# מבני נתונים – 234218

# תרגיל רטוב 2 – חלק יבש

## מגישים:


**עמית שלו , 318369428**

**שון וולפסון , 209157387**

## תיאור מבני הנתונים שמומשו לצורך התרגיל:

1. בתחילה, התבססנו על עץ ה-AVL המאוזן שמימשנו בתרגיל הרטוב הקודם, **ויצרנו עץ דרגות מבוסס AVL**, כאשר כל Node מכיל 2 דרגות:
   1) **כמות הצמתים שיש מתחת לצומת הנוכחי בעץ -** (Degree) – לצורך חישוב ה-Rank של צומת כפי שנלמד בהרצאה.
   2) **סכום הדרגות** (Grade) **של employees מתחת לצומת הנוכחי** – לצורך חישוב סכום הדרגות שמעל הצומת הנוכחי, בדומה לחישוב Rank כפי שנלמד בהרצאה.

   ** נציין כי לכל אורך התרגיל השדות מעודכנים בהתאם לנלמד בתרגול, בעת הכנסה או הסרה של צומת מן העץ. העדכון מתבצע ב- O(1), ולכן הסרה או הכנסה של צומת נשמרת בסיבוכיות זמן של O(log(n)).
2. **Dynamic Hash Table** - הטבלה מומשה ע"י מערך דינאמי, ומכילה:
   1) **Size** - גודל הטבלה הנוכחי.
   2) **Capacity** - תפוסה של המערך (כמה עובדים קיימים כרגע).
   3) **מצביע למערך דינאמי של עצי דרגות** – כל תא יצביע לעץ דרגות AVL.

הטבלה תומכת בפעולות הבאות:

1) **הכנסת** עובד לטבלה ע"י חישוב מודולו של ה-ID שלו עם גודל המערך (id%size). מאפשר פיזור אחיד בממוצע על הקלט – כפי שנלמד בתרגול.
2) **הסרת** עובד מהטבלה.
3) **החזרת** עובד מהטבלה.
4) **הרחבה וצמצום של הטבלה** – אם ה-Capacity שווה ל-Size, יוצרים טבלה חדשה בגודל 2*Size, ומעתיקים את תוכן הטבלה המלאה לטבלה החדשה (לפי פונקציית מודולו של 2*Size). אם ה- Capacity שווה ל – Size/4, אזי באופן דומה צמצמנו את הטבלה לגודל חדש של Size/2. ****<u>אבחנה</u>:** לכל אורך התכנית גודל הטבלה מקיים: $n \leq size \leq 4n$ כאשר $n$ מייצג את מספר העובדים בטבלה.

3. **Union-Find** - מבנה Union-Find המייצג חברות, כאשר כל קבוצה מייצגת חברה אחת, וכל איבר בקבוצה מייצג "חברת בת" (חברות שנרכשו וכן החברה הרוכשת). נדגיש כי שמרנו שהשורש של כל קבוצה יהיה טיפוס קבוע ("**Magic Box**"), ושמרנו כי **הוא זה שיכיל את כל המידע עבור קבוצה** המייצגת חברה אחת. כל "**Magic Box**" מכיל את השדות הבאים:
   1) עץ הפוך של חברות – **כל חברה הכילה שדה נוסף (Degree)** – כך שסכימת השדה לאורך מסלול ה- Find של כל חברת-בת בקבוצה תהיה שווה לערך חברת-הבת כרגע. באופן דומה לחישוב גובה ארגז בבעיית הארגזים כפי שנלמד בתרגול.
   2) **עץ דרגות** מבוסס AVL שמכיל **עובדים בעלי משכורת.** העץ ממוין לפי משכורת (וכן לפי מיון משני של ID).
   3) **Hash Table** של **כלל העובדים** תחת החברה.
   4) מספר **חברות הבת** (כולל חברת "האם").
   5) מספר **העובדים.**
   6) מספר **העובדים ללא משכורת.**

7) **סכום הדרגות של העובדים ללא משכורת.**
8) ה- **Value של החברה** (כלל הקבוצה).

