מבני נתונים – 234218

תרגיל רטוב 1 – חלק יבש

מגישים:

עמית שלו , 318369428

שון וולפסון , 209157387

תיאור מבני הנתונים:

1. בתחילה, מימשנו עץ AVL גנרי אשר מכיל:

- מחלקה בשם **Node**, אשר מייצגת צומת בעץ, ומכילה את המידע הרלוונטי, הגובה של הצומת בעץ, ומצביעים לבן השמאלי ולבן הימני. העץ מחזיק בתוכו מצביע ל-Node הראשון (ראש העץ).
- פונקציית השוואה שלפיה משווים בין שני צמתים בעץ (Functor).
- מספר הצמתים בעץ.

2. העץ תומך בפעולות הבאות – n מציין את מספר הצמתים בעץ:

- כלל הפעולות שנלמדו בכיתה: הכנסת, הסרת ומציאת איבר ב- $O[\log(n)]$. כולל גלגולים בעת הצורך ב- $O(1)$.
- GetMaximumNode – החזרת הצומת המקסימלי (נלך ימינה בעץ עד שנגיע לצומת ללא בן ימני) - $O[\log(n)]$ לפי עומק העץ.
- מיזוג שני עצים, כפי שנלמד בתרגול על מיזוג שני עצים בינאריים ב- $O(n)$.
- InOrder, PostOrder, PreOrder כפי שנלמד בתרגול ב- $O(n)$.
- מעברים נוספים:

i. InOrder על כמות צמתים מוגבלת מראש - ישמש ספציפית ל-GetHighestEarnerInEachCompany.

ii. InOrder שיבוצע בין שני צמתים בעץ (ללא מעבר על כל הצמתים) - ישמש ספציפית ל- GetNumEmployeesMatching.

iii. ReverseInOrder - תשמש ספציפית ל- GetAllEmployeesBySalary.

2. במסגרת התרגיל, <u>מימשנו את מבנה הנתונים שמכיל את המידע הבא</u> (כל עץ הינו AVL):

- מצביע ל-**עץ העובדים לפי מזהה:** מכיל את כל העובדים במערכת, הממוינים לפי ID.
- מצביע ל-**עץ העובדים לפי משכורות:** מכיל את כל העובדים במערכת, הממוינים לפי המשכורת שלהם.
- מצביע ל-**עץ החברות לפי מזהה:** מכיל את כלל החברות במערכת, מסודרות לפי ה-ID.
- מצביע ל-**עץ החברות בעלות עובדים:** מכיל את כלל החברות במערכת שיש להן לפחות עובד אחד.
- בנוסף, שמרנו שני שדות המכילים את מזהה העובד בעל המשכורת הגבוהה ביותר במערכת, וכן את משכורתו.
- <u>הבחנה: כי כל צומת והמידע שמוכל בה הם shared_ptr, ולכן למעשה יש instance יחיד לכל אובייקט (חברה או עובד), שמוכל בעצים הרלוונטיים.</u>

3. המחלקות שבהן נעזרנו במימוש מבנה הנתונים:

- **Company:** לכל צומת שמייצג חברה שמורים המזהים הנדרשים בעת יצירת חברה, וכן שני שדות המכילים את מזהה העובד בעל המשכורת הגבוהה ביותר בחברה ואת משכורתו. בנוסף, ישנם עוד שני מצביעים לשני עצים נוספים:
  i. **עץ עובדים בחברה לפי מזהה** - מכיל את העובדים תחת החברה, מסודרים לפי ID.
  ii. **עץ עובדים בחברה לפי משכורת** - מסודרים לפי המשכורת.
- **Employee**: לכל צומת שמורים המזהים הנדרשים בעת יצירת עובד. בנוסף, ישנו מצביע לחברה בה הוא עובד (המצביע אינו shared_ptr, כדי להימנע ממעגל הצבעות).

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

<u>פעולות כלליות נוספות על הנדרש</u>:

1. <u>עדכון העובד המשתכר ביותר</u>:

- במבנה הנתונים הכללי – אם עץ כלל העובדים ריק, נשים 0 בשדות המזהה והמשכורת של העובד המשתכר ביותר. אחרת, נשיג את הצומת המקסימלי בעץ כלל העובדים הממוין לפי המשכורת - $\boldsymbol{O[log(n)]}$.
- בחברה – אותו רעיון רק על עץ העובדים לפי משכורת בחברה - $\boldsymbol{O[log(n_{Company})]}$.

