# מבנה נתונים 1

## תרגיל בית רטוב 2

*Wet*2

שם סטודנט: אימן ברהם

תעודת זהות: 212201057

שם סטודנט: עומר מחאמיד

תעודת זהות: 308198134

מבנה הנתונים שלנו:
יש לנו $UnionFind$ של חברות נקרא לו בשם $allCompanies$. ונשמור בכל חולייה שדה של $extraValue$ שאין לו משמעות לבדו אך כשסוכמים אותו על גבי מסלול יעיד על ערך החברה האמיתי, עדכון שדה זה נעשה כמו שיטת הארגזים שנלמדה בתרגול.

יש גם $HashTable$ של כל העובדים עם $Chain\ Hashing$ (והשתמשנו בעץ $AVL$ במקום $LinkedList$).

השתמשנו בעץ דרגות $RankTree$ של עובדים עם $salary > 0$, בשם $allEmployeesWithSalary$, ובכל חולייה שמרנו מידע נוסף, שהוא סכום ה-$grades$ בתת עץ של החולייה. עדכון שדה דומה ואף זהה לעדכון השדה $size$ ששומר גודל כל תת עץ כמו שלמדנו בהרצאה. נקרא לו בשם $allEmployees$. לכל $employee$ שמרנו $id$, $salary$ וגם מצביע על עץ הדרגות של החברה שהוא עובד בה, בשם $company$.
נוסף על כך שמרנו את מס׳ העובדים עם $salary = 0$ וסכום ה-$grades$ של עובדים אלה ואופן עדכון שדות אלה נראה למטה בפונקציות עצמן. נקרא למשתנים אלה בשם $numOfZeroSalaryEmployees$ ו-$sumGradesOfZeroSalaryEmployees$, בנוסף שמרנו שתי השדות האלה לכל חברה בנפרד.

בכל חברה שמרנו שני מבנים, בדומה לאלה ששמרנו לכל העובדים, $HashTable$ של עובדי החברה $allEmployees$ ו- $RankTree$ עם אותו מידע נוסף של עובדי החברה עם $salary > 0$ בשם $employeesWithSalary$. נוסף על כך שמרנו לכל חברה את מס׳ העובדים עם $salary = 0$ וסכום ה-$grades$ של עובדים אלה ואופן עדכון שדות אלה נראה למטה בפונקציות עצמן. נקרא להם באותו שם של משתנים מקבילים לכל העובדים באותם שמות, אך ניגש לאלה תוך מצביע על $company$. כל זה בנוסף ל-$value$ ו-$id$ של כל חברה ששמרנו בתוכה.
הערה: כל $HashTable$ שהשתמשנו בו יש לו מערך דינמי שאנו מגדילים פי 2 ועושים $ReHash$ כאשר הפאקטור $\alpha \geqslant 1$ ונקטין פי 2 כאשר פאקטור $\alpha \leqslant \frac{1}{4}$.

❖ $Init(k)$:
נאתחל את $allCompanies$ להיות $UnionFind$ בגודל $k$ כאשר לחברה למקום ה-$i$ במבנה יהיה $id = i$.
נוסף על כך נאתחל את שני המבנים $allEmployees$ ו- $employeesWithSalary$ של כל חברה ב-$O(1)$.
ונאתחל את שני המשתנים $numOfZeroSalaryEmployees$ ו-
$sumGradesOfZeroSalaryEmployees$ להיות 0.
ואז נאתחל את $allEmployees$ הכללי ו- $allEmployeesWithSalary$ ב-$O(1)$. וגם המשתנים הכלליים $numOfZeroSalaryEmployees$ ו- $sumGradesOfZeroSalaryEmployees$ להיות 0.

ונחזיר מצביע על מבנה הנתונים שהקצנו.

<u>סיבוכיות זמן:</u> כל אתחול ב-$O(1)$ ומעבר על המערך בתוך ה-$UnionFind$ מתבצע ב-$O(k)$. לכן סה״כ $O(k)$.

