מבני נתונים 1

234218

תרגיל רטוב

2

מספר

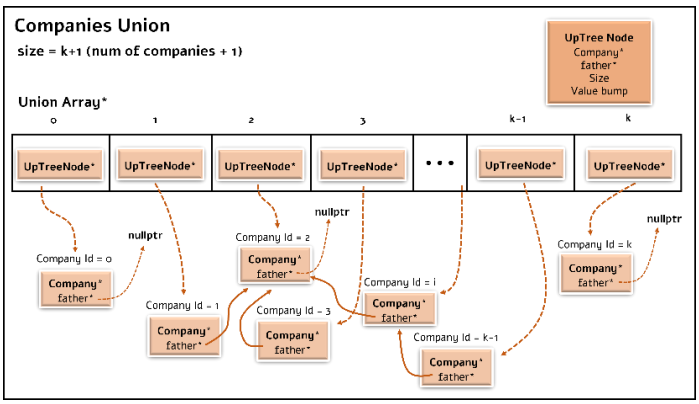
הוגש ע"י:

עדי צחורי	315374066
שם	מספר זהות
מתן צחורי	208936989
שם	מספר זהות

מבני נתונים 1 - 234218 - אביב 2022

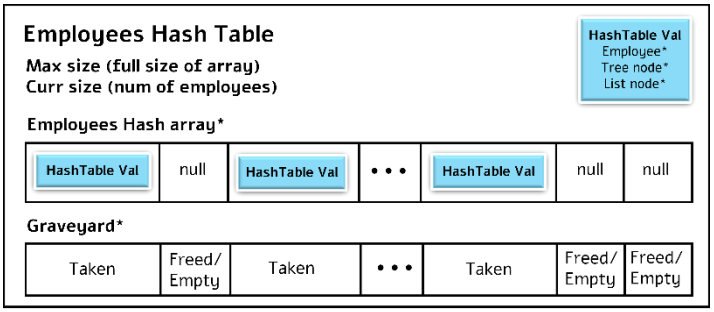
גיליון רטוב 2

מבני הנתונים בהם נשתמש למימוש המבנה שלנו:



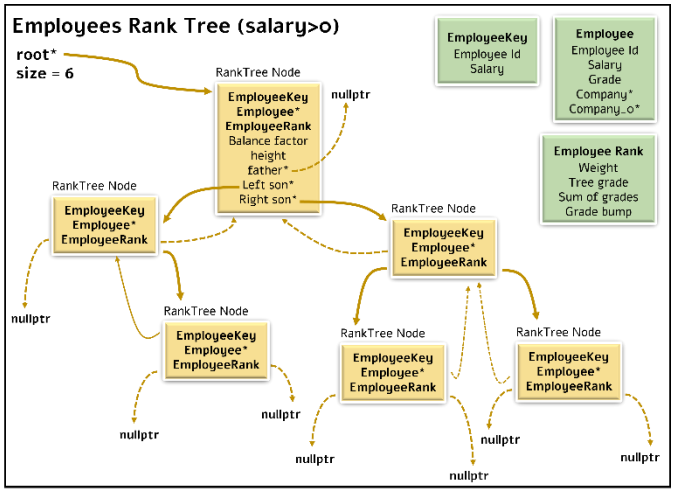
1. ***Union Find*** – מבנה UF עבור החברות, ממומש באמצעות כיווץ מסלולים, עצים הפוכים, ואיחוד לפי גודל.
המבנה מורכב ממערך בגודל קבוע שמכיל מצביעים ל-$UpTreeNode$.

1.1. ***UpTree Node*** – צומת בעץ הפוך במבנה ה-UF.
השדות של $UpTreeNode$:
- $*$ ***Company*** – מצביע לחברה.
- $*$ ***father*** – מצביע לצומת האב של הצומת הנוכחי.
- ***size*** – גודל העץ ההפוך אליו משתייך הצומת.
- ***value bump*** – משתנה המשמש לחישוב השווי של חברה במבנה ה-UF.



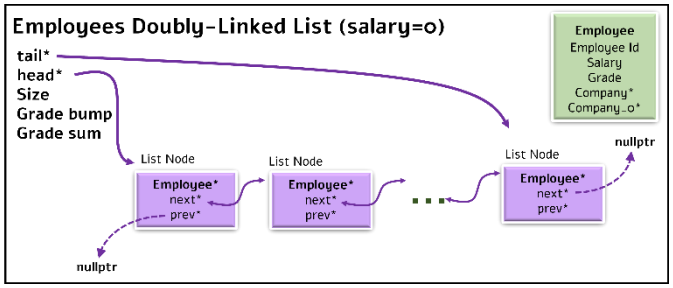
2. ***Hashtable*** – טבלת ערבול הממומשת באמצעות 2 **מערכים דינאמיים**, אחד עבור העובדים עצמם, והשני עבור ה-$graveyard$, כפי שנלמד בכיתה.
הטבלה משמשת עבור גישה לעובדים של חברה ב-$O(1)$ בממוצע על הקלט.
כל תא במבנה מכיל $Employee\ Hashtable\ Val$.
נשמור על פקטור עומס תקין, ונשתמש במערך דינאמי שיגדל פי 2 כאשר חצי מהמערך מלא, ויקטן פי 2 כאשר רבע מהמערך מלא.