2. <u>הריסת עץ</u>:

- מעבר PostOrder רקורסיבי על העץ, והריסת הצמתים. ההריסה מתבצעת קודם על הבנים, ואז על האב שלהם – לכן מתבצעת פעולה ב- $O(1)$ על n צמתים – ולכן <u>סיבוכיות הזמן - $\boldsymbol{O(n)}$, ומקום נוסף מהרקורסיה כעומק העץ – $\boldsymbol{O[log(n)]}$.</u>

3. <u>הוספה או הסרה של צומת בעץ</u>:

- ההסרה וההוספה מתבצעות רקורסיבית כפי שנלמד בהרצאה - <u>סיבוכיות זמן - $\boldsymbol{O[log(n)]}$, ומקום נוסף מהרקורסיה כעומק העץ – $\boldsymbol{O[log(n)]}$.</u>

<u>הפעולות הנדרשות:</u>

<u>Init():</u>

1. אתחלנו שלושה Functors – השוואה לפי שני מזהי עובדים, השוואה לפי שתי משכורות עובדים והשוואה לפי שני מזהי חברות - $\boldsymbol{O(1)}$.
2. בעזרתם, אתחלנו מבנה נתונים חדש – הקצנו על ה- heap ארבעה מצביעים לעצי AVL ריקים (כפי שמפורט במבנה הנתונים), וכן את השדות של העובד המשתכר ביותר אתחלנו ל-0. אתחול עצים ריקים מתבצע ב- $\boldsymbol{O(1)}$ - השמה של nullptr בראש העץ.

<u>**סיכום:** סיבוכיות זמן - $O(1)$, סיבוכיות מקום נוסף - $O(1)$</u>

<u>AddCompany():</u>

1. נבדוק האם החברה קיימת במערכת, ע"י חיפוש בעץ כלל החברות. אם החברה קיימת, נחזיר שגיאה. חיפוש החברה בעץ כפי שנלמד בהרצאה - $\boldsymbol{O[log(k)]}$.
2. אחרת, אתחלנו מחלקת חברה חדשה עם הפרמטרים שקיבלנו - $\boldsymbol{O(1)}$, והכנסנו את המחלקה לעץ כלל החברות במערכת כפי שנלמד בהרצאה - $\boldsymbol{O[log(k)]}$.

סיכום: סיבוכיות זמן - $O[log(k)]$, סיבוכיות מקום נוסף מהרקורסיה - $O[log(k)]$

AddEmployee():

1. נבדוק האם העובד קיים במערכת, ע"י חיפוש בעץ כלל העובדים במערכת. אם העובד קיים, נחזיר שגיאה. חיפוש העובד בעץ כפי שנלמד בהרצאה - $\boldsymbol{O[log(n)]}$.
2. נחפש את החברה הדרושה, ע"י חיפוש בעץ כלל החברות במערכת. אם החברה לא קיימת נחזיר שגיאה. חיפוש החברה - $\boldsymbol{O[log(k)]}$. נציין כי אם החיפוש הצליח, החזרנו את החברה עצמה.
3. נבדוק אם לחברה יש עובדים (באמצעות שדה של מספר העובדים בחברה) - $\boldsymbol{O(1)}$. במידה ואין, נכניס אותה כעת לעץ החברות עם העובדים. ההכנסה מתבצעת כפי שנלמד בהרצאה, ומאחר ולעץ זה מספר צמתים $k \geq$, הפעולה מתבצעת ב- $\boldsymbol{O[log(k)]}$.
4. אתחלנו מחלקה של עובד חדש, עם הפרמטרים שקיבלנו, וביצענו השמה של המצביע לחברה הרלוונטית - $\boldsymbol{O(1)}$. הוספנו את העובד החדש לארבעה עצים – שני עצים של כלל העובדים, לפי ID ולפי משכורות - $\boldsymbol{O[log(n)]}$, וגם לשני העצים (ID ומשכורות) הנמצאים תחת החברה הרלוונטית - $\boldsymbol{O[log(k)]}$. ההכנסה לחברה כללה עדכון של שדה כמות העובדים בחברה.
5. לבסוף, אם לעובד משכורת גדולה יותר מהשדות השמורים במבנה הנתונים הראשי ובחברה הרלוונטית, עדכנו את העובד המשתכר לפי המתואר "בפעולות כלליות נוספות" לעיל. $\boldsymbol{O[log(n)] + O\left[log\left(n_{Company}\right)\right] = O[log(n)]}$.

