אור אבנר 203057781

אופיר אורנשטיין 203258918

**תרגיל רטוב 1- חלק יבש**

על מנת לתמוך בפעולות הנדרשות בתרגיל זה יצרנו מספר מחלקות בc++.

ראשית, יצרנו **מחלקה של בית ספר** המכילה שלושה עצי AVL מאוזנים עליהם נפרט בהמשך והמתודות בה הן המימושים לפונקציות הנדרשות. על כן נדרשנו לממש **מחלקה נוספת לעץ AVL מאוזן** המכילה את כל השדות והמתודות הנדרשות למימוש הפעולות שלמדנו בכיתה בסיבוכיות שנלמדה:

1. בנאי דיפולטיבי לעץ בO(1).
2. הורס דיפולטיבי (רקורסיבי) בO(n).
3. פעולות הכנסת צומת והוצאת צומת בO(logn).
4. קבלת גודל העץ בO(1) (ע"י שמירת משתנה גודל ועדכונו בכל הוצאה והכנסה).
5. מימוש צומת המכילה – מפתח, ערך, מצביע להורה, מצביע לבן שמאלי ומצביע לבן ימני וגובה העץ.
6. יצרנו מתודה להמרת העץ למערך ממוין לפי המפתח וכן מתודה להמרת מערך ממוין לעץ. 2 המתודות משתמשות בטכניקות שנלמדו בתרגול בסיבוכיות של O(n).

וכמובן יצרנו פונקציות ופונקציות עזר שמאפשרות לנו את זה.

בנוסף יצרנו את שלוש המחלקות הבאות:

1. **מחלקת PowerID**- זו מחלקה ששדותיה הם המזהה של המוטנט והעוצמה שלו והיא מממשת את האופרטורים >, ==, =+, כדי לתת מענה ליחס הסדר הנדרש כאשר אנו מבצעים מיון לפי עוצמה והעוצמות שווה ומה שקובע את הסדר הוא המזהה.
2. **מחלקת מוטנט**- מכילה את כל השדות הדרושים לתיאור מוטנט:
3. StudentID.
4. Grade.
5. Power (מטיפוס PowerID).
6. מצביע לצומת של הצוות.
7. **מחלקת צוות**- מכילה את כל השדות הדרושים לתיאור צוות:
8. teamID.
9. עץ AVL המכיל את כל המוטנטים באותו צוות וממוין לפי ה-Power של כל מוטנט (כמו עץ 3 המוזכר מטה, רק לא חייב להכיל את כל המוטנטים).

**במחלקת בית הספר** ישנם שלושה עצי AVL מאוזנים:

1. עץ AVL של מוטנטים שהמפתח שלו הוא מזהה הסטודנט (studentID) לפיו הוא מחפש בעץ (עץ זה מכיל את כל המוטנטים שקיימים בבית הספר בכל רגע נתון).

עץ זה מכיל את כל המוטנטים בבית הספר לכן סיבוכיות המקום היא O(n) כאשר n זה מספר המוטנטים בבית הספר.

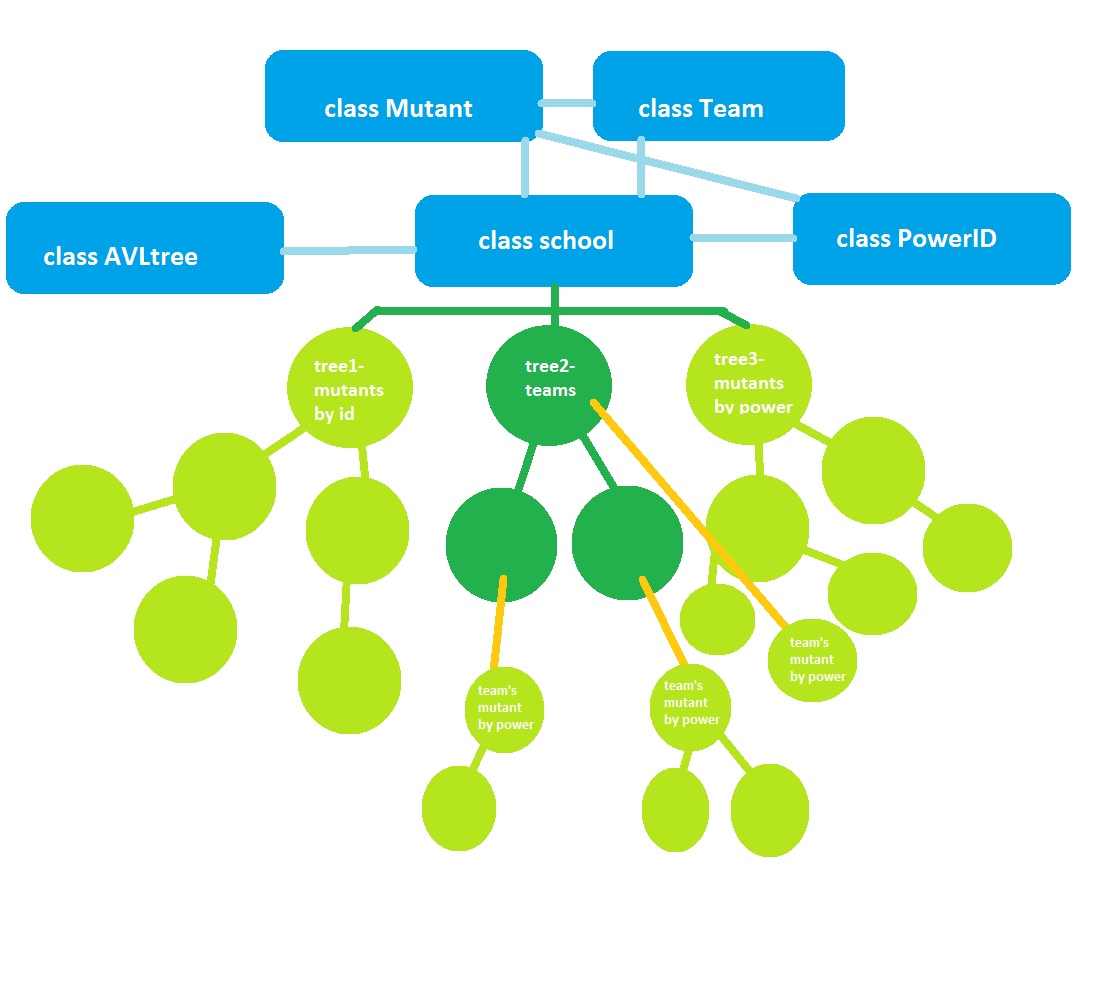
1. עץ AVL של צוותים שהמפתח שלו הוא מזהה הצוות (TeamID) לפיו הוא מחפש בעץ. כל צומת בעץ היא למעשה צוות וקיים בה מצביע לעץ קטן נוסף כמו עץ 3 שמסודר לפי מפתח של PowerID-(עוצמה עם משמעות למזהה כאשר העוצמה זהה) המכיל רק את המוטנטים ששייכים לצוות. מכוון שבכל רגע נתון אנחנו מכילים את כל הצוותים ואת כל המוטנטים שיש להם צוות (מקסימום n מוטנטים ולכן נסמן את מספרם בm כך שm<=n) והשדות קבועים בכל איבר אז סיבוכיות המקום היא k\*O(1)+m\*O(1)=O(k)+O(n)=O(k+n).
2. עץ AVL של מוטנטים שהמפתח שלו זה PowerID- לפיו הוא מחפש בעץ (עץ זה מכיל את כל המוטנטים שקיימים בבית הספר בכל רגע נתון).

עץ זה מכיל את כל המוטנטים לכן סיבוכיות המקום היא O(n).

סה"כ סיבוכיות המקום 2\*O(n)+O(k+n)=O(k+n).

בית הספר הוא מבנה נתונים בעל שלושה שדות – אחד עבור כל מצביע לכל עץ .

במחלקת בית הספר מימשנו את כל המתודות הבאות כך שהפונקציות הנדרשות בתרגיל, המקבלות לרוב מצביע למבנה נתונים שהוא בית הספר, יוכלו להפעיל את המתודות על בית הספר עצמו לאחר בדיקה שהמצביעים שהתקבלו בפרמטרים אינם NULL.



**הפונקציות שמימשנו:**

1. Init() – יוצרת ומאתחלת מבנה נתונים ריק של בית הספר. פונקציה זו אינה מקבלת פרמטרים ומחזירה מצביע למבנה הנתונים של בית הספר. אנו יוצרים אובייקט חדש של בית ספר שבאופן דיפולטיבי יוצר לנו את שלושת העצים ריקים.

מכוון שביצענו מספר ידוע מראש של פעולות סיבוכיות הזמן היא O(1).

\*\*\*מכאן ואילך כל הפונקציות הנדרשות מקבלות כפרמטר מצביע למבנה הנתונים (בית הספר) ופרמטרים נוספים, בכל הפונקציות נבדוק שמצביע (ומצביעים נוספים אם ישנם) אינם null ובמידה והם כן- נחזיר הודעת שגיאה מתאימה.במידה ולא, נפעיל על המבנה את המתודה המתאימה לפונקציה שמימשנו במחלקת בית הספר (כי המבנה מטיפוס בית ספר) ונחזיר את הערך המוחזר מהן.

1. AddStudent – פונקציה זו מקבלת את המצביע למבנה הנתונים בית הספר DS, את מזהה המוטנט/הסטודנט StudentID, את מספר הכיתה אליה נכנס המוטנט Grade, ואת רמת הכוח של המוטנט Power. פונקציה זו מוסיפה את המוטנט לכיתה. פונקציה זו מפעילה על המבנה שהוא בית הספר את המתודה add\_student ומעבירה לה את כל שאר הפרמטרים.

במתודה זו, ראשית ניצור אובייקט חדש מסוג מוטנט ונאתחל את כל השדות על פי הנתונים. יצירת האובייקט ואתחולו הוא בO(1) מכוון שמדובר במספר סופי של פעולות וידוע מראש. במידה והייתה שגיאת אלוקצייה נחזיר הודעת שגיאה מתאימה.

לאחר מכן נוסיף לעץ מצביע למוטנט ע"י כך שנפעיל על עץ המוטנטים (שהמפתח שלו זה מזהה הסטודנט, עץ 1) את מתודת העץ להכנסת אובייקט חדש לעץ (ממומש לפי הנלמד בכיתה, סיבוכיות O(logn)). במידה והסטודנט קיים בבית הספר או שהייתה בעיה בהקצאת הזיכרון- סיימנו ונחזיר הודעת שגיאה מתאימה.

אחרת, נוסיף לעץ מצביע למוטנט ע"י כך שנפעיל את מתודת ההכנסה לעץ על עץ המוטנטים עם המפתח PowerID (אם לא הצליחה הקצאת הזיכרון נחזיר הודעת שגיאה מתאימה). הוספת אובייקט לעץ AVL זו פעולה שסיבוכיות הזמן שלה היא O(logn) כפי שלמדנו בכיתה ולכן סה"כ במקרה הגרוע ביותר בו חיפשנו את הסטודנט והוא לא קיים בבית הספר ולכן יצרנו אותו והוספנו לעצים סיבוכיות הזמן תהיה: O(1)+2\*O(logn)=O(logn). כאשר n זהו מספר הסטודנטים הקיימים בבית הספר.

1. AddTeam- פונקציה זו מקבלת מצביע למבנה נתונים של בית ספר ומזהה צוות וצריכה להוסיף את הצוות למבנה. פונקציה זו מפעילה על בית הספר (המבנה) את המתודה שלו add\_team ומעבירה את מזהה הצוות כפרמטר. במידה והמזהה אינו חיובי נחזיר הודעת שגיאה מתאימה. אחרת נפעיל על עץ 2- עץ הצוותים את מתודת הכנסת איבר לעץ- סיבוכיות הזמן כפי שהסברנו קודם וכנלמד היא O(logk) כאשר k הוא מספר הצוותים. במידה וקיים בעץ צוות עם מזהה זה או שפעולת הקצאת הזיכרון כשלה נחזיר הודעת שגיאה מתאימה, אחרת הצלחנו. סה"כ סיבוכיות O(logk).
2. MoveStudentToTeam- פונקציה זו מוסיפה סטודנט לפי מזהה לצוות לפי מזהה. היא מקבלת כפרמטרים מצביע למבנה הנתונים (בית הספר), מזהה הסטודנט studentID, מזהה הצוות אליו יכנס הסטודנט המוטנט teamID . פונקציה זו מפעילה על בית הספר (המבנה) את המתודה שלו move\_student\_to\_team ומעבירה את שני המזהים. מתודה זו בודקת שהפרמטרים חיוביים ובמידה ולא מחזירה הודעת שגיאה מתאימה. אם תקינים היא מאתרת את המוטנט בעזרת המזהה שלו בעץ מספר 1 ע"י שימוש במתודת החיפוש של העץ בסיבוכיות של O(logn) כנלמד. לאחר מכן מאתרת באותו אופן את הצוות אליו המוטנט יעבור בעץ הצוותים – עץ 2 בסיבוכיות של O(logk). אם אחד מהמזהים לא נמצא בעץ תוחזר הודעת שגיאה מתאימה. לאחר מכן, הפונקציה מכניסה לעץ הצוותים באמצעות מתודת ההכנסה של העץ את המוטנט לעץ המוטנטים של הצוות החדש אותו איתרה בO(logn\_team1) . במידה וההכנסה נכשלה נחזיר הודעת שגיאה ונצא, אחרת נבדוק אם למוטנט היה צוות אחר (יש לו מצביע לצוות שלו) ואם כן נסיר אותו משם בO(logn\_team2) . נעדכן את המצביע להצביע על הצוות החדש. סה"כ סיבוכיות הזמן היא   
    O(logn)+O(logk)+ O(logn\_team1)+O(logn\_team2)= O(logn+logk).
3. GetMostPowerful – פונקציה זו מקבלת כפרמטרים מצביע למבנה הנתונים, מזהה צוות ומצביע למזהה סטודנט בו היא מחזירה את מזהה הסטודנט של הסטודנט החזק ביותר בצוות. במידה וערך מזהה הצוות teamID שלילי – מחזירה את מזהה הסטודנט החזק ביותר בכל המערכת. נפעיל על בית הספר את המתודה שלו שאליה נעביר את הפרמטרים כפרמטרים והיא תחזיר את הנדרש. ראשית, נבדוק האם המזהה אכן שלילי- במידה וכן, בעץ 3 יש לנו מצביע למוטנט החזק ביותר בכל המערכת (ראה הרחבה בסוף לגבי תחזוקת מצביע זה), נחזיר את המזהה שלו וסיימנו ב O(1), במידה ולא- נחפש בO(logk) את מזהה הצוות וכשנגיע לצומת זו –קיים בה בה מצביע למוטנט הכי חזק בצוות ונחזיר את מזהה המוטנט בO(1), (או אם אין מוטנטים בצוות, או במערכת כולה במקרה הקודם- נחזיר -1) .בסה"כ סיבוכיות הזמן של הפעולה במקרה הגרוע היא O(logk).
4. RemoveStudent- פונקציה זו מקבלת כפרמטרים מצביע למבנה הנתונים בית הספר ומזהה סטודנט. הפונקציה מפעילה על המבנה (בית הספר) את המתודה המתאימה remove\_student ומעבירה לה כפרמטר את מזהה הסטודנט.

המתודה בודקת אם המזהה חיובי ואם לא- מחזירה הודעת שגיאה מתאימה. לאחר מכן, המתודה מפעילה על עץ 1 (עץ המוטנטים עם המפתח StudentID) את המתודה של העץ לחיפוש על פי מזהה הסטודנט שהתקבל כפרמטר. במידה והסטודנט קיים- נסיר אותו מהמערכת, אחרת נחזיר הודעת כשלון מתאימה. נחפש בעץ 1 שמסודר לפי studentID את מזהה הסטודנט בO(logn). לאחר שמצאנו נלך לשדה במוטנט שמצביע על הצוות שלו (רק אם קיים כי הוא נמצא בצוות כלשהו) בעץ 2, ניגש לצוות ולעץ המוטנטים שלו (לפי מפתח הPower) ונסיר אותו מעץ זה בסיבוכיות של O(logn). לאחר מכן נסיר את הסטודנט מעצים 1 ו3 גם כן בסיבוכיות של O(logn) באמצעות מתודת העץ. סה"כ סיבוכיות הזמן של הפונקציה היא 4\*O(logn)=O(logn).

1. GetAllStudentsByPower- פונקציה זו מקבלת כפרמטרים מצביע למבנה נתונים (בית הספר), מזהה צוות, מצביע למערך, ומצביע למספר האיברים במערך. הפונקציה מחזירה במערך את כל המוטנטים שנמצאים בצוות הרצוי ממויינים לפי הכוח שלהם בסדר יורד. אם מזהה הצוות שלילי נחזיר מערך ובו ממויינים כל המוטנטים במערכת. נפעיל על בית הספר את המתודה המתאימה- get\_all\_students\_by\_power המקבלת את אותם פרמטרים פרט למצביע למבנה ומחזירה את ערך ההחזרה הרצוי. ראשית ניצור עץ (בO(1)) שיצביע לעץ הרלוונטי- אם מזהה הצוות קטן מאפס אז לעץ 3 ואחרת יצביע לעץ המוטנטים של הצוות הרלוונטי (נמצא את הצוות ע"י הפעלת מתודת החיפוש לפי מזהה צוות על עץ 2 בO(logk)). אם אין צוות כזה נחזיר הודעת שגיאה מתאימה. אם אין מוטנטים בעץ בכלל או בצוות נחזיר 0 במספר והצלחה כערך החזרה. אחרת, נשתמש במתודת העץ שמעבירה את העץ מעץ למערך ממויין בO(n) במקרה שהמזהה שלילי ועוברים על כל המוטנטים ובO(n\_team) במקרה שקיים צוות כזה ועוברים על המוטנטים בו (נפרט בהמשך איך) ונקצה מערך של שלמים בגודל מספר הסטודנטים בעץ. נעבור איבר איבר במערך המוטנטים ב-O(n/n\_team) ומכוון שהוא ממוין בסדר הפוך מהרצוי לנו- נמלא את המערך אותו אנו מחזירים בכוון ההפוך כאשר אנו מעתיקים רק את השדה של מזהה הסטודנט מהמוטנט. לבסוך נשחרר את מערך המוטנטים שיצרנו. אם הייתה לנו שגיאה בהקצאת הזיכרון נחזיר הודעה מתאימה.סה"כ סיבוכיות הפונקציה O(n/n\_team)+O(logk)=O(n/n\_team+logk). אם אין מוטנטים שהוקצו לצוות – נחזיר null במצביע ו0 בגודלו.
2. IncreaseLevel- פונקציה זו מקבלת מצביע למבנה הנתונים (בית הספר), כיתה Grade, ועוצמה PowerIncrease ומעלה את העוצמה של כל תלמידי הכיתה בעוצמה שהתקבלה. נשתמש במתודת הבית הספר increase\_level אליה נעביר את הכיתה והעוצמה כפרמטרים. בה נבדוק את תקינות הפרמטרים ובמידה ואינם תקינים נחזיר הודעת שגיאה מתאימה. לאחר מכן נשתמש בפונקציית בעזר שמימשנו merge שמקבלת עוצמה כיתה ועץ מוטנטים עם מפתח מטיפוס PowerID. פונקציה זו מעבירה את העץ למערך ממויין באמצעות מתודת העץ (נפרט בהמשך לגבי זה) ב O(m) כאשר m היא כמות המוטנטים בעץ ויוצרת שני מערכים נוספים- אחד למוטנטים בכיתה הרצויה והשני לאלו שלא. הפונקציה עוברת על מערך המוטנטים, מעתיקה אותם למערכים המשניים לפי התאמתם לכיתה ומי שבכיתה הרלוונטית – מעדכנת לו את העוצמה. מעבר זה מתבצע בO(m). לאחר מכן נעבור שוב על m האיברים ונאחד את שני המערכים למערך אחד ממוין. לאחר מכן נחזיר את המערך להיות עץ AVL בO(m) בדיוק כפי שנלמד בתרגול. בסיום הפעולה העץ יהיה מאוזן וחוקי. סה"כ סיבוכיות merge O(m). ראשית נשלח לmerge את עץ 3 עם הכיתה והעוצמה שבפרמטרים, סיבוכיות O(n). כעת נעביר את עץ 2 מעץ למערך באמצעות מתודת העץ ונעבור איבר-איבר על כל הצוותים בO(k) ובכל צוות נשלח לפונקציית merge את עץ המוטנטים של הצוות עם הכיתה שהתקבלה כפרמטר ועוצמה 0- זאת משום שעצי המוטנטים מכילים מצביע למוטנט ולכן שינוי העוצמה שעשינו קודם לא נדרש בשנית (כל העצים מצביעים לאותם מוטנטים). עברנו על כל הצוותים ועל כל המוטנטים ולכן סיבוכיות הפעולה O(n+k) שזהו גם החסם של כל הפונקציה מכוון ש O(n)+O(k+n)=O(k+n).
3. Quit- הפונקציה מקבלת מצביע למבנה הנתונים בית הספר ומשחררת את המבנה ע"י קריאה לדיסטרקטור הדיפולטיבי של בית הספר. בדיסטרקטור אנו נשתמש במתודה של העץ שהופכת אותו מעץ למערך ממויין בO(k) ונהפוך את עץ הצוות למערך בו כל איהר הוא מצביע לצוות. כעת נעבור צוות צוות ונשחרר בו את עץ הסטודנטים שלו. סה"כ עברנו על k עצים ובהם בסה"כ לכל היותר n מוטנטים ולכן סיבוכיות הפעולה היא O(n+k). נשחרר את המערך. כעת נהפוך את עץ 1 למערך ונעבור בו על כל המוטנטים ונשחרר אותם (זה נדרש משום שבעץ יש רק מצביע למוטנט. סה"כ סיבוכיות הפעולה היא O(n+n)=O(n). נשחרר את המערך. לבסוף נשחרר את בית הספר בO(1) ויהרסו גם העצים המוגדרים בו (שיחררנו כבר את כל העצמים). סה"כ סיבוכיות הזמן O(n+k)+O(n+n)+O(1)=O(n+k)

**בונוס:**

על מנת לבצע את Level Increase בO(n) בלבד היינו יוצרים במחלקת בית הספר עץ AVL נוסף (עץ 4) לכל הצוותים הלא ריקים, כלומר שיש בהם לפחות מוטנט אחד. מספר הצוותים האלו הוא במקרה הגרוע ביותר n (כאשר יש מוטנט אחד בכל צוות ) ולכן קטן בהכרח מk הצוותים שיש שיכולים להיות גם צוותים ריקים וגם כל אלו שאינם ריקים כך שהקטנו את העץ.

כל צומת בעץ זה תהיה מסוג צוות ובה מצביע לעץ המוטנטים הקטן (כמו עץ3) שממויין לפי PowerID עליו מצביע גם הצוות בעץ הצוותים שתיארנו לעיל (עץ 2).

נעדכן בO(n) את עץ 3 שמכיל את כלל המוטנטים בO(n) כמו קודם. אך כעת כדי לעדכן את עצי המוטנטים אליהם מצביעים הצוותים ולאזן אותם מחדש לפי העוצמות אנו לא נדרשים לעבור בO(k) על כל הצוותים אלא רק על הצוותים שבהם יש מוטנטים, שבמקרה הכי גרוע זה בO(n) ושם לבצע את ההעתקה למערך ממויין, מיזוג וכו' כמו קודם.

סה"כ הסיבוכיות הנדרשת כעת היא O(n)+O(n)+O(n)= O(n)

כמובן שיצירת העץ הנוסף דורשת גם תחזוקה בפעולות נוספות:

move\_student\_to\_team- אם הסטודנט השתייך לצוות כבר, נבדוק אם הוא היה חבר הצוות היחיד ואם כן כשנסיר אותו מן הצוות נסיר גם את הצוות שלו מעץ 4. גם פעולת חיפוש הצוות וגם פעולות ההסרה מעץ 4 שתיהן עומדות בסיבוכיות הפעולה שהיא O(logk). כמו כן נבדוק אם הוא הסטודנט הראשון המצטרף לצוות אליו הוא עובר. אם כן, נוסיף את הצוות לעץ 4 בo(logk). לכן סה"כ לא פגענו בסיבוכיות והיא עדיין O(logn+logk).

remove\_student- כאשר נסיר סטודנט מבית הספר נבצע הכל בדיוק כמו לפני השינוי וכאשר נגיע לצוות שלו- נבדוק אם היה החבר היחיד בצוות. אם כן, נחפש את הצוות בעץ 4 בO(logn) כי כפי שהסברנו המספר המקסימלי של איברים שיכולים להיות בעץ זה הוא n כאשר כל מוטנט בצוות אחר וחיפוש בעץ כזה הוא O(logn). כשנמצא נסיר את הצוות מעץ 4 בO(logn). סה"כ לא פגענו בסיבוכיות הפעולה והיא עדיין מתבצעת בO(logn).

**הסבר על המתודה tree\_to\_array שמעבירה את העץ מעץ למערך ממויין לפי המפתח:**

מקצים מערך בגודל מספר הצמתים בעץ שכל תא בו מטיפוס צומת עםkey וvalue. מכוון שהעץ הוא עץ AVL על מנת שהמערך יהיה ממויין נסייר בעץ inorder כאשר לשדה המפתח נעתיק את המפתח של העץ ובשדה בvalue נשים מצביע לאיבר המתאים בעץ.

**הסבר כיצד אנו מתחזקים את המצביע לסטודנט החזק ביותר בכל עץ שמסודר לפי כוח (עץ 3 או עצי המוטנטים בכל צוות):**

בכל פעולת הכנסת מוטנט לעץ מוטנטים נערוך השוואה בינו לבין המוטנט החזק ביותר עליו מצביע כרגע המצביע (ראשית מבחינת power ואם יש שיוון אז מבחינת id – מימשנו אופרטורים מתאימים במחלקה PowerID). אם הוא חזק מהמוטנט החזק ביותר נעדכן את המצביע כך שיצביע עליו.

במקרה של הסרת סטודנט בין אם זה בשל הסרתו מבית הספר או רק מהצוות אם עובר צוות- נעבור שוב על כל המוטנטים בעץ בO(logn) – תמיד נלך שמאלה כמה שניתן כי שם ימצא הסטודנט החזק ביותר וכך נעדכן את המצביע לסטודנט החזק ביותר.