2.1. ***Employee Hashtable Val*** – אובייקט המשמש עבור טבלת ערבול, מאפשר גישה למיקום של העובד ברשימה/בעץ של חברה.
האובייקט מכיל את השדות:
- $*$ ***Employee*** – מצביע לעובד.
- $*$ ***tree node*** – מצביע לצומת שמכיל את העובד בעץ העובדים של החברה (אם $salary > 0$).
- $*$ ***list node*** – מצביע לצומת שמכיל את העובד ברשימה המקושרת של העובדים (אם $salary = 0$).



3. ***Rank tree*** – עץ דרגות מבוסס עץ AVL כפי שנלמד בהרצאה ובתרגול.
העץ משמש עבור העובדים בחברה מסוימת בעלי $salary > 0$.
הטיפוס של הצמתים בעץ נקבעים לפי הטיפוס של $Node$.

3.1. $Node = \{key, data, rank\}$ – אובייקט המשמש לייצוג צומת בעץ דרגות, כאשר:
- $Employee\ key$ = ***key***
- $(Employee\ *)$ מצביע לעובד = ***data***
- $Employee\ Rank$ = ***rank***



4. ***Doubly – Linked List*** – רשימה מקושרת דו-כיוונית (בהתאם לנלמד בכיתה) עבור העובדים בחברה מסוימת בעלי $salary = 0$.
בנוסף, אנו מחזיקים משתנים עבור חישוב סכומי הדרגות של העובדים ברשימה. המשתנים הנוספים:
- ***sum of grades*** – סכום הדרגות של העובדים ברשימה.
- ***grade bump*** – משתנה המשמש לחישוב התוספת שניתנה לדרגה של עובד ברשימה.

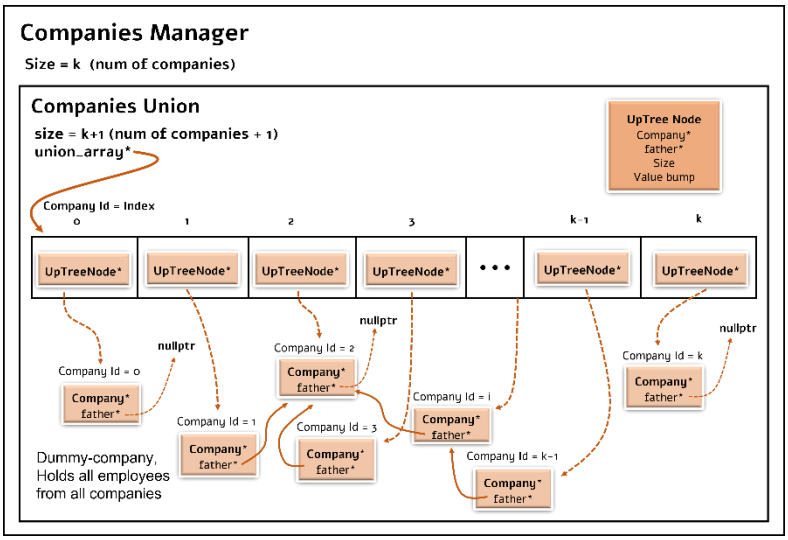
מבני נתונים 1 - 234218 - אביב 2022

גיליון רטוב 2

תיאור מבנה הנתונים שלנו:

מבנה הנתונים שלנו ***Companies Manager*** מכיל 2 אובייקטים:

1. מערך ***Companies Union*** עבור כל החברות במבנה, כפי שתואר מעלה.
   המערך בגודל $k+1$, כאשר בתא 0 נמצאת חברת-דמה ($company\_0$) המכילה את כל העובדים שהוכנסו למבנה.
2. ***size*** – שדה בו נשמר מספר החברות הקיימות במבנה, k.



כמו כן, המבנה מתבסס על מחלקות המייצגות עובד וחברה:

1. ***Company*** – אובייקט המשמש לייצוג חברה. השדות של $Company$:
   - ***Company Id*** – מזהה ייחודי של החברה.
   - ***Value*** – השווי של החברה.
   - ***Employee Hash Table*** – טבלת ערבול ממומשת במערך דינאמי, המכילה את כל העובדים בחברה.
   - ***Zero Salary Employee List*** – רשימה מקושרת דו-כיוונית המכילה את כל העובדים בחברה שיש להם $salary = 0$.
   - ***Salary filtered Employee tree*** – עץ דרגות (מתואר מטה) המכיל את כל העובדים בחברה שיש להם $salary > 0$.
     העץ ממוין לפי $salary$ בסדר עולה, ואז לפי $Employee\ id$ בסדר עולה.