❖ $AddEmployee(int\ EmployeeID,\ int\ CompanyID,\ int\ Grade)$:
ניצור $Employee$ עם $EmployeeID$ ו- $grade$ הנתונים. נוסיף אותו ל-$allEmployees$.
ניגש ב- $allCompanies$ ונחפש את החברה עם $companyID$, נקרא לה $company$. ניגש ל- $HashTable$ $company \rightarrow allEmployees$ ונוסיף את $Employee$. נבצע השמה

$Employee \rightarrow company \leftarrow company$.
נגדיל את $employee \rightarrow company \rightarrow numOfZeroSalaryEmployees$ ב-1.
נגדיל את $employee \rightarrow company \rightarrow sumGradesOfZeroSalaryEmployees$ ב-$employee \rightarrow grade$
נגדיל את $numOfZeroSalaryEmployees$ ב-1.
נגדיל את $sumGradesOfZeroSalaryEmployees$ ב-$employee \rightarrow grade$. ונחזיר $SUCCESS$.

<u>ולכן סיבוכיות זמן כוללת:</u> יצירת עובד ב-$O(1)$. מס׳ סופי של הוספות ל-$HashTable$ ב-$O(1)$ בממוצע על הקלט. הגדלת מונים ב-$O(1)$. חיפוש ב- $UnionFind$ מתבצע ב- $O(\log^*(k))$ (בשל שימוש במס׳ סופי של פעולות $Union$ ו-$Find$), לכן סה״כ $O(\log^*(k))$ ממוצע על הקלט. משוערך עם:

$$companyValue,\ acquireCompany,\ averageBumpGradeBetweenSalaryGroup,\ sumOfBumpGradeBetweenTopWorkersByGroup,\ removeEmployee$$

❖ $RemoveEmployee(int\ EmployeeID)$:

נבצע חיפוש ב- $HashTable$ המכיל את כל העובדים במבנה הנתונים, משם נשמור מצביע זמני שנקרא לו $company$, שמצביע על החברה שבה עובד העובד בעל תעודת זהות $EmployeeID$, ונבצע בחברה שלו חיפוש ב- $HashTable$ של עובדי החברה, ונשמור את השכר- $salary$ שלו במשתנה זמני. ואז נמחוק את העובד משתי טבלאות הערבול.

אם $employee \rightarrow salary = 0$: נבצע השמה $employee \rightarrow salary \leftarrow SalaryIncrease$.
ניגש לעץ $employee \rightarrow company \rightarrow employeesWithSalary$ ונוסיף את $employee$ לעץ זה.
ניגש לעץ $allEmployeesWithSalary$ ונוסיף את העובד, נקטין את
$employee \rightarrow company \rightarrow numOfZeroSalaryEmployees$ ב-1.
נקטין את $employee \rightarrow company \rightarrow sumGradesOfZeroSalaryEmployees$ ב-$employee \rightarrow grade$.
נקטין את $numOfZeroSalaryEmployees$ ב-1, נקטין את $sumGradesOfZeroSalaryEmployees$ ב-$employee \rightarrow grade$, ובסוף נחזיר $SUCCESS$.

אחרת נקבל כי השכר שלו גדול ממש מ- 0, אזי במקרה זה העובד נמצא גם בעץ דרגות של מבנה הנתונים, וגם בעץ דרגות של החברה בה הוא עובד, ולכן נבצע חיפוש עליו בשני העצים הנ"ל, ואז נמחוק אותו גם משם. ונחזיר $SUCCESS$.

<u>סיבוכיות זמן:</u> חיפוש בטבלת ערבול מתבצע בסיבוכיות זמן משוערך של $O(\log n)$ בממוצע על הקלט, וחיפוש בעץ דרגות מבוסס $AVL$, מתבצע בסיבוכיות זמן של $O(\log n)$ במקרה הגרוע. <u>ולכן סיבוכיות זמן כוללת:</u> $2 * O(\log n) + 2 * O(\log n) \ = \ O(\log n)$, משוערך, בממוצע על הקלט עם: $AddEmployee$.

❖ $AcquireCompany(int\ AcquirerID,\ int\ TargetID,\ double\ Factor)$:

נבצע חיפוש על החברות עם תעודות הזהות $AcquirerID,\ TargetID$, בתוך מבנה ה- $UnionFind$ של החברות, במקרה והמצביע המוחזר מפונקציית ה- $Find$ מצביע לאותה חברה, שהיא החברה הרוכשת של כל קבוצת החברות המחוברות ביניהם, אזי נחזיר $INVALID-INPUT$ , כי במקרה הזה בפועל חברה

מסויימת מנסה לקנות את עצמה, אחרת, מבצעים $Union$ בין שתי קבוצות החברות של $Acquirer$ ושל $Target$, באיחוד לפי גודל כמו שלמדנו בהרצאה על $Union$.