מבנה ה- Union-Find תומך בפעולות הבאות:

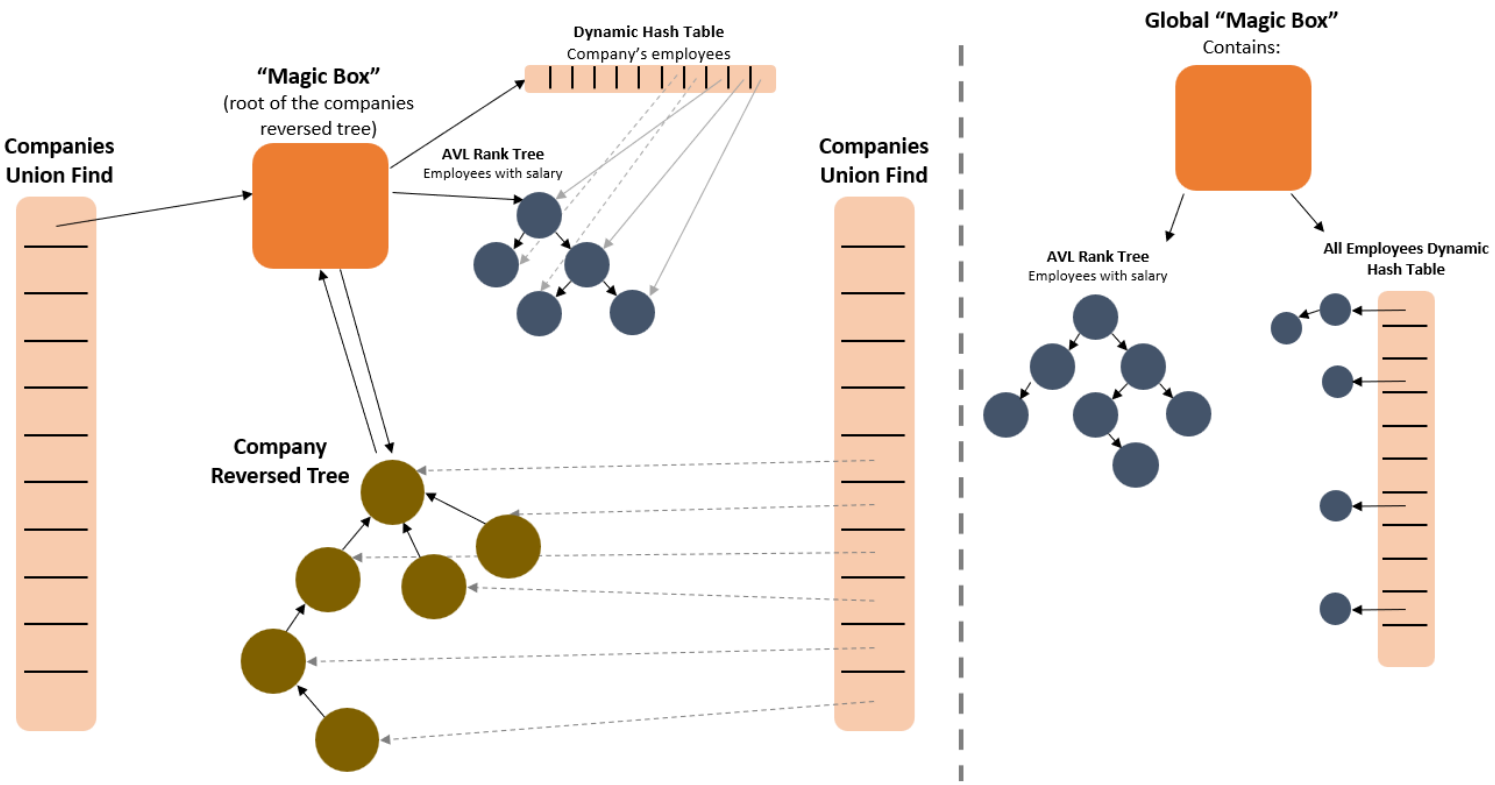
1) **איחוד חברות לפי גודל** כפי שנלמד בהרצאה.
2) **חיפוש חברות עם כיווץ מסלולים** כפי שנלמד בהרצאה.
3) **מציאת ערך (Value) של חברה**, באופן דומה לבעיית הארגזים שנלמדה בתרגול.