2. ***Employee*** – אובייקט המשמש לייצוג עובד. השדות של $Employee$:
   - ***Employee id*** – מזהה ייחודי של העובד.
   - ***Employee Salary*** – המשכורת של העובד.
   - ***Grade*** – דרגת העובד.
   - ***Company*** * – מצביע לחברה בה עובד העובד.
   - ***Company_0*** * – מצביע לאובייקט של חברת הדמה.

2.1. ***Employee Rank*** – אובייקט המשמש לייצוג המידע הנוסף בעץ הדרגות של עובדים בחברה מסוימת.
האובייקט מכיל את הדרגה של עובד (בהתאם לנלמד בכיתה), וכן מידע נוסף עבור חישוב סכומי הדרגות בעץ.
המידע הנוסף הוא:
   - ***tree grade*** – הדרגה המשוקללת של העובד בעץ העובדים.
   - ***sum of grades*** – סכום הדרגות של כל העובדים בתת העץ שבו העובד הנוכחי הוא השורש.
   - ***grade bump*** – משתנה המשמש לחישוב התוספת שניתנה לדרגה של העובד בעץ העובדים.

2.2. ***Employee key*** – אובייקט המשמש כמפתח (key) לזיהוי עובד בעץ דרגות של עובדים בחברה מסוימת.
השדות של $Employee\ key$:
   - ***Employee id*** – מזהה ייחודי של העובד.
   - ***Employee Salary*** – המשכורת של העובד.

מבני נתונים 1 - 234218 - אביב 2022

גיליון רטוב 2

האלגוריתמים בהם השתמשנו:

1. **סיור בעץ** $\boldsymbol{post-order\ /\ pre-order\ /\ in-order}$ כפי שנלמד בכיתה – לטובת מציאת איבר בעץ.
סיבוכיות הזמן של סיור בעץ AVL היא $\log(h)$ כאשר h גובה העץ.
2. **הכנסה לעץ / הוצאה מעץ** – כפי שנלמד בכיתה, עם מימוש של גלגולים RR, LL, RL, LR.
ההוספה של איבר לעץ מתבצעת כפי שנלמד בכיתה, תוך שמירה על עץ מסודר.
סיבוכיות הזמן של הכנסה/הוצאה מעץ AVL היא $\log(h)$ כאשר h גובה העץ.
3. **מיזוג עצים** כפי שנלמד בכיתה – לטובת מימוש הפונקציה $AcquireCompany$, והעברת כל העובדים של החברה הנרכשת אל החברה הרוכשת.
מיזוג 2 עצי AVL שגודלם n_1, n_2 מתבצע בסיבוכיות זמן של $O(n_1+n_2)$, וגם בסיבוכיות מקום של $O(n_1+n_2)$.
בעת מיזוג שתי חברות, נקבל שיש בשתיהן ביחד לכל היותר n עובדים, לכן סיבוכיות זמן ומקום של לכל היותר $O(n)$.
4. $\boldsymbol{Double\ hashing}$ עבור מניעת התנגשויות בטבלת הערבול – כפי שנלמד בכיתה, ובעזרת שימוש במערך דינאמי. השתמשנו בפונקציית ערבול $mod(array\ size)$.
5. **מיזוג חברות** $\boldsymbol{(union)}$ במבנה UF – מתבצע כאשר חברה אחת רוכשת חברה אחרת.
בהתאם לנלמד בכיתה, נדאג למזג את החברה שנמצאת בעץ הפוך קטן יותר, לתוך החברה שנמצאת בעץ הפוך גדול יותר (במידה והחברה הרוכשת נמצאת בעץ הקטן יותר, נדאג לבצע תיקון מתאים על מנת שהחברה הרוכשת תישאר "החברה המייצגת" לאחר הרכישה).