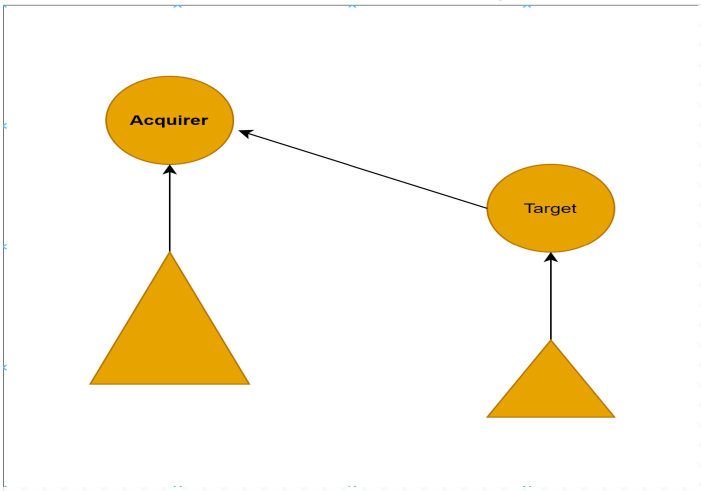
עכשיו נאחד את טבלאות הערבול של שתי הקבוצות ע"י כך שניצור טבלה חדשה בגודל של סכום הגדלים של שתי הטבלאות הקודמות פי 2, ואז נעבור על כל העובדים הנמצאים בשתי הטבלאות ונכניס כל אחד מהם מחדש לטבלה החדשה, את עצי הדרגות של שתי החברות גם צריך למזג אותם, נבצע זאת ע"י העברת שני העצים המקוריים למערכים $acquirerArray,\ targetArray$, לפי סיור $InOrder$ כדי שיהיו ממוינים, ואז נמזג את שני המערכים ונבנה עץ מהמערך החדש כך שיכיל את כל עובדי שתי הקבוצות עם שכר חיובי ,בניית העץ מתבצעת באופן רקורסיבי כך שמתחילים מאמצע המערך, שמים אותו כשורש של העץ, ואז בונים את תת העץ השמאלי, על החלק השמאלי במערך, והתת עץ הימני מהמחלק הימני במערך, כדי לשמור על תכונות עץ דרגות מבוסס $AVL$.

מכיוון שהפונקצייה פועלת באופן רקורסיבי על תת העץ השמאלי ותת העץ הימני, אזי בהתחלה מיצקים את עלים העץ, ואז עולים ברקורסיה כדי לבנות את הצמתים הפנימיים, ולכן אחרי שורות הרקורסיה מבצעים את עדכון השדה הנוסף של עץ הדרגות $numberOfSons,\ gradeOfSubTree$, באופן הבא:

$$node \rightarrow numberOfSons = numberOfSons(left) + numberOfSons(right) + 1$$
$$node \rightarrow gradeOfSubTree = gradeOfSubTree(left) + gradeOfSubTree(right) + node \rightarrow grade$$

הרקורסיה ובניית העץ מהעלים עד השורש שומרת על נכונות השדות של עץ הדרגות.

עדכון השדה הנוסף $extraValue$ במבנה של $UnionFind$, ועדכון העצים ההפוכים ע"י איחוד לפי גודל:




קרה 1: $size\{Acquierer\} \geqslant size\{Target\}$

אזי באיחוד לפי גודל נעדכן את שדה

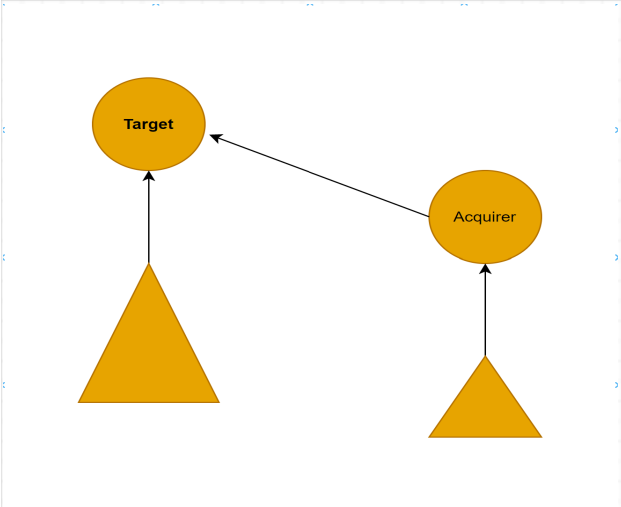
- $father$ של $Target$

היות שווה ל- $Acquirer$

$extra(Acquirer) + = value(main(Target$
$* factor$

$extra(Target) - = extra(Acquirer)$

$size(Acquirer) + = size(Target$




מקרה 2: $size\{Target\} \geqslant size\{Acquirer\}$

1. אזי באיחוד לפי גודל נעדכן את שדה ה- $father$ של $Acquirer$ להיות שווה ל- $Target$

$$extra(Acquirer) + = value(main(Target)) * factor - extra(Target)$$

$$size(Target) + = size(Acquirer)$$

$$Target \rightarrow mainCompany = Acquirer \rightarrow mainCompany$$

בנוסף נעבור על כל העובדים של החברה $Target$, ומעדכנים את שדה ה- $Company$ שלו להיות שווה ל- $mainCompany$ של השורש של העץ ההפוך אחרי העדכון, נעדכן את השדה הנוסף:

- $Acquirer \rightarrow numOfZeroSalaryEmployees + = Target \rightarrow numOfZeroSalaryEmployees$
- $Acuirer \rightarrow sumGradesOfZeroSalaryEmployees + = Target \rightarrow sumGradeOfZeroSalaryEmployees$

ונחזיר $SUCCESS$.

<u>סיבוכיות זמן:</u> חיפוש על שתי החברות מתבצע בסיבוכיות זמן משוערכת של $O(\log^* k)$ בממוצע על הקלט.

- פעולת ה- $union$ מתבצעת בסיבוכיות של $O(1)$, כי מבצעים מספר סופי של פעולות(רק עדכון מצביעים, ושדות של השורש).
- מעבר על טבלאות הערבול ומיזוגם לטבלה חדשה מעבצע בסיבוכיות של מספר העובדים בשתי החברות, כלומר $O(n_{target} + n_{acquirer})$, וגם מעבר על עצי הדרגות של שתי החברות ומיזוגם ובניית העץ החדש לוקח אותה סיבוכיות זמן, מכיוון שנוגעים בכל צומת ומבצעים עליהן מספר סופי של פעולות.

<u>ולכן סיבוכיות זמן כוללת:</u> בממוצע על הקלט: $O(\log^* k) + O(1) + O(n_{target} + n_{acquirer})$, משוערך עם:

$$companyValue,\ addEmployee,\ sumOfBumpGradeBetweenTopWorkersByGroup,\ averageBumpGradeBetweenSalaryByGroup$$

❖ $EmployeeSalaryIncrease(int\ EmployeeID,\ int\ SalaryIncrease)$:

לשם נוחות נניח $SalaryIncrease > 0$, נחפש את העובד ב-$HashTable$ של כל העובדים. נקרא למצביע עליו $employee$.

אם $employee \rightarrow salary = 0$: נבצע השמה $employee \rightarrow salary \leftarrow SalaryIncrease$. ואז ניגש לעץ $employee \rightarrow company \rightarrow employeesWithSalary$ ונוסיף את $employee$ לעץ זה.

ניגש לעץ $allEmployeesWithSalary$ ונוסיף את העובד, נקטין את $employee \rightarrow company \rightarrow numOfZeroSalaryEmployees$ ב-1.

נקטין את $employee \rightarrow company \rightarrow sumGradesOfZeroSalaryEmployees$ ב-$employee \rightarrow grade$

נקטין את $numOfZeroSalaryEmployees$ ב-1, נקטין את $sumGradesOfZeroSalaryEmployees$ ב-

$employee \to grade$.
אחרת מתקיים כי $employee \to salary > 0$: ניגש לעץ
$employee \to company \to employeesWithSalary$ ונסיר את $employee$ לעץ זה, ניגש לעץ
$allEmployeesWithSalary$ ונסיר את העובד, נבצע השמה $employee \to salary \leftarrow SalaryIncrease$.
ניגש לעץ $employee \to company \to employeesWithSalary$ ונוסיף את $employee$ לעץ זה.
ניגש לעץ $allEmployeesWithSalary$ ונוסיף את העובד, בסוף נחזיר $SUