אור אבנר 203057781

אופיר אורנשטיין 203258918

**תרגיל רטוב 2- חלק יבש**

ראשית, יצרנו **מחלקה של בית ספר** המכילה מבנה HashTable עם כל המוטנטים בבית הספר, מבנה UnionFind עבור הצוותים בבית הספר ומספר הצוותים בבית הספר.

על כן מימשנו את **מבנה ה HashTable**שנלמד בכיתה לפי שיטת Double Hashing (Open Addressing). בהתחלה אנו מקצים את המערך הדינמי. לאורך כל הדרך נדאג להקטין ולהגדיל אותו כך שגודלו ישאר חזקה של 2- זאת נעשה כאשר אנו מוסיפים או מסירים איבר ובשאיפה שהמערך תמיד יהיה חצי מלא. פונקציית הערבול תהיה הכפלה של הkey שהוא מזהה הסטודנט אצלנו בערך קבוע והוספת מספר הצעד כפול 3 וכל זה מודולו גודל המערך- כך שלעולם לא נעבור פעמיים על אותם מקומות בחיפוש מקום פנוי או הרצוי. כל פעולות הinsert, member,delete תהיינה בO(1) בממוצע על הקלט כפי שלמדנו בתרגול.

בנוסף מימשנו את מבנה **ה UnionFindהמשודרג** (באמצעות עצים הפוכים וכיווץ מסלולים וunion שבו מחברים עץ קטן לגדול) שנלמד בהרצאה. פירוט בעמוד הבא.

כמו כן מימשנו **מחלקה עבור עץ AVL דרגות בינארי מאוזן** המכילה את כל השדות והמתודות הנדרשות למימוש הפעולות שלמדנו בכיתה בסיבוכיות שנלמדה:

1. בנאי דיפולטיבי לעץ בO(1).
2. בנאי המקבל מערך צמתים וגודל בO(n).
3. הורס דיפולטיבי (רקורסיבי) בO(n).
4. מיזוג עצים בO(n).
5. פעולות הכנסת צומת והוצאת צומת בO(logn).
6. מציאת צומת לפי מפתח בO(logn).
7. מציאת הדרגה (rank) של מפתח מסוים בO(logn).
8. מימוש פעולת select לפי אינדקס בO(logn).
9. קבלת הצומת בעץ בעל הכוח הכי גדול בO(1) (נתחזק ע"י בדיקה בכל הכנסה והוצאה).
10. קבלת גודל העץ בO(1) (ע"י שמירת משתנה גודל ועדכונו בכל הוצאה והכנסה).
11. מימוש צומת המכילה – מפתח, ערך, מצביע להורה, מצביע לבן שמאלי ומצביע לבן ימני וגובה העץ.
12. יצרנו מתודה להמרת העץ למערך ממוין לפי המפתח וכן מתודה להמרת מערך ממוין לעץ. 2 המתודות משתמשות בטכניקות שנלמדו בתרגול בסיבוכיות של O(n).
13. יצרנו מתודה ספציפית לצורך התרגיל- get\_fighters\_total\_power עליה נפרט בהמשך המסמך בO(logn).

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. מזהה הסטודנט.
4. Power (מטיפוס PowerID).
5. מזהה הצוות.
6. **מחלקת צוות**- מכילה את השדות הבאים:
7. מספר הנצחונות של הצוות עד כה.
8. עץ AVL דרגות בינארי מאוזן המכיל את כל המוטנטים באותו צוות וממוין לפי ה-Power של כל מוטנט .

**במחלקת בית הספר** אם כן יש את השדות הבאים:

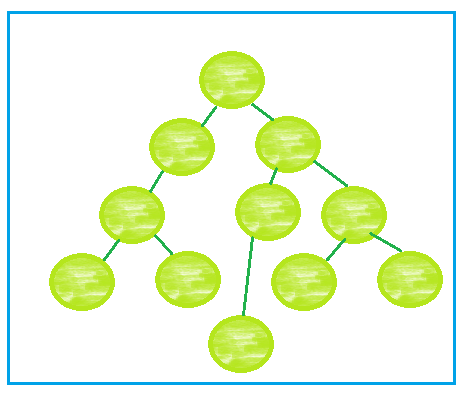
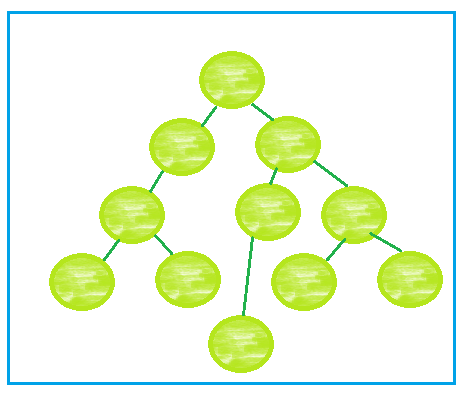
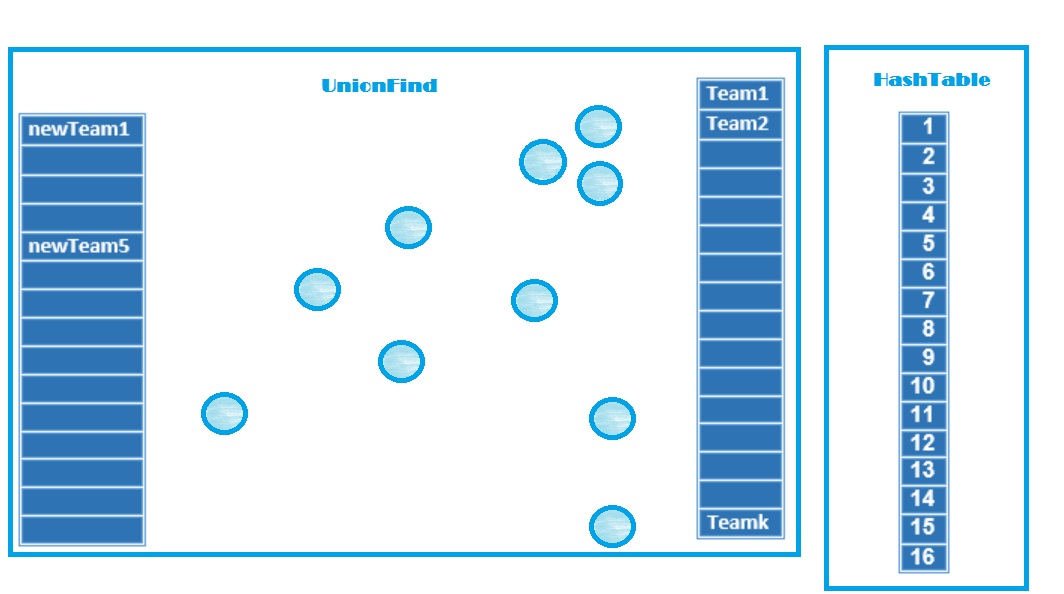
1. HashTable מטיפוס מוטנט עבור כלל המוטנטים בבית הספר. מבנה זה הוא דינמי ולכן גודלו המקסימלי ביחס למספר המוטנטים יהיה לינארי בדיוק בזמן הגדלתו כלומר - סיבוכיות המקום O(k) כאשר k הוא מספר המוטנטים בבית הספר.
2. UnionFind מטיפוס צוות עבור כלל הצוותים בבית הספר. סיבוכיות המקום O(n) במקרה הגרוע (והכללי) כאשר n הוא מספר הצוותים ההתחלתי בבית הספר מכוון שיש לנו 2 מערכים באורך n למצביעים לNODES של הצוותים (ראה שרטוט) -מערך לצוותים ומערך לצוותים המאוחדים שבהתחלה אותחל כך שלכל צוות יש תא. בנוסף יש N NODES לצוותים שמכילים מצביע לעץ המוטנטים של הצוות ולהורה ופרטים נוספים.
3. מספר הצוותים ההתחלתי בבית הספר- סיבוכיות מקום O(1).

בנוסף יכולים להיות לנו עד n עצי AVL (אחד לכל צוות) שיכיל סה"כ ביחד k מוטנטים. לכן סיבוכיות המקום הכוללת היא O(k).

**סה"כ סיבוכיות המקום O(n)+O(k) +O(k)=O(k+n) במקרה הגרוע.**

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.

**שרטוט המבנים**



**TEAM**

**TEAM**

**הפונקציות שמימשנו:**

1. **Init –** יוצרת ומאתחלת מבנה נתונים ריק של בית הספר. פונקציה מקבלת כפרמטר את מספר הצוותים ההתחלתי בבית הספר ומחזירה מצביע למבנה הנתונים של בית הספר. אנו יוצרים בעזרת הבנאי הדיפולטיבי אובייקט חדש של בית ספר שבאופן דיפולטיבי יוצר לנו אובייקט מטיפוס HashTable בגודל דיפולטיבי של 16. גודל המערך הראשוני הוא סופי ולכן האתחול שלו גם לוקח זמן סופי של O(1) כפי שנלמד בהרצאות. בנוסף יוצרים ומאתחלים באופן דיפולטיבי אובייקט UnionFind עבור n הצוותים ב-O(n) ושומרים את מספר הצוותים הראשוני n בO(1).

**סה"כ סיבוכיות זמן במקרה הגרוע: O(1)+O(n)+O(1)=O(n)**

\*\*\*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 מזהה הצוות אליו נכנס המוטנט ואת רמת הכוח של המוטנט Power. פונקציה זו מוסיפה את המוטנט לצוות. פונקציה זו מפעילה על המבנה שהוא בית הספר את המתודה add\_student ומעבירה לה את כל שאר הפרמטרים.

במתודה זו, ראשית נבדוק את תקינות הארגונטים. במידה ואינם תקינים נחזיר שגיאה מתאימה. אחרת 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. שנית, נכניס את המוטנט לטבלת הערבול של המוטנטים- הוספה כזו תיקח O(1) במשוערך בממוצע על הקלט. לאחר מכן נמצא את הצוות של הסטודנט בUnionFind (Find ב-O(log\*n) משוערך כפי שנלמד בכיתה) ונכניס את הסטודנט לעץ המוטנטים של הצוות ב O(logk\_team). במקרה הגרוע ביותר כל k הסטודנטים בבית הספר יהיו באותו צוות והסיבוכיות תהיה O(logk). כמובן תוך הכנסת הסטודנט לעץ נדאג לעדכן את מספר המוטנטים תחת כל צומת במסלול הרלוונטי ואת סכום הכוחות שתחת כל צומת, בכל צומת במסלול.

במידה והסטודנט חזק מהסטודנט החזק ביותר שיש כרגע בצוות נעדכן את השדה בעץ ששומר את נתון זה. במידה והייתה שגיאת אלוקצייה באחד השלבים נבטל את הפעולות שכן בוצעו ונחזיר הודעת שגיאה מתאימה.

**סה"כ סיבוכיות הזמן המשוערכת בממוצע על הקלט: O(1)+O(log\*n)+O(logk)=O(log\*n+logk) כאשר n מספר הצוותים הראשוני וk מספר המוטנטים בבית הספר.**

.