הוכחת סיבוכיות זמן של פונקציות:

1. $\boldsymbol{init}$ – בעת אתחול מבנה הנתונים $CompaniesManager$ מתבצעות הפעולות הבאות:
1.1. מאתחלים מערך בגודל $k+1$ עבור החברות (מבנה UF) – כל תא i מייצג את החברה ה-i, והחברה שנמצאת בתא 0 היא חברה-דמה המכילה את כל העובדים שהוכנסו למבנה.
1.2. עבור כל $k+1$ התאים במערך ה-UF, יוצרים אובייקט של חברה ב-$O(1)$, ויוצרים אובייקט של עץ הפוך עבור החברה גם ב-$O(1)$.
יצירת אובייקט עבור חברה מסוימת כולל אתחול מספר קבוע של שדות ב-$O(1)$, אתחול טבלת ערבול עבור העובדים ב-$O(1)$ (יצירת מערך בגודל התחלתי קבוע), אתחול רשימה מקושרת ריקה ב-$O(1)$ עבור העובדים בעלי משכורת=0, ואתחול עץ דרגות ריק ב-$O(1)$ עבור העובדים בעלי משכורת>0.
יצירת אובייקט $UpTreeNode$ כולל אתחול של מספר קבוע של שדות ב-$O(1)$.
בסה"כ אתחול המבנה מתבצע ב-$O(k)$.

2. $\boldsymbol{AddEmployee}$ –
2.1. תחילה מוצאים את חברה-0 במבנה UF של החברות באמצעות פעולת $find$ של המבנה. מתבצע ב-$O(1)$ משום שחברה-0 לא מתמזגת לעולם עם חברה אחרת.
2.2. בודקים בטבלת הערבול של חברה-0 האם העובד כבר הוכנס למבנה. בהתאם לנלמד בכיתה על טבלת ערבול, זה מתבצע ב-$O(1)$ משוערך בממוצע על הקלט (חישוב הסיבוכיות המשוערכת מופיע בהמשך).
2.3. אם העובד לא הוכנס למבנה, מוצאים את החברה בעלת $CompanyID$ במבנה UF של החברות באמצעות פעולת $find$ של המבנה. זה מתבצע ב-$O(\log^*(k))$ משוערך, בהתאם לנלמד בכיתה על מבנה UF עם המימוש שתיארנו (חישוב הסיבוכיות המשוערכת מופיע בהמשך).
2.4. יוצרים אובייקט עבור העובד ב-$O(1)$ (אתחול מספר קבוע של שדות) בהתאם לפרמטרים שהתקבלו בפונקציה.
2.5. מוסיפים את העובד החדש לטבלת הערבול של החברה ב-$O(1)$ (בממוצע על הקלט), ולאחר מכן מוסיפים אותו לראש הרשימה המקושרת של החברה (עובד חדש הוא תמיד בעל משכורת=0), גם זה מתבצע ב-$O(1)$.
בסה"כ הוספת עובד מתבצעת ב-$O(\log^*(k))$ משוערך (עם הפונקציות הנדרשות) בממוצע על הקלט.

3. $\boldsymbol{RemoveEmployee}$ –
3.1. מוצאים את חברה-0 במבנה UF של החברות – מתבצע ב-$O(1)$ (כפי שהסברנו קודם לכן).
3.2. בודקים בטבלת הערבול של חברה-0 האם העובד כבר הוכנס למבנה. מתבצע ב-$O(1)$ משוערך בממוצע על הקלט (חישוב הסיבוכיות המשוערכת מופיע בהמשך).

3.3. אם העובד קיים במבנה, הוא מחזיק מצביע לחברה בה הוא עובד, לכן ניגשים אליה ב-$O(1)$.

3.4. מסירים את העובד תחילה מהחברה בה הוא עובד, ולאחר מכן מחברה-0, ולבסוף מוחקים את האובייקט של העובד.
בשתי החברות מבצעים את אותו תהליך – מוצאים את העובד בטבלת הערבול ב-$O(1)$, ניגשים למיקום של העובד בעץ/ברשימה באמצעות המצביע השמור בטבלת הערבול, מסירים אותו מהעץ/מהרשימה בהתאם (מתבצע ב-$O(\log(n))$ בעץ או ב-$O(1)$ ברשימה), ולבסוף מסירים את העובד מטבלת הערבול.

בסה"כ הסרת עובד מתבצעת ב-$O(\log(n))$ משוערך עם הוספת עובד, בממוצע על הקלט.

<u>הוכחת סיבוכיות משוערת עבור $AddEmployee$ ו-$RemoveEmployee$:</u>

הוצאת/הכנסת עובד למבנה מסתמכת תמיד על טבלת הערבול של חברה-0, שמכילה את כל n העובדים (ובפרט גדולה/שווה לכל טבלאות הערבול של החברות האחרות).
טבלת הערבול הדינאמית שמימשנו דואגת לשמור על פקטור עומס תקין, ולכן מקיימת את הנחת הפיזור האחיד הפשוט, ולכן כפי שראינו בתרגול פעולת הכנסה/הוצאה של אובייקט מהטבלה מתבצעת ב-$O(1)$ משוערך בממוצע על הקלט.
לאחר מציאת העובד בטבלת הערבול של חברה-0, מבצעים פעולות נוספות בכל אחת מהפונקציות (כפי שתואר מעלה), ולכן נקבל בכל פונקציה את הסיבוכיות הרצויה.

4. $\boldsymbol{AcquireCompany}$ – מתבצע באמצעות פעולת $unify$ של המבנה UF.

4.1. מוצאים עבור 2 החברות את "החברה המייצגת" של כל אחת במבנה UF באמצעות פעולת $find$, בהתאם לנלמד בכיתה.
מתבצע ב-$O(\log^*(k))$ משוערך, נוכיח בהמשך.

4.2. אם מצאנו 2 "חברות מייצגות" שונות, נבצע מיזוג חברות בהתאם לגודל העצים ההפוכים. מבצעים מספר קבוע של פעולות עדכוני שדות ומצביעים, לכן זה מתבצע ב-$O(1)$.

4.2.1. אם החברה הרוכשת A שייכת לעץ הפוך שגודלו גדול/שווה לעץ ההפוך של החברה הנרכשת B, נמזג את העץ ההפוך של חברה B לתוך העץ ההפוך של חברה A ונדאג לעדכן את שדה ה-$value\_bump$ של הצמתים בעצים ההפוכים באופן הבא:

$$A.value\_bump \mathrel{+}= (B.value\_bump + B.value) * factor$$
$$B.value\_bump \mathrel{-}= A.value\_bump$$

4.2.2. אחרת, אם החברה הרוכשת A שייכת לעץ הפוך שגודלו קטן יותר מהעץ ההפוך של החברה הנרכשת B, נמזג את העץ ההפוך של חברה A לתוך העץ ההפוך של חברה B, אך נדאג להחליף ביניהן את הסדר כך שחברה A היא המייצגת בעץ המאוחד.

- נחשב את הערך האמיתי של כל חברה ($value + value\_bump$) ונעדכן זאת בחברה עצמה.
- נחליף את המצביעים לחברות בצמתים כך שהצומת שהכיל את חברה A יכיל את חברה B ולהיפך.
- נעדכן את המצביעים לצמתים במערך ה-UF, כך שכל תא יחזיק את מצביע לצומת הנכון לאחר ההחלפה.
- נעדכן את השדות של ה-$value$ וה-$value\_bump$ בצמתים של A ו-B על מנת לוודא שכל הצמתים האחרים בעצים ההפוכים של A ו-B ימשיכו להחזיק בערכים הנכונים לאחר המיזוג.

4.3. נבצע מיזוג של השדות הפנימיים של 2 החברות (נעדכן את הגודל של החברה הרוכשת A, נמזג את הרשימות המקושרות, נמזג את עצי הדרגות, נמזג את טבלאות הערבול, נבצע סיור $in-order$ על העץ המאוחד ונעדכן בטבלת הערבול כך שכל עובד יכיל מצביע למקום הנכון בעץ המאוחד, וכן נעדכן את כל העובדים כך שיכילו מצביע לחברה הרוכשת A).
עוברים על כל העובדים בשתי החברות מספר קבוע של פעמים, לכן זה מתבצע ב-$O(n_A + n_B)$.

בסה"כ רכישת חברה מתבצעת ב-$O(\log^*(k) + n_A + n_B)$ משוערך, בממוצע על הקלט.
(חישוב הסיבוכיות המשוערכת מופיע בהמשך)

5. $\boldsymbol{EmployeeSalaryIncrease}$ –

5.1. מוצאים את חברה-0 במבנה UF של החברות באמצעות פעולת $find$ של המבנה. מתבצע ב-$O(1)$ (כפי שהסברנו קודם לכן).

5.2. בודקים בטבלת הערבול של חברה-0 האם העובד קיים במבנה. מציאת עובד בטבלה מתבצעת ב-$O(1)$ בממוצע על הקלט.

5.3. אם העובד קיים במבנה, הוא מחזיק מצביע לחברה בה הוא עובד, שומרים את המצביע הזה, מתבצע ב-$O(1)$.

5.4. מסירים את העובד מ-2 החברות (ללא מחיקה של האובייקט של העובד), מעדכנים את המשכורת של העובד (רק אם הסכום להעלאה גדול ממש מ-0) ומכניסים חזרה את העובד ל-2 החברות.

אם לפני ההעלאה, העובד היה בעל משכורת=0, נסיר אותו מהרשימה המקושרת של כל חברה, ולאחר העלאת המשכורת הוא ייכנס לעץ העובדים של כל חברה. מתבצע ב-$O(\log(n))$, בהתאם להסרה והוספה של איברים לעץ.
בסה"כ העלאת משכורת של עובד מתבצעת ב-$O(\log(n))$ בממוצע על הקלט.