סיכום: סיבוכיות זמן - $O[log(n) + log(k)]$, מקום נוסף מהרקורסיה - $O[log(n)]$

RemoveCompany():

1. נמצא את החברה במערכת, ע"י חיפוש בעץ כלל החברות. אם החברה לא קיימת נחזיר שגיאה. חיפוש החברה כפי שנלמד בהרצאה - $\boldsymbol{O[log(k)]}$.
2. אחרת, החברה קיימת. אם יש לה עובדים (ע"י בדיקת שדה מספר העובדים), נחזיר שגיאה.
3. נסיר את החברה מעץ כלל החברות, כפי שלמדנו בהרצאה $\boldsymbol{O[log(k)]}$.

סיכום: סיבוכיות זמן - $O[log(k)]$, מקום נוסף מהרקורסיה - $O[log(k)]$

RemoveEmployee():

1. נבדוק האם העובד קיים במערכת, ע"י חיפוש בעץ כלל העובדים במערכת. אם העובד לא קיים, נחזיר שגיאה - $\boldsymbol{O[log(n)]}$.

2. אחרת, מצאנו את העובד הנדרש. נגיע לצומת החברה שבה הוא עובד ע"י המצביע השמור אצל העובד – $\boldsymbol{O(1)}$. לאחר מכן, נוכל לבצע ישירות הסרה של העובד מ-4 עצים, כפי שנלמד בהרצאה - נסיר את העובד מ-2 העצים השמורים במבנה הראשי (ID ומשכורות) - $\boldsymbol{O[log(n)]}$, וכן מ-2 העצים המתאימים תחת החברה בה עבד - $\boldsymbol{O[log(n_{company})]}$.
ברור כי $\boldsymbol{n_{company} \leq n}$, ולכן סה"כ, פעולת ההסרה התבצעה סה"כ ב- $\boldsymbol{O[log(n)]}$.
3. עדכנו את כמות העובדים בחברה, ואם הוא היה העובד האחרון, הסרנו את החברה מעץ החברות עם עובדים. <u>הבחנה:</u> מספר הצמתים בעץ החברות עם העובדים, הוא לכל היותר מספר העובדים שבמערכת, שמתקבל אם לכל חברה בעץ יש עובד יחיד. לכן הוצאת החברה מעץ זה, תתבצע גם בחסם של $\boldsymbol{O[log(n)]}$.
4. לבסוף, אם הסרנו את העובד המשתכר ביותר בחברה / במבנה הנתונים הכללי, עדכנו את העובד המשתכר לפי המתואר "בפעולות כלליות נוספות" לעיל -

$$\boldsymbol{O[log(n)] + O[log(n_{Company})] = O[log(n)]}$$.

<u>**סיכום:** סיבוכיות זמן - $O[log(n)]$, מקום נוסף מהרקורסיה - $O[log(n)]$</u>

<u>GetCompanyInfo():</u>

1. נמצא את החברה במערכת, ע"י חיפוש בעץ כלל החברות. אם החברה לא קיימת נחזיר שגיאה. חיפוש החברה כפי שנלמד בהרצאה - $\boldsymbol{O[log(k)]}$.
2. אחרת, החברה קיימת ונשים את הערכים השמורים בחברה במצביעים שהועברו - $\boldsymbol{O(1)}$

<u>**סיכום:** סיבוכיות זמן - $O[log(k)]$, סיבוכיות מקום נוסף - $O(1)$</u>

<u>GetEmployeeInfo():</u>

1. נבדוק האם העובד קיים במערכת, ע"י חיפוש בעץ כלל העובדים במערכת. אם העובד לא קיים, נחזיר שגיאה - $\boldsymbol{O[log(n)]}$.
2. אחרת, העובד קיים. נשים את הערכים השמורים בעובד במצביעים שהועברו - $\boldsymbol{O(1)}$

<u>**סיכום:** סיבוכיות זמן - $O[log(n)]$, סיבוכיות מקום נוסף - $O(1)$</u>

<u>IncreaseCompanyValue():</u>

1. נמצא את החברה במערכת, ע"י חיפוש בעץ כלל החברות. אם החברה לא קיימת נחזיר שגיאה. חיפוש החברה כפי שנלמד בהרצאה - $\boldsymbol{O[log(k)]}$.
2. אחרת, מצאנו את החברה. נבצע עדכון של שדה Value במחלקת החברה - $\boldsymbol{O(1)}$

<u>**סיכום:** סיבוכיות זמן - $O[log(k)]$, סיבוכיות מקום נוסף - $O(1)$</u>

<u>PrompoteEmployee():</u>

1. נבדוק האם העובד קיים במערכת, ע"י חיפוש בעץ כלל העובדים במערכת. אם העובד לא קיים, נחזיר שגיאה - $\boldsymbol{O[log(n)]}$.

2. אחרת, מצאנו את העובד. נגיע לחברה שבה הוא עובד ע"י המצביע השמור בעובד - $\boldsymbol{O(1)}$.
3. אתחלנו עובד חדש עם אותם הפרמטרים של העובד הנוכחי, והוספנו לו את המשכורת הנדרשת - $\boldsymbol{O(1)}$.
4. הסרנו את העובד "הישן" משני העצים הנמצאים במבנה הנתונים הראשי (ID ומשכורת) - $\boldsymbol{O[log(n)]}$, ולאחר מכן הסרנו אותו מהחברה בה הוא נמצא - $\boldsymbol{O[log(n_{Company})]}$.
5. לבסוף, הוספנו את העובד החדש עם המשכורת המעודכנת לארבעת העצים - $\boldsymbol{O[log(n)] + O[log(n_{Company})] = O[log(n)]}$.
6. בדקנו ועדכנו את העובד המשתכר ביותר במבנה הנתונים הכללי ובחברה - $\boldsymbol{O[log(n)] + O[log(n_{Company})] = O[log(n)]}$.

סיכום: סיבוכיות זמן - $O[log(n)]$, סיבוכיות מקום נוסף - $O(1)$

HireEmployee():

1. נבדוק האם העובד קיים במערכת, ע"י חיפוש בעץ כלל העובדים במערכת. אם העובד לא קיים, נחזיר שגיאה - $\boldsymbol{O[log(n)]}$.
2. נמצא את החברה שאליה צריך להעביר את העובד, ע"י חיפוש בעץ כלל החברות. אם החברה לא קיימת נחזיר שגיאה. חיפוש החברה כפי שנלמד - $\boldsymbol{O[log(k)]}$.
3. נסיר את העובד מכלל העצים במערכת $\boldsymbol{O[log(n)]}$, ונוסיף אותו מחדש לחברה החדשה $\boldsymbol{O[log(n)] + O[log(k)]}$.
4. אם זהו העובד הראשון בחברה החדשה, או העובד האחרון בחברה שממנה העברנו אותו, נסיר או נוסיף בהתאמה את החברה מעץ החברות בעלות עובדים. נשים לב כי מספר החברות שמחזיקות עובדים $n \geq$, והמספר שווה ל-n אם בכל חברה יש עובד אחד. לכן סה"כ - $\boldsymbol{O[log(n)]}$.
5. נעדכן את העובד המשתכר ביותר בחברה שאליה העברנו - $\boldsymbol{O[log(n_{Company})] = O[log(n)]}$.

סיכום: סיבוכיות זמן - $O[\log(n) + log(k)]$, מקום נוסף - $O[log(n)]$

AcquireCompany():

1. נבדוק אם החברות קיימות במערכת, ע"י חיפוש בעץ כלל החברות. אם אחת החברות לא קיימת נחזיר שגיאה - $\boldsymbol{O[log(k)]}$.
2. בדיקת היתכנות הרכישה בעזרת שדות ה- Value בחברות - $\boldsymbol{O(1)}$.
3. נאחד את שני זוגות העצים של החברות (ID ומשכורות), כפי שנלמד בתרגול:
   - מעבר InOrder על שני העצים והוצאתם למערכים ממוינים. מעבר InOrder מתבצע ב - $\boldsymbol{O(n_{Acquire}) + O(n_{target})}$ בהתאמה לכל עץ.
   - מיזוג המערכים למערך אחד ממוין בגודל המתאים – $\boldsymbol{O(n_{Acquire} + n_{target})}$.

- יצירת עץ ריק כמעט שלם, והכנסת הערכים מהמערך הממוין המאוחד, בעזרת InOrder - $\boldsymbol{O(n_{Acquire}) + O(n_{target})}$.

4. נהרוס את העצים הישנים של שתי החברות – כפי שכתוב לעיל בפעולות כלליות נוספות - $\boldsymbol{O[log(n_{Acquire})] + O[log(n_{target})]}$. נבצע השמה של העץ החדש לחברה הרוכשת - $\boldsymbol{O(1)}$, ונמחק את החברה שנרכשה - $\boldsymbol{O[log(k)]}$.

סיכום: סיבוכיות זמן - $O[\log(k) + n_{Acquire} + n_{target}]$, מקום נוסף מרקורסיית InOrder ויצירת המערכים - $\boldsymbol{O(n_{Acquire} + n_{target})}$.

GetHighestEarner():

אם $ID > 0$:

1. נבדוק אם החברה קיימת במערכת, ע"י חיפוש בעץ כלל החברות. אם אחת החברות לא קיימת נחזיר שגיאה - $\boldsymbol{O[log(k)]}$.
2. נחזיר את השדות המתאימים השמורים במחלקת החברה - $\boldsymbol{O(1)}$.

אם $ID < 0$:

3. נחזיר את השדה המתאים שנשמר במבנה הנתונים הכללי - $\boldsymbol{O(1)}$.

סיכום: $ID > 0$: סיבוכיות זמן – $O[log(k)]$, מקום נוסף $O(1)$.

$ID < 0$: סיבוכיות זמן – $O(1)$, מקום נוסף $O(1)$.

GetAllEmployeesBySalary():

אם $ID > 0$:

4. נבדוק אם החברה קיימת במערכת, ע"י חיפוש בעץ כלל החברות. אם היא לא קיימת נחזיר שגיאה - $\boldsymbol{O[log(k)]}$.
5. נבצע מעבר ReverseInOrder, ונחזיר את העובדים שבעץ במערך – סיבוכיות זמן ומקום נוסף $\boldsymbol{O(n_{company})}$.
6. נקצה מערך חדש כמספר העובדים בעץ, ונכניס לתוכו את המזהים של העובדים. סה"כ עוברים על מערך בגודל $n_{company}$, ומבצעים פעולה ב- $O(1)$ על כל עובד. כלומר סיבוכיות זמן ומקום נוסף - $\boldsymbol{O(n_{company})}$

אם $ID < 0$:

7. נבצע את אותו האלגוריתם, רק על עץ כלל העובדים במערכת. כלומר סיבוכיות זמן ומקום נוסף - $\boldsymbol{O(n)}$.

סיכום: $ID > 0$: סיבוכיות זמן – $O[log(k)+n_{company}]$, מקום נוסף $O(n_{company})$.

$ID < 0$: סיבוכיות זמן – $O(n)$, מקום נוסף $O(n)$.

GetHighestEarnerInEachCompany():

1. נקצה מערך חדש כגודל מספר החברות הנדרש (NumOfCompanies).
2. לצורך תמיכה בפעולה זו בסיבוכיות הנדרשת, נבצע מעבר Inorder על עץ החברות **עם העובדים**, בעזרת מונה יורד לפי מספר החברות הנדרש. לאחר מעבר על מספר החברות הנדרש, נחזור מהרקורסיה ונחזיר מערך בגודל מספר החברות שעברנו עליהן.
   - סיבוכיות זמן ההגעה לצומת הקטנה ביותר – כגובה העץ - $O[log(k)]$.
   - מעבר InOrder על NumOfCompanies צמתים בעץ. נבחין כי לא נעבור על חברות "מיותרות" שכן אנו בעץ חברות עם עובדים – $O(\text{NumOfCompanies})$.
   - כשסיימנו לעבור על מספר החברות, נצא מן העץ - סיבוכיות זמן כגובה העץ - $O[log(k)]$.
3. נעבור על המערך ונוציא מכל חברה את מזהה העובד המשתכר ביותר - $O(\text{NumOfCompanies})$ – כגודל המערך.

סיכום : סיבוכיות זמן – $O[log(k) + \text{NumOfCompanies}]$, מקום נוסף $O(n_{\text{NumOfCompanies}})$.