על בסיס המימושים שנכתבו לעיל, מבנה הנתונים שלנו יכיל את המידע הבא:

- **מבנה Union-Find** – יכיל את **כל המידע על החברות וחלוקת העובדים הקיימים.**
- **"Magic-Box" Global** - שייצג את קבוצת **"כלל העובדים במבנה".**
- שדה שישמור את **מספר החברות במערכת.**

** נציין כי כל עובד הוא shared_ptr, ולכן למעשה יש instance יחיד לכל אובייקט מסוג עובד, שמוכל במבנה.

**אילוסטרציה של המבנה הכללי:**



**הפעולות הנתמכות:** (כתלות במספר החברות , k ובמספר העובדים הכולל במערכת n)

**Init(k)**

1. אתחול MagicBox גלובאלי (שלבי האתחול נכונים גם לשלב 3):
   a. יצירת מצביע ל- HashTable ריק, בגודל קבוע של 4. האתחול מתבצע ע״י מעבר על טבלה בגודל קבוע, ויצירת מצביעים לעצי דרגות ריקים – O(1).
   b. יצירת מצביע לעץ דרגות AVL ריק – O(1).
   c. השמת null במצביע לעץ חברות הפוך – O(1).
   d. אתחול שדות קבועים נוספים שמתוארים בפירוט ה- UnionFind לעיל – O(1).
2. אתחול שדה של מספר החברות במבנה ל- k – O(1).
3. אתחול מצביע למבנה Union-Find. האתחול כולל את השלבים הבאים:
   a. יצירת שני מערכים בגודל k : **Base** ו- **All_Companies** - סיבוכיות זמן O(1).
   b. מעבר איטרטיבי מ- $1 \leq i \leq k$, כשבכל שלב ניצור מצביע לטיפוס MagicBox חדש (אתחול ב- O(1) כפי שמפורט בשלב 1), וכן מצביע לטיפוס Company חדש עם ערך Id ו-Value ששווה ל-i. בנוסף, בכל איטרציה נבצע:
      i. אתחול התא ה- $i-1$ במערך **Base** להצביע ל-MagicBox
      ii. אתחול התא ה- $i-1$ במערך **All_Companies** להצביע ל-Company שיצרנו.
      iii. השמת מצביע דו כיווני מה-Company ל- MagicBox.

**סה״כ בכל איטרציה מתבצע מספר קבוע של פעולות בסיבוכיות זמן של - O(1). לכן, כל האתחול של המבנה מתבצע ב- O(k).

**סיכום:** סיבוכיות זמן - O(k) , סיבוכיות מקום נוסף – O(k)

**addEmployee ()**

1. אם CompanyID גדול ממספר החברות במבנה, או קטן/שווה ל-0, נחזיר שגיאה.
2. נבדוק אם EmployeeID נמצא ב- HashTable הגלובאלי, המכיל את המידע על כלל העובדים במערכת. אם כן, נחזיר שגיאה. **חיפוש ב- HashTable מתבצע ב- O(1) בממוצע על הקלט**.
3. אחרת, **נוסיף את העובד למבנה לפי השלבים הבאים:**
   a. נאתחל עובד חדש עם ה-ID , Grade ומשכורת 0 – O(1)
   b. נמצא את החברה עם ה- CompanyID שקיבלנו במבנה ה-Union-Find. מציאת החברה ע״י פעולת Find כפי שנלמד בהרצאה, ונחזיר את שורש הקבוצה (MagicBox).
   c. נוסיף את העובד ל-HashTable של ה- MagicBox, וכן ל-HashTable הגלובאלי.

**אם מספר העובדים הקיימים בטבלה (Current Capacity) כעת שווה לגודל הטבלה (Size), נבצע הגדלה של הטבלה פי 2, והעתקת התוכן כפי שנלמד בתרגול והוסבר לעיל.

   d. נבצע השמה של מצביע לMagicBox המייצג את החברה לעובד שיצרנו, ונעדכן את מספר העובדים בחברה, מספר העובדים במבנה, Current Capacity של טבלאות הערבול, וכן נוסיף את ה-Grade של העובד לשדה סכום הדרגות של העובדים ללא משכורת, הן ב- MagicBox הגלובאלי והן בחברה. **כלל העדכונים מתבצעים ב– O(1) .**

כפי שנלמד בתרגול, תחת הנחת הפיזור האחיד פעולות Insert ו-Remove לטבלאות ערבול דינאמיות, מתבצעות ב- **O(1) משוערך בממוצע על הקלט.** בנוסף, מתבצעת פעולת **Find עם קיצור מסלולים.** נראה בהמשך כי הפונקציות שאיתן אנו משערכים את הסיבוכיות, מכילות סדרת פעולות קבועה של Union ו- Find, ולכן כפי שנלמד - הסיבוכיות המשוערכת עימן הינה $O(\log^*(k))$.

**סיכום:** סיבוכיות זמן - $O(\log^*(k))$, משוערך בממוצע על הקלט. סיבוכיות מקום נוסף – $O(n)$ במקרה של הרחבת טבלת הערבול הדינאמית

**removeEmployee ()**

1. נבדוק אם EmployeeID נמצא ב- HashTable הגלובאלי, המכיל את המידע על כלל העובדים במערכת. אם לא, נחזיר שגיאה. **חיפוש ב- HashTable מתבצע ב- O(1) בממוצע על הקלט**.
2. אחרת, נסיר את העובד מן המבנה לפי השלבים הבאים:
   1) ניגש לחברה של העובד ע"י מצביע השמור בו אל שורש הקבוצה (MagicBox).
   2) נסיר את העובד מ-HashTable של ה-MagicBox, וכן מ-HashTable הגלובאלי.

**אם מספר העובדים הקיימים בטבלה (Current Capacity) כעת שווה לרבע מגודל הטבלה (Size), נקטין את הטבלה פי 2, ונעתיק את התוכן בהתאם כפי שנלמד בתרגול וצוין לעיל.

   3) אם העובד בעל משכורת, נסיר אותו מ**עץ דרגות** מבוסס AVL שמכיל **עובדים בעלי משכורת.** ההסרה תתבצע הן מהעץ הגלובאלי, והן מהעץ של החברה - **O(log(n))**.
   4) נעדכן את מספר העובדים בחברה, מספר העובדים במבנה, Current Capacity של טבלאות הערבול, וכן אם לעובד לא הייתה משכורת, נחסיר את ה-Grade של העובד מהשדה של סכום הדרגות של העובדים ללא משכורת, הן ב- MagicBox הגלובאלי והן ב MagicBox המייצג את החברה. **כלל העדכונים מתבצעים בזמן קבוע – O(1) .**

כפי שנלמד בתרגול, תחת הנחת הפיזור האחיד, פעולות Insert ו-Remove לטבלאות ערבול דינאמיות, **מתבצעות ב- O(1) משוערך בממוצע על הקלט.** בנוסף, מתבצעות מספר סופי של פעולות עדכון ערכים ב- O(1), וכן מתבצעת פעולת הסרה מעץ AVL כפי שנלמד בתרגול ב-$O(log(n))$ .

**סיכום:** סיבוכיות זמן $O(log(n))$, משוערך בממוצע על הקלט עם הפונקציה addEmployee.
סיבוכיות מקום נוסף – $O(log(n))$ במקרה של הסרת עובד ברקורסיה מתוך עץ AVL

**acquireCompany()**

1. נבדוק אם מזהי החברות גדולים ממספר החברות שבהן אתחלנו את המבנה, או קטנים/שווים ל-0. אם כן, נחזיר שגיאה. – **O(1)**
2. נבצע פעולת **Find** על 2 מזהי החברות. הפעולה מחזירה את שורש העץ ההפוך (MagicBox). **נבדוק אם ה- MagicBox שהחזרנו זהה.** אם כן, זוהי אותה חברה ונחזיר שגיאה.
3. אחרת, נבצע **איחוד** של החברות באופן הבא:
   **1)** ביצוע **Union** לעץ ההפוך של החברות, לפי הגודל – כפי שנלמד בהרצאה. נציין כי מעכשיו, את **כל המידע המאוחד החדש נשמור בשורש הקבוצה (MagicBox) שהכיל את עץ החברות ההפוך בעל מספר הצמתים הגדול יותר (אליו איחדנו).**

2) בעת הפעולה, **עדכנו ב-O(1)** את שדות **Degree** של שני שורשי העצים ההפוכים לפי השיטה שהוצגה בבעיית הארגזים, כך שסכימת השדות בתהליך ה-Find תתן את ערך החברה הנוכחי. נציין כי בבעיית הארגזים הוספנו את "הגובה" של המבנה הנמוך יותר, ובתרגיל זה הדבר מקביל להוספת ה **Value*Factor** של החברה הנרכשת (הערך Value של החברה הנרכשת שמור ב- MagicBox שמייצג אותה).

3) **איחוד עצי AVL** של העובדים עם משכורת בדומה לנלמד בתרגול – מעבר InOrder על שני העצים, הוצאתם לשני מערכים ממויינים, מיזוג המערכים למערך ממויין אחד, יצירת עץ ריק כמעט שלם והכנסת הערכים אליו - $\boldsymbol{O(n_{AcquirerID})} + \boldsymbol{O(n_{TargetID})}$.

4) נהרוס את העצים הישנים, ונבצע השמה של העץ המאוחד ל-MagicBox אליו איחדנו.

5) **עדכון פרמטרים ב- O(1)** הכוללים את סכום דרגות העובדים ללא משכורת, מספר העובדים ללא משכורת ומספר העובדים בחברה. בנוסף, עדכנו את שדה "ערך החברה הנוכחי" השמור ב- MagicBox שאליו איחדנו:

   i. אם איחדנו את החברה הנרכשת לחברה הרוכשת, הוספנו $currentValue += purchased \rightarrow currentValue \cdot Factor$

   ii. אחרת: $currentValue = purchased \rightarrow currentValue \cdot Factor + acquirer \rightarrow currentValue$

6) **איחוד טבלאות הערבול** של עובדי כל חברה -יצירת טבלת ערבול דינאמית חדשה בגודל השווה לסכום הגדלים של 2 **הטבלאות שאנו מאחדים, עם פונקציית ערבול מתאימה.** נבצע מעבר על 2 טבלאות הערבול באופן איטרטיבי, ונכניס כל עובד לטבלה החדשה.

נציין כי מאבחנה שנכתבה בתחילת הגיליון, גודל שתי הטבלאות שאנו מאחדים נשמר תחת חסם של $\boldsymbol{n < size \leq 4n}$ כאשר $n$ מייצג את מספר העובדים בטבלה. לכן, בהנחת הפיזור האחיד, כל פעולת **המעבר האיטרטיבי** על הטבלאות שאנו רוצים לאחד מתבצע ב- $\boldsymbol{O(n_{Acquire})} + \boldsymbol{O(n_{target})}$ **בממוצע על הקלט.** נציין כי גודל הטבלה החדשה שווה לסכום הגדלים, ולכן לאחר ההכנסה של העובדים לא יתבצע צמצום או הגדלה של הטבלה החדשה. לכן, בסה"כ **הכנסת כל עובד יחיד לטבלה, תתבצע ב- *O(1)* בממוצע על הקלט.**

בנוסף, מתבצעות **פעולות Find עם קיצור מסלולים ו-Union לפי גודל.** נראה בהמשך כי הפונקציות שאיתן אנו משערכים את הסיבוכיות, מכילות גם סדרת פעולות קבועה של Union ו-Find, ולכן כפי שנלמד הסיבוכיות המשוערכת עימן הינה $\boldsymbol{O(\log^*(k))}$.

**סיכום:** סיבוכיות זמן $O(\log^*(k) + n_{AcquirerId} + n_{TargetID})$, משוערך, בממוצע על הקלט.

סיבוכיות מקום נוסף – $O(n)$ עבור עץ העובדים החדש ויצירת טבלת ערבול מאוחדת חדשה.

**employeeSalaryIncrease()**

1. נבדוק אם EmployeeID נמצא ב- HashTable הגלובאלי, המכיל את המידע על כלל העובדים במערכת. אם לא, נחזיר שגיאה. **חיפוש ב- HashTable מתבצע ב- O(1) בממוצע על הקלט.**
2. אחרת, ניגש לחברה של העובד ע"י מצביע השמור בו אל שורש הקבוצה (MagicBox) ונבצע את הפעולות הבאות:

1) אם העובד היה ללא משכורת, נעלה לו את השדה של המשכורת, ונוסיף אותו לעץ הדרגות שמכיל עובדים בעלי משכורת, הן בעץ העובדים עם משכורת הגלובאלי, והן בMagicBox המייצג את החברה. נחסיר את דרגת העובד הנוכחי משדה סכום הדרגות של העובדים ללא משכורת, ונחסיר אחד משדה מספר העובדים ללא משכורת בחברה ובמבנה הכללי.
2) אחרת, לעובד הייתה משכורת. לכן, הסרנו אותו מן העצים, הוספנו לו משכורת, והכנסנו אותו מחדש לעצים. זאת על מנת לשמור על האיזון, שכן העצים ממוינים לפי משכורת (וכן לפי מיון משני של ID). **הוספה והסרה מעץ AVL – O(log(n)) כפי שנלמד בתרגול.**

**סיכום:** סיבוכיות זמן O( $log(n)$) בממוצע על הקלט. סיבוכיות מקום נוסף – O($log(n)$) עבור רקורסיות הכנסת והסרת עובדים מן העצים.

**promoteEmployee ()**

1. נבדוק אם EmployeeID נמצא ב- HashTable הגלובאלי, המכיל את המידע על כלל העובדים במערכת. אם לא, נחזיר שגיאה. **חיפוש ב- HashTable מתבצע ב- O(1) בממוצע על הקלט**.
2. אחרת, ניגש לחברה של העובד ע"י מצביע השמור בו אל שורש הקבוצה (MagicBox) ונבצע את הפעולות הבאות:

1) אם העובד היה ללא משכורת, נעלה את ה-Grade שלו ונעדכן את שדה סכום הדרגות של העובדים ללא משכורת, הן ב- MagicBox המייצג את החברה והן בשדה הגלובאלי.
2) אחרת, העובד היה קיים בעץ העובדים בעלי שכר. כדי לשמור על שמורת עץ הדרגות, נסיר אותו מהעץ שמכיל עובדים בעלי משכורת, הן בעץ העובדים עם משכורת הגלובאלי, והן ב MagicBox המייצג את החברה. לאחר ההסרה נוסיף לו את הדרגה כנדרש, ונוסיף אותו לעצים מחדש. **הוספה והסרה של מעץ AVL – O(log(n)) כפי שנלמד בתרגול.**

**סיכום:** סיבוכיות זמן O( $log(n)$) בממוצע על הקלט. סיבוכיות מקום נוסף – O($log(n)$) עבור רקורסיות הכנסת והסרת עובדים מן העצים.

**sumOfBumpGradeBetweenTopWorkersByGroup()**

1. אם CompanyID גדול ממספר החברות במבנה, או קטן מ-0, נחזיר שגיאה.
2. אחרת אם $CompanyID \neq 0$ נבצע פעולת **Find** למציאת החברה ונחזיר את השורש של הקבוצה (MagicBox).
3. נלך לעץ הדרגות בחברה המכיל עובדים בעלי משכורת. העץ ממוין לפי משכורת העובדים, ולפי מיון משני של ה- ID של העובדים במקרה של-2 עובדים בעלי משכורת זהה. נרצה להגיע לצומת בעץ שנמצא ב-Rank ששווה למספר העובדים בעץ (Tree->getSize()) פחות m. אם הערך (אינדקס) שנקבל קטן מ-0, אין מספיק עובדים עם משכורת בחברה, ולכן נחזיר שגיאה.
4. כעת, אם נוכל להגיע לצומת זו ולחשב את סכום הדרגות שמעליה, נקבל את סכום הדרגות של m העובדים בעלי המשכורת הגבוהה ביותר בחברה. מאחר וכל צומת של עובד מכילה את סכום כלל הדרגות של כל הצמתים שמתחתיה, בעת ירידה לצומת לפי ה- **Rank הדרוש**, כפי שנלמד בתרגול, נוכל להגיע לערך הרצוי באופן הבא:

1) נאתחל משתנה זמני grades_above_rank, **ונרד בעץ עד לצומת בדרגה הדרושה.**
2) אם ירדנו ימינה, נמשיך בחיפוש בהתאם לאלגוריתם Select(rank).