6. $\boldsymbol{PromoteEmployee}$ –

6.1. מוצאים את חברה-0 במבנה UF של החברות באמצעות פעולת $find$ של המבנה. מתבצע ב-$O(1)$ (כפי שהסברנו קודם לכן).
6.2. בודקים בטבלת הערבול של חברה-0 האם העובד קיים במבנה. מציאת עובד בטבלה מתבצעת ב-$O(1)$ בממוצע על הקלט.
6.3. אם העובד קיים במבנה, הוא מחזיק מצביע לחברה בה הוא עובד, שומרים את המצביע הזה, מתבצע ב-$O(1)$.
6.4. כעת רק בחברה האמיתית אליה שייך העובד: ניגשים למיקום של העובד בטבלת הערבול ב-$O(1)$, לאחר מכן ניגשים למיקום של העובד ברשימה/בעץ בהתאם באמצעות המצביע השמור בטבלת הערבול, גם זה מתבצע ב-$O(1)$. מעדכנים את הדרגה שלו בהתאם לפרמטר שהתקבל (שדה $Grade$ השמור באובייקט של העובד עצמו).
ברשימה המקושרת – מוסיפים את ההעלאה של הדרגה לשדה של סכום הדרגות הכולל של העובדים ברשימה. מתבצע ב-$O(1)$.
בעץ העובדים – לאחר העלאת הדרגה לעובד, מטפסים לאורך מסלול החיפוש עד לשורש ומעדכנים בכל צומת במסלול את השדה של סכום הדרגות של העובדים בתת העץ. מתבצע ב-$O(\log(n))$.
בסה"כ עדכון דרגה של עובד מתבצעת ב-$O(\log(n))$ בממוצע על הקלט.

7. $\boldsymbol{sumOfBumpGradeBetweenTopWorkersByGroup}$ –

7.1. מוצאים את החברה המבוקשת במבנה UF של החברות באמצעות פעולת $find$ של המבנה. מתבצע ב-$O(\log^*(k))$ משוערך (חישוב הסיבוכיות המשוערכת מופיע בהמשך).
7.2. אם אין מספיק עובדים בחברה (פחות מ-m) או שאין מספיק עובדים בעלי משכורת>0 (פחות מ-m עובדים בעץ) מחזירים שגיאה.
7.3. אחרת, מוצאים את העובד ה-m מימין בעץ (הצומת בעץ שהדרגה שלו היא $tree\_size - m$), ואז מחשבים את סכום הדרגות של העובדים בעץ עד לצומת זה, לפי האלגוריתמים שנלמדו בכיתה למציאת צומת/חישוב סכום דרגות בעץ דרגות. זה מתבצע ב-$O(\log(n))$.
7.4. מחזירים את הסכום המבוקש: סכום הדרגות של כל העובדים השמור בשורש העץ, פחות הסכום שחישבנו בסעיף 7.3.
בסה"כ חישוב הסכום המבוקש מתבצע ב-$O(\log^*(k) + \log(n))$ משוערך.
(חישוב הסיבוכיות המשוערכת מופיע בהמשך)

8. $\boldsymbol{averageBumpGradeBetweenSalaryByGroup}$ –

8.1. מוצאים את החברה המבוקשת במבנה UF של החברות באמצעות פעולת $find$ של המבנה. מתבצע ב-$O(\log^*(k))$ משוערך (חישוב הסיבוכיות המשוערכת מופיע בהמשך).
8.2. מוצאים את העובד בעל המשכורת הגבוהה ביותר בחברה ואת העובד בעל המשכורת הנמוכה ביותר בחברה, ומשווים אותם לפרמטרים שהתקבלו בפונקציה על מנת לבדוק אם קיימים עובדים בחברה בטווח המשכורות המבוקש.
8.3. אם קיימים עובדים בטווח המשכורות המבוקש, בודקים האם עובדים בעלי משכורת=0 נכללים בטווח (העובדים ברשימה) והאם עובדים בעלי משכורת>0 נכללים בטווח (העובדים בעץ).
8.4. מאתחלים משתנה $num\_emp = 0$ עבור חישוב מספר העובדים בחברה שהמשכורת שלהם נמצאת בטווח המבוקש, ומשתנה $sum = 0$ עבור חישוב סכום הדרגות של העובדים בעלי משכורת בטווח.

- אם 0 כלול בטווח, אז מוסיפים את מספר העובדים ברשימה ל-num_emp, ומוסיפים ל-sum את סכום הדרגות של כל העובדים ברשימה:

$$num\_emp \mathrel{+}= list.size$$
$$sum \mathrel{+}= list.grade\_sum + (list.size \cdot list.grade\_bump)$$

זה מתבצע ב-$O(1)$.

- אם העץ כלול בטווח ($lowest\_salary$ או $highest\_salary$ גדולים מ-0), אז מוצאים את הצומת בעץ שמתאים לעובד שהמשכורת שלו קרובה ביותר מלמטה/שווה ל-$highest\_salary$, נסמן ב-H, ומוצאים את הצומת בעץ שמתאים לעובד שהמשכורת שלו קרובה ביותר מלמעלה/שווה ל-$lowest\_salary$, נסמן ב-L.
זה מתבצע ב-$O(\log(n))$.