GetNumEmployeesMatching():

$ID > 0$:

1. לצורך תמיכה בפעולה זו בסיבוכיות הנדרשת, נבצע מעבר Inorder על עץ החברות בין שני גבולות. לשם מציאת הגבולות, נחפש בעץ העובדים תחת החברה את הצומת הכי קרובה למינימום הנדרש (מלמעלה), ואת הצומת הכי קרובה למקסימום (מלמטה). לצורך כך נשתמש באלגוריתם חיפוש. 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 $\boldsymbol{O[log(n_{Company})]}$.
2. מעבר Inorder בין 2 הגבולות:
   - נרד בעץ באופן דומה לאלגוריתם החיפוש עד שנגיע לגבול הראשון. מובטח שהוא קיים בעץ לפי מציאת הגבולות שלעיל - $\boldsymbol{O[log(n_{Company})]}$.
   - לאחר שהגענו לגבול התחתון, נבצע מעבר InOrder, עד שנגיע לגבול העליון. בעקבות מעבר ה-InOrder , למעשה נעבור על כל הצמתים שבין הגבול התחתון לגבול העליון (ולא מעבר) - $\boldsymbol{O(TotalNumOfEmployees)}$. ייתכן כי עד שנגיע לגבול התחתון, נעבור על צמתים קטנים ממנו. אך מספר צמתים זה, חסום ע"י גובה העץ ולכן יוסיף לסיבוכיות הזמן הכוללת $\boldsymbol{O[log(n_{Company})]}$.
   - על כל צומת מבצעים פעולה יחידה ב- $\boldsymbol{O(1)}$.

- ולכן, סה"כ, המעבר האיטרטיבי חסום ע"י:

$$\boldsymbol{O\left[log\left(n_{Company}\right) + TotalNumOfEmployees\right]}$$

3. כדי שנוכל להחזיר את המזהים, וכן לבדוק את טווח המשכורות, ביצענו את המעבר שצוין לעיל כדי לספור את מספר הצמתים בין הגבול העליון לתחתון. אם הוא חיובי, הקצנו מערך בגודל זה – סיבוכיות מקום נוסף $\boldsymbol{O(TotalNumOfEmployees)}$.
4. עברנו על כל צמתי המערך (שהם עובדים), ובדקנו אם הם עומדים בתנאי השכר וה-Grade. אם כן, עדכנו מונה מתאים – סיבוכיות זמן $\boldsymbol{O(TotalNumOfEmployees)}$.

<u>$ID < 0$:</u>

1. באופן זהה, רק שהביצוע הינו על עץ כל העובדים במערכת. חיפוש הגבולות – $\boldsymbol{O[log(n)]}$.
2. מעבר InOrder בין 2 הגבולות: $\boldsymbol{O[log(n) + TotalNumOfEmployees]}$

סיכום: <u>$ID > 0$</u> : סיבוכיות זמן – $O\left[log\left(n_{Company}\right) + TotalNumOfEmployees\right]$ מקום נוסף – הקצאת מערך ו-InOrder $O[TotalNumOfEmployees + \log(k)]$.

<u>$ID < 0$</u> : סיבוכיות זמן – $O[log(n) + TotalNumOfEmployees]$, מקום נוסף – מערך ו-InOrder $O[TotalNumOfEmployees + \log(n)]$.

<u>Quit():</u>

1. הקצנו וייבאנו למערך בגודל n את כל העובדים במערכת, ע"י מעבר InOrder - $\boldsymbol{O(n)}$.
2. נעבור על כל המערך, וניגש דרך כל עובד למצביע השמור בו אל החברה שלו. הרסנו את 2 עצי העובדים של החברה בעזרת האלגוריתם שצוין ב<u>פעולות כלליות נוספות</u>:
   - סיבוכיות זמן - הריסת עץ אחד של חברה: $\boldsymbol{O\left(n_{Company}\right)}$.
   - אם הרסנו את עצי החברה, לא יתבצע דבר אם ניגש אליה שוב - $\boldsymbol{O(1)}$
   - מאחר וכל עובד נמצא רק תחת חברה אחת, וקיים רק עותק אחד של חברה במבנה הנתונים, הרי שאם נהרוס $\boldsymbol{n_{Company}}$ עובדים של כל חברה, סה"כ נהרוס:
   - $\boldsymbol{n_{Company_1} + n_{Company_2} + n_{Company_3} + \cdots + n_{Company_k} = n \rightarrow O(n)}$
3. כעת, נהרוס את עצי העובדים במבנה הנתונים הכללי - $O(n)$, וכן את עצי החברות -$O(k)$.

<u>**סיכום**: סיבוכיות זמן – $O(k+n)$, מקום נוסף – מערך העובדים $O(n)$.</u>

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