1. **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. אחרת, המתודה משתמשת במתודה get שזה כמו Member של הHashTable ומקבלת את המוטנט שמתאים למזהה המוטנט בO(1) בממוצע על הקלט. לאחר מכן נמצא את הצוות של הסטודנט בUnionFind (Find ב-O(log\*n) משוערך כפי שנלמד בכיתה) ונסיר את הסטודנט מעץ המוטנטים של הצוות ב O(logk\_team). במקרה הגרוע ביותר כל k הסטודנטים בבית הספר יהיו באותו צוות והסיבוכיות תהיה O(logk). בעת הסרת המוטנט מהעץ נדאג גם לבדוק אם המוטנט שהסרנו היה החזק ביותק בצוות ואם כן- נמצא את הסטודנט החזק ביותר בצוות בO(logk) . כמו כן, נדאג שבעת ההסרה והגלגולים יעודכנו בכל צומת רלוונטית במסלול מספר המוטנטים שיש כעת תחתיה וסכום כוחותיהם, לא נדרש כמובן לעדכן הכל ולכן תחזוקה זו תהיה ב O(logk) ותתבצע תוך כדי הסרת המוטנט ולא כפעולה נפרדת שדורשת מעבר נוסף. לאחר מכן נסיר את המוטנט מהHashTable בO(1) בממוצע על הקלט כפי שנלמד בכיתה .

**סה"כ סיבוכיות הזמן המשוערכת בממוצע על הקלט: O(1)+O(log\*n)+O(logk)+O(1)=O(log\*n+logk) כאשר n מספר הצוותים הראשוני וk מספר המוטנטים בבית הספר.**

1. **JoinTeams-** פונקציה זו מקבלת כפרמטרים מצביע למבנה הנתונים בית הספר ומזהים של שני צוותים ותפקידה לאחד את שני הצוותים לכדי צוות אחד. הפונקציה מפעילה על המבנה (בית הספר) את המתודה המתאימה join\_teams ומעבירה לה כפרמטרים את שני מזהי הצוותים. המתודה בודקת אם המזהים חיוביים וקטנים מn ואם לא- מחזירה הודעת שגיאה מתאימה. אחרת, נשתמש בFind של הUnionFind כדי למצוא את שני הצוותים (Find ב-O(log\*n) משוערך כפי שנלמד בכיתה). לאחר מכן נשתמש בUnion של הUnionFind כדי לאחד את שני הצוותים במבנה. פעולה זו לוקחת O(log\*n) משוערך כפי שנלמד בכיתה. בנוסף נמזג את שני עצי הAVL של הצוותים לכדי עץ אחד באמצעות המתודה merge של העץ. מתודה מקבלת שני עצי AVL בינארים מאוזנים והופכת כל אחד מהם למערך ממויין בO(k\_taem) באמצעות מתודה **tree\_to\_array** (הסבר מטה). לאחר מכן ממזגת בין המערכים לתוך מערך חדש ממויין בO(k1+k2) מכוון שעוברים על שני המערכים ולבסוף הופכת את המערך חזרה לעץ בO(k1+k2) כפי שנלמד בתרגול ומחזירה אותו. בבניית העץ כמובן יתעדכנו גם הערכים המתאימים במספר המוטנטים שתחת כל צומת ומה סכום הכוחות של כל המוטנטים שתחת אותה צומת. כמו כן, נעדכן את המשתנה שזוכר מי הוא המוטנט החזק ביותר בעץ הצוות להיות החזק מבין השניים.

**סה"כ סיבוכיות הזמן המשוערכת: 2\*O(log\*n)+O(log\*n)+O(k1)+O(k2)+2\*O(k1+k2)=O(log\*n+k1+k1) כאשר n מספר הצוותים הראשוני וk1,k2 הם מספר המוטנטים בצוותים team1,team2 בהתאמה.**

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.