מחשבים את מספר העובדים בעץ ואת סכום הדרגות של העובדים בעץ עד לצומת H, וכנ"ל עד לצומת L (לפי האלגוריתמים שנלמדו בכיתה למציאת צומת/חישוב סכום דרגות בעץ דרגות). מתבצע ב-$O(\log(n))$.
כעת מחשבים את מספר האנשים בטווח ואת סכום הדרגות בטווח:

$$num\_emp += numEmpUpTo(H) - numEmpUpTo(L)$$
$$sum += sumEmpGradeUpTo(H) - sumEmpGradeUpTo(L)$$

8.5. מחשבים את דרגת העובד הממוצעת המבוקשת על ידי:

$$avg = \frac{sum}{num\_emp}$$

בסה"כ חישוב הממוצע המבוקש מתבצע ב-$O(\log^*(k) + \log(n))$ משוערך.
(חישוב הסיבוכיות המשוערכת מופיע בהמשך)

9. $\boldsymbol{companyValue}$ –

9.1. מאתחלים משתנה עבור חישוב השווי של החברה $real\_value = 0$.

9.2. מוצאים את החברה המבוקשת במבנה UF של החברות באמצעות פעולת $find$ של המבנה, מתבצע ב-$O(\log^*(k))$ (חישוב הסיבוכיות המשוערכת מופיע בהמשך).
מוסיפים את השווי השמור בחברה ל-$real\_value$:

$$real\_value += compamy.value$$

9.3. מתקדמים במעלה העץ ההפוך, ובכל צומת בעץ מוסיפים את השדה של המידע הנוסף $value\_bump$ ל-$real\_value$. אורך מסלול החיפוש בעץ ההפוך הוא $O(\log^*(k))$ משוערך.

$$real\_value += uptree\_node.value\_bump$$

בסה"כ חישוב השווי האמיתי של חברה מתבצע ב-$O(\log^*(k))$ משוערך.

<u>הוכחת סיבוכיות משוערת עבור הפונקציות $addEmployee$, $acquireCompany$, $companyValue$, $averageBumpGradeBetweenSalaryByGroup$, $sumOfBumpGradeBetweenTopWorkersByGroup$:</u>
נוכיח בשיטת הצבירה שזמן הריצה הדרוש עבור מציאת חברה בכל אחת מהפונקציות הנ"ל הוא $O\left(log^*(k)\right)$.
בכל אחת מהפונקציות הללו אנחנו מבצעים מספר קבוע של פעולות חיפוש חברה, באמצעות שימוש בפעולה $find$ של המבנה UF.
בהתאם לנלמד בהרצאה, ובהתאם למימוש שבחרנו, זמן הריצה של פעולת חיפוש הוא $O\left(log^*(k)\right)$, לכן עבור סדרה של m פעולות מבין הפונקציות הללו זמן הריצה הכולל של כל m הפעולות הוא $O\left(m \cdot log^*(k)\right)$.
לכן לפי שיטת הצבירה, המחיר המשוערך של כל פעולת חיפוש בכל פונקציה הוא $\frac{O\left(m \cdot log^*(k)\right)}{m} = O\left(log^*(k)\right)$.
לאחר מציאת החברה ע"י פעולת $find$ ב-UF, מבצעים פעולות נוספות בכל אחת מהפונקציות (כפי שתואר מעלה), ולכן נקבל בכל פונקציה את הסיבוכיות הרצויה.

10. $\boldsymbol{Quit}$ –

10.1. מוצאים את חברה-0 במבנה UF של החברות באמצעות פעולת $find$ של המבנה, מתבצע ב-$O(1)$.

10.2. עוברים על טבלת הערבול של חברה-0, בודקים בכל תא האם שמור עובד בתא זה (לפי ה-$graveyard$), ומוחקים את כל האובייקטים של העובדים. אורך המערכים בשלב זה הוא לכל היותר $4n$ לפי הדרך בה הגדרנו את המערך הדינאמי, לכן זה מתבצע ב-$O(n)$.

10.3. עוברים על כל $k+1$ התאים במערך של UF, ומוחקים את המבנים הפנימיים של כל חברה (טבלת הערבול, הרשימה והעץ). מאחר והחזקנו מצביעים לעובדים בכל המבנים הפנימיים של החברה, כל המבנים הללו כעת מכילים מצביעים ריקים ולכן הם נמחקים מבלי להשאיר זיכרון תפוס.

10.4. לאחר מחיקת המבנים הפנימיים של החברה, מוחקים את האובייקט של החברה עצמה. זה מתבצע ב-$O(k)$.

