד לשורש העץ, כולל עדכון size ו-height. ברגע שהפעולה נתקלת בצומת עם balance factor 2 או 2- היא קוראת ל-rotate שדואגת לבצע את ה"גלגולים" הנדרשים תוך צבירת מספר הגלגולים שמתבצעים במסלול עד השורש. מדובר במסלול מצומת לעלה, ולכן במקרה הגרוע סיבוכיות זמן הריצה תהיה O(logn).

**rotate(self, node)** – הפעולה מסווגת את הצומת הבעייתי לאחד מסוגי הגלגול שלמדנו (גלגול ימינה, גלגול שמאלה, גלגול שמאלה ואז ימינה, וגלגול ימינה ואז שמאלה) בהתאם לערך ה-balance factor שלו ושל בנו הישיר. הפעולה סך הכול בודקת את ערך ה-balance factor של node ושל בנו, ולכן סיבוכיות זמן הריצה שלה היא O(1).

**rotateR(self, node)** – הפעולה מבצעת גלגול ימינה כפי שנלמד בהרצאה. הפעולה סך הכול משנה מצביעים ולכן סיבוכיות זמן הריצה שלה היא O(1). באופן סימטרי, rotateL(self, node) מבצעת גלגול שמאלה.

**rotateLR(self, node)** – הפעולה מבצעת שני גלגולים כפי שנלמדנו בהרצאה. תחילה היא מבצעת גלגול שמאלה על מנת להביא את הצמתים למצב שגלגול ימינה יביא לעץ מאוזן. לאחר הגלגול שמאלה, הפעולה קוראת ל-rotateR על מנת להשלים את המלאכה. באופן סימטרי, rotateRL(self, node) מבצעת שני גלגולים גם כן, תחילה גלגול ימינה, ולאחר מכן גלגול שמאלה. סיבוכיות זמן הריצה של שתי פעולות אלה הינו O(1).

**retrieve(self, i)** – הפעולה קוראת ל-select עם i+1 ומחזירה את ערך הצומת שמוחזר. בכך, הפעולה מחזירה את ערך הצומת עם הדרגה ה-i+1, או במילים אחרות הצומת במקום ה-i בעץ. סיבוכיות זמן הריצה של הפעולה במקרה הגרוע הוא כזמן הריצה של select – O(logn).

**insert(self, i, val)** – הפעולה מכניסה צומת עם ערך val במקום ה-i בעץ ומחזירה את מספר פעולות האיזון שהתבצעו. הפעולה למעשה מגיעה לצומת ה-i בעזרת קריאה ל-select, ובמידה ואין לאותו צומת בן שמאלי, היא מכניסה שם את הצומת החדש, אך אם הבן השמאלי תפוס, פונים לpredecessor ומכניסים את הצומת החדש כבן שמאלי שלו. לסיום מתבצעת קריאה ל-balance על מנת לדאוג שהעץ יהיה מאוזן לאחר ההכנסה. סיבוכיות זמן הריצה של הפעולה במקרה הגרוע הוא O(logn), בשל השימוש ב-select. מציאת ה-predecessor לוקחת כמובן O(1).

**delete(self, i)** – הפעולה מוחקת את הצומת במקום ה-i בעץ. הפעולה קוראת ל-deleteNode עם הצומת הנדרש בעזרת קריאה ל-select. deleteNode דואגת לנתק את הצומת שהיא מקבלת מהעץ בזמן קבוע. לסיום, delete קוראת ל-balance על מנת לדאוג לאיזון העץ. סיבוכיות זמן הריצה במקרה הגרוע היא O(logn).

**listToArray(self)** – הפעולה מחזירה את רשימת האיברים שהעץ מייצג. לשם כך, אנחנו פונים לאיבר הראשון, ולאחר מכן מבצעים פעולת successor על מנת למצוא את האיבר שבא אחריו. סיבוכיות זמן הריצה במקרה הגרוע היא O(n), שכן אנחנו מבצעים n-1 קריאות successor.

**split(self, i)** – הפעולה מפצלת את העץ לפי הצומת במקום ה-i. היא מחזירה רשימה באורך 3 כאשר האיבר הראשון בה הוא עץ המכיל את האיברים שנמצאים לפני המקום ה-i, האיבר השני הוא ערך הצומת במקום ה-i, והאיבר השלישי הוא עץ המכיל את האיברים שנמצאים אחרי המקום ה-i. אנחנו משתמשים ב-select כדי להגיע לצומת במקום ה-i, ולאחר מכן עולים אל השורש ובעזרת join והאלגוריתם שנלמד בהרצאה, אנחנו "צוברים" את שני העצים שעלינו להחזיר. סיבוכיות זמן הריצה של הפעולה במקרה הגרוע היא O(logn).

**concat(self, lst)** – הפעולה משרשרת לסוף העץ הנוכחי את העץ lst. בעזרת הפעולה join ובעזרת האלגוריתם שנלמד בכיתה, סיבוכיות זמן הריצה של הפעולה נשמר O(logn).

**search(self, val)** – הפעולה מחזירה את המיקום של האיבר עם הערך val. הפעולה מבצעת קריאות ל-precessor עד שהיא מגיעה לערך הנדרש. סיבוכיות זמן הריצה במקרה הגרוע של הפעולה הינו O(n), שכן אנחנו נבצע במקרה הגרוע ביותר n קריאות ל-precessor.

**rank(self, node)** – הפעולה מחזירה את הדרגה של node. הפעולה עובדת לפי האלגוריתם שנלמד בכיתה, החיפוש מתחיל מ-node וממשיך עד שמגיעים לשורש. בכל פעם שאנחנו "עולים שמאלה", אנחנו מוסיפים את ה-size של הבן השמאלי של האבא + 1. סיבוכיות זמן הריצה של הפעולה היא O(logn).

**select(self, i)** – הפעולה מקבלת דרגה של צומת בעץ ומחזירה את הצומת עם הדרגה הזו. הפעולה עובדת בהתאם לאלגוריתם שנלמד בהרצאה. החיפוש מתחיל מהשורש - אם הדרגה שאנחנו מחפשים קטנה יותר מדרגת השורש, נפנה לבן השמאלי שלו. אם הדרגה גדולה מדרגת השורש, נפנה לבן הימני ונחסר מהדרגה שאנחנו מחפשים את דרגת השורש. כך באופן איטרטיבי עד שנגיע לצומת המבוקש. סיבוכיות זמן הריצה של הפעולה היא O(logn) במקרה הגרוע.

**successor(self, node)** – הפעולה מחזירה את ה-successor של node לפי האלגוריתם שנלמד בהרצאה. באופן סימטרי, predecessor מחזירה את ה- predecessor. סיבוכיות זמן הריצה של שתי פעולות אלה הינו O(1).