3) אם ירדנו שמאלה, נעלה את grades_above_rank בסכום כל הדרגות הגדולות מהבן הימני של הצומת שממנו ירדנו (כולל), ונוסיף את הדרגה של הצומת עצמה.
4) כאשר הגענו לצומת בדרגה הדרושה, החזרנו את grades_above_rank ועוד סכום כל הדרגות הגדולות מהבן הימני שלו (כולל).

5. אם $CompanyID == 0$, אז נבצע את האלגוריתם בעץ העובדים הגלובאלי.

**סיכום הפעולות – חיפוש בעץ AVL וחישוב דרגה** – O(log(n)). בנוסף, מתבצעת פעולת **Find** עם קיצור מסלולים. ראינו ונראה גם בהמשך כי הפונקציות שאיתן אנו משערכים את הסיבוכיות, מכילות סדרת פעולות קבועה של Union ו- Find, ולכן כפי שנלמד הסיבוכיות המשוערכת עימן הינה $O(\log^*(k))$.

**סיכום:** סיבוכיות זמן $O(\log^*(k) + log(n))$ משוערך. סיבוכיות מקום נוסף – $O(1)$

**averageBumpGradeBetweenSalaryByGroup()**

1. אם CompanyID גדול ממספר החברות במבנה, או קטן מ-0, נחזיר שגיאה.
2. אחרת אם $CompanyID \neq 0$ נבצע פעולת **Find** למציאת החברה ונחזיר את השורש של הקבוצה (MagicBox).
3. לצורך תמיכה בפעולה זו בסיבוכיות הנדרשת **נמצא את הגבולות בעץ העובדים עם המשכורות בחברה.** נחפש בעץ העובדים את הצומת הכי קרוב למינימום הנדרש (מלמעלה), ואת הצומת הכי קרובה למקסימום (מלמטה). עבור הצומת הכי-קרוב-למינימום, נשמור משתנה temp שיחזיק בכל פעם את הצומת האחרון שממנו הלכנו ימינה (כלומר הצומת האחרון שיוחזק יהיה ההכי קטן שגדול או שווה מן המינימום). באופן אנלוגי עבור הצומת הכי קרוב למקסימום (עבור צעדים שמאלה) – סה"כ פעולת **החיפוש בעץ AVL** של העובדים מתבצע ב- $\boldsymbol{O[log(n)]}$.
4. אם אחד הגבולות לא קיים (האלגוריתם מחזיר nullptr), אזי נבדוק: אם lowerSalary==0, נחזיר את סכום כלל הדרגות של עובדים ללא משכורת בחברה, חלקי מספר העובדים ללא משכורת בחברה. אחרת, אם lowerSalary>0 נחזיר שגיאה.
5. לאחר מציאת העובדים שמהווים את החסמים בעץ, נמצא את הדרגה של כל אחד בעץ, כפי שנלמד בתרגול. האלגוריתם מתבצע ב- $\boldsymbol{O[log(n)]}$. נסמן את דרגת החסם העליון ב- $Rank_{HIGH}$ ואת דרגת החסם התחתון ב- $Rank_{LOW}$.
6. כעת, ניעזר בפונקציה sumOfBumpGradeBetweenTopWorkersByGroup, כדי למצוא את סכום הדרגות מעל הדרגה של הגבול העליון (לא כולל), נסמנו ב - $S_{HIGH}$ וכן את סכום הדרגות מעל הגבול התחתון (כולל), נסמנו ב- $S_{LOW}$.
7. אם ה- lowerSalary > 0, הממוצע הינו $(S_{HIGH} - S_{LOW})/(Rank_{HIGH} - Rank_{Low} + 1)$.
8. אם lowerSalary=0 , נוסיף לסכום במונה את סכום כלל הדרגות של עובדים ללא משכורת בחברה, ולמכנה נוסיף את מספר העובדים ללא משכורת בחברה.
9. אם $CompanyID == 0$ נבצע את אותו אלגוריתם במידע הגלובאלי.

**סיכום הפעולות – חיפוש בעץ AVL** למציאת גבולות ודרגות – O(log(n)). קריאה ל- **sumOfBumpGradeBetweenTopWorkersByGroup** - $O(\log^*(k) + log(n))$.

בנוסף, מתבצעת פעולת **Find** עם קיצור מסלולים. ראינו ונראה גם בהמשך כי הפונקציות שאיתן אנו משערכים את הסיבוכיות, מכילות גם סדרת פעולות קבועה של Union ו- Find, ולכן כפי שנלמד הסיבוכיות המשוערכת עימן הינה $O(\log^*(k))$.

**סיכום:** סיבוכיות זמן $O(\log^*(k) + log(n))$ משוערך. סיבוכיות מקום נוסף – $O(1)$

**companyValue()**

1. אם CompanyID גדול ממספר החברות במבנה, או קטן/שווה ל-0, נחזיר שגיאה.
2. נחשב את דרגת החברה באופן זהה לחישוב גובה הארגזים כפי שנלמד בתרגול על בעיית הארגזים. לשם כך, ביצענו **2 פעולות Find**. הפעולה הראשונה כדי להבטיח שהצומת של החברה הוא השורש של העץ ההפוך, או לכל היותר במרחק אחד מן השורש. כפי שראינו בתרגול, בעת ביצוע ה-**Find**, מתעדכן הערך Degree השמור בצמתי החברות, המשמש לחישוב ה-Value.
3. כעת, אם הצומת הוא השורש של העץ ההפוך, החזרנו את השדה Degree השמור בצומת. אחרת, בעקבות פעולת ה-**Find** הראשונה, נוסיף לשדה Degree גם את ה-Degree של השורש.

סה"כ מתבצעת פעולת **Find** עם קיצור מסלולים. ראינו עד כה כי הפונקציות שאיתן אנו משערכים את הסיבוכיות, מכילות גם סדרת פעולות קבועה של Union ו- Find, ולכן כפי שנלמד הסיבוכיות המשוערכת עימן הינה $O(\log^*(k))$.

**סיכום:** סיבוכיות זמן $O(\log^*(k))$ משוערך. סיבוכיות מקום נוסף – $O(1)$

**Quit()**

1. **הריסת טבלאות הערבול** ע"י מעבר איטרטיבי על גודל הטבלאות והריסת הצמתים המוכלים בהם. **מאחר וגודל כל טבלה הוא בחסם של $O(n_{Company})$, סה"כ נקבל**

$$O(n_{Company_1}) + O(n_{Company_2}) \dots + O(n_{Company_k}) = O(n)$$

**כי מתקיים -** $n_{Company_1} + n_{Company_2} + \cdots n_{Company_k} = n$

2. **באופן דומה, הריסת עצי דרגות** העובדים ושחרור כל הצמתים– **O(n)**.
3. **הריסת מבנה ה- Union-Find** – שחרור כל ההצבעות לכל ה- MagicBoxes ו- Companies שבמבנה. **סה"כ $O(k)$ כמספר החברות הראשוני.**

**סיכום**: סיבוכיות זמן – $O(k+n)$, מקום נוסף – $O(1)$.

לסיכום, נדגיש כי כלל הפונקציות $addEmployee$ , $acquireCompany$ $averageBumpGradeBetweenSalaryByGroup$ $sumOfBumpGradeBetweenTopWorkersByGroup$ כללו סדרה סופית של פעולות Union ו- Find, ולכן לפי הנלמד, הסיבוכיות המשוערכת של פעולות אלה הינה $O(\log^*(k))$.

**סה"כ סיבוכיות מקום:** מפירוט סיבוכיות המקום של כל פונקציה, וכן שמירת עותקים יחידים של חברות ועובדים במערכת, סיבוכיות המקום הכוללת היא $O(k+n)$.