10.5. לבסוף מוחקים את המבנה כולו. זה מתבצע ב-$O(1)$.

בסה"כ מחיקת המבנה כולו מתבצעת ב-$O(k+n)$.

11. בונוס - $\boldsymbol{bumpGradeToEmployees}$ –

11.1. עוברים על כל $k+1$ החברות במערך של UF, ובכל אחת מהחברות מבצעים תהליך דומה לזה שתואר בפונקציה 8 $averageBumpGradeBetweenSalaryByCompany$.

11.2. מוצאים את העובד שהמשכורת שלו היא הגבוהה ביותר בחברה, ואת העובד שהמשכורת שלו הנמוכה ביותר. משווים את המשכורות שלהם לטווח שהתקבל בפונקציה על מנת לקבוע האם הרשימה/העץ שייכים לטווח. מציאת העובדים מתבצעת במקרה הגרוע ב-$O(\log(n))$ בעץ, ובמקרה הטוב ב-$O(1)$.

- אם הרשימה כלולה בטווח, אז מעלים את הדרגה לכל העובדים ששייכים לרשימה. זה מתבצע ב-$O(1)$.
  מגדילים את השדה $grade\_bump$ של הרשימה ב-$bump\_amount$
  $grade\_bump += bump\_amount$
- אם העץ כלול בטווח ($lowest\_salary$ או $highest\_salary$ גדולים מ-0), אז מוצאים את הצומת בעץ שמתאים לעובד שהמשכורת שלו קרובה ביותר מלמטה/שווה ל-$highest\_salary$, נסמן ב-H, ומוצאים את הצומת בעץ שקרוב ביותר מלמטה ל-$lowest\_salary$ על מנת שהעדכון יחול על כל הצמתים שמשכורתם בטווח, נסמן ב-L.
  זה מתבצע ב-$O(\log(n))$.
  - בדומה לנלמד בתרגול, מסיירים בעץ מהשורש ועד לצומת H, ובכל צומת במסלול בודקים האם "יצאנו מהטווח" או "נכנסנו לטווח" (האם המשכורת של הצומת הנוכחי גדולה/קטנה מ-$highest\_salary$).
    אם "יצאנו מהטווח" נקטין את השדה $grade\_bump$ של הצומת בערך שהתקבל, ואם "נכנסנו לטווח" נגדיל את השדה $grade\_bump$ של הצומת בערך שהתקבל.
    לאחר מכן מסיירים חזרה מ-H לשורש ומעדכנים את סכום הדרגות בכל צומת במסלול.
  - חוזרים על התהליך עם צומת L עם הערך שהתקבל בסימן שלילי.

מבצעים מספר קבוע של פעולות לאורך מסלול חיפוש מסוים בעץ (מציאת צמתים לפי משכורת, עדכון ערכים ב-$rank$ של צמתים במסלול וכו'), לכן זה מתבצע ב-$O(\log(n))$.

ביצוע העלאה בדרגה לכל העובדים במבנה דורש $O(\log(n))$ פעולות לכל חברה, ויש לנו $k+1$ חברות, לכן בסה"כ זה מתבצע ב-$O(k \cdot \log(n))$.

<u>הוכחת סיבוכיות מקום:</u>

מערך בגודל $k+1$ עבור UF, בכל תא נשמר מצביע לצומת בעץ הפוך (מכיל מספר קבוע של שדות, לכן דורש $O(1)$ מקום), לכן זה דורש $O(k)$ מקום.

$k+1$ אובייקטים של חברות.

n אובייקטים של עובדים.

בחברה-0 נמצאים כל n העובדים, לכן גודל טבלת הערבול שלה הוא לכל היותר $4n$, גודל הרשימה המקושרת שלה הוא n צמתים שגודל כל אחד מהם הוא $O(1)$, ובעץ הדרגות שלה יש n צמתים שגודל כל אחד מהם הוא $O(1)$, כלומר בסה"כ נדרש $O(n)$ מקום עבור חברה-0.

נסמן את מספר העובדים בכל חברה $1 \le i \le k$ ב-n_i. בחברה ה-i גודל טבלת הערבול הוא לכל היותר $4n_i$, גודל הרשימה המקושרת שלה הוא n_i צמתים, ובעץ הדרגות שלה יש n_i צמתים, לכן נדרש $O(n_i)$ מקום.

נשים לב ש-n העובדים מתחלקים בין כל k החברות האמיתיות, לכן מתקיים שהמקום שנדרש עבור כל החברות יחד הוא:

$$\sum_{i=1}^{k} 4n_i + n_i + n_1 = 6 \cdot \sum_{i=1}^{k} n_i = 6(n_1 + \cdots + n_k) = 6 \cdot n$$

לכן בסה"כ סיבוכיות המקום של המבנה היא $O(n+k)$.