1. **TeamFight**- פונקציה זו מקבלת כפרמטרים מצביע למבנה הנתונים בית הספר, מזהים של שני צוותים ומספר שלם שהוא מספר הלוחמים. פונקציה זו מדמה מלחמה בין שני הצוותים כך שכל צוות מביא את מספר הלוחמים החזקים ביותר שלו כמספר שהתקבל בארגומנט (אם לאחד הצוותים אין מספיק לוחמים אז כל לוחמיו ישתתפו והוא יהיה בנחיתות מספרית של לוחמים). הצוות המנצח הוא זה שסכום הכוחות של הלוחמים שלו במלחמה הוא הגבוה מבין השניים. הפונקציה מפעילה על המבנה (בית הספר) את המתודה המתאימה team\_fight ומעבירה לה את יתר הארגומנטים כפרמטרים. המתודה בודקת אם המזהים חיובי וקטן מn ואם מספר הלוחמים חיובי ואם לא- מחזירה הודעת שגיאה מתאימה. אחרת, נשתמש בFind של הUnionFind כדי למצוא את שני הצוותים (Find ב-O(log\*n) משוערך כפי שנלמד בכיתה). לאחר שמצאנו את הצוותים ניגש לעצי המונטים שלהם ונפעיל את המתודה get\_fighters\_total\_power (ראה הסבר מטה) שמקבלת את מספר m הלוחמים ומחזירה את סכום כוחותיהם של m הלוחמים החזקים ביותר בעץ או של כל הלוחמים אם מספר המוטנטים בעץ קטן מm. סיבוכיות הזמן של מתודה זו היא O(logk\_team) לכל צוות כלומר O(logk1)+O(logk2)=O(logk). לאחר מכן, נשווה בין שני סכומי הכוחות ונוסיף ניצחון אחד (בשדה num\_of\_wins של הקבוצה) לקבוצה שהסכום בה היה גבוה יותר בO(1). במידה והיה תיקו לא מתווסף ניצחון לאף קבוצה.

**סה"כ סיבוכיות הזמן המשוערכת: O(log\*n+logk) 2\*O(log\*n)+O(logk)+O(1)=.**

הסבר על המתודה get\_fighters\_total\_power:

מתודה זו משתמשת ביכולות עץ הדרגות כדי למצוא את האיבר החוצץ בעץ שהחל ממנו צריכים לסכום ואז מורידה מסכום כוחות הלוחמים של כל העץ (שמור בשורש העץ) את סכום הכוחות של המוטנטים שלא משתתפים בלחימה (כלומר את הערך בצומת של תת העץ שלא רלוונטי) ומחזירה ערך זה. במידה ונדרש מוסיפים אברים שחסרים שהורדו עם תת העץ. סה"כ סיבוכיות הזמן היא O(logk\_team).

1. **GetNumOfWins**- פונקציה זו מקבלת כפרמטרים מצביע למבנה הנתונים בית הספר ומזהה צוות ומצביע ומחזירה במצביע את מספר הנצחונות של הצוות. הפונקציה מפעילה על המבנה (בית הספר) את המתודה המתאימה get\_num\_of\_wins ומעבירה לה כפרמטרים את המזהה והמצביע. המתודה בודקת אם המזהה חיובי וקטן מn ואם המצביע אינו NULL ואם לא- מחזירה הודעת שגיאה מתאימה. אחרת, נשתמש בFind של הUnionFind כדי למצוא את הצוות לפי המזהה שלו (Find ב-O(log\*n) משוערך כפי שנלמד בכיתה). לאחר שמצאנו נחזיר את ערך השדה num\_of\_wins של הצוות במצביע בO(1).

**סה"כ סיבוכיות הזמן המשוערכת: O(log\*n) O(log\*n)+ O(1)=.**

1. **GetStudentTeamLeader** - פונקציה זו מקבלת כפרמטרים מצביע למבנה הנתונים בית הספר ומזהה סטודנט ומצביע ומחזירה במצביע את מזהה הסטודנט החזק ביותר בצוות של הסטודנט שאת המזהה שלו קיבלנו כפרמטר. הפונקציה מפעילה על המבנה (בית הספר) את המתודה המתאימה get\_student\_team\_leader ומעבירה לה כפרמטרים את המזהה והמצביע. המתודה בודקת אם המזהה חיובי ואם המצביע אינו NULL ואם לא- מחזירה הודעת שגיאה מתאימה. אחרת, המתודה משתמשת במתודה get שזה כמו Member של הHashTable ומקבלת את המוטנט שמתאים למזהה המוטנט בO(1) בממוצע על הקלט. לאחר מכן ניגש לשדה במוטנט בו שמרנו את מזהה הצוות שלו בO(1) ובעזרתו נמצא את הצוות עצמו בUnionFind (Find ב-O(log\*n) משוערך כפי שנלמד בכיתה) ואז ניגש לעץ המוטנטים של הצוות וניקח מהמשתנה biggest את המוטנט החזק ביותר בצוות בO(1) ונחזיר את המזהה שלו.

**סה"כ סיבוכיות הזמן המשוערכת בממוצע על הקלט: O(1)+O(1)+O(log\*n)+O(1)=O(log\*n) .**

1. **Quit**- הפונקציה מקבלת מצביע למבנה הנתונים בית הספר ומשחררת את המבנה ע"י קריאה לדיסטרקטור הדיפולטיבי של בית הספר. כחלק מפעילותו הדיסטרקטור קורא לדיסטרקטורים הדיפולטיבים של הHashTable ושל הUnionFind בו אנו משחררים שלושה מערכים באורך n 3\*O(1) וסה"כ עוברים על העצים של כל n הצמתים (אם קיימים) – O(n) במקרה הגרוע, ומשחררים סה"כ k איברים כי לכל מוטנט יש צומת בעץ כלשהו והוא אינו יכול להיות בכמה עצים- סה"כ O(k). בנוסף נעבור על המערך הדינמי של הHashTable שהוא בסדר גודל של k בגלל אופן התחזוקה שלנו ונשחרר אותו בO(1).

**סה"כ סיבוכיות הזמן 3\*O(1)+O(n)+O(k)+O(1)=O(n+k) במקרה הגרוע.**

**בונוס:**

אם בHashTable נשתמש בשיטת **Chain Hashing** בדיוק כפי שנלמד בתרגול אך במקום שכל תא בטבלת הערבול יצביע לרשימה מקושרת, **כל תא בטבלת הערבול יצביע לעץ AVL מאוזן של מוטנטים**. אם איבר x הוכנס למבנה, אז הוא נמצא ברשימה שבומצבעת ע"י T[h(x)]. המקרה הגרוע ביותר הוא כשישנם k מוטנטים בעץ (אם כולם באותו תא) – ננתח את הסיבוכיות של הפעולות במקרה זה:

Member/Delete- גש לתא ה T[h(x)] וחפש את x בעץ שלו. החזר תשובה/מחק בהתאם. במקרה הגרוע ביותר (כפי שפירטנו מעלה) סיבוכיות הזמן היא O(logk) כפי שמחפשים בעץ. אם פונקציית הערבול מפזרת אחיד אז החיפוש בטבלת הערבול יקח O(1) בממוצע על הקלט.

Insert- הוסף את x לעץ המוצבע ע"י T[h(x)]. סיבוכיות הזמן היא O(1) במקרה הגרוע.

**שינויים בפונקציות:**

**בAddStudent**- ההכנסה של המוטנט לטבלת הערבול תקח O(logk) במקרה הגרוע (Member+insert=O(logk)+O(1)=O(logk)) (במקום O(1) בממוצע על הקלט וO(n) במקרה הגרוע כפי שהיה קודם) ולכן **סה"כ סיבוכיות הזמן המשוערכת של הפעולה תהיה כעת: O(logk)+O(log\*n)+O(logk)=O(log\*n+logk) כאשר n מספר הצוותים הראשוני וk מספר המוטנטים בבית הספר.**

**בRemoveStudent-** מחיקה Delete של מוטנט מטבלת הערבול תיקח במקרה הגרוע O(logk) (במקום O(1) בממוצע על הקלט וO(n) במקרה הגרוע כפי שהיה קודם) ולכן **סה"כ סיבוכיות הזמן המשוערכת של הפעולה תהיה כעת: O(logk)+O(log\*n)+O(logk)=O(log\*n+logk) כאשר n מספר הצוותים הראשוני וk מספר המוטנטים בבית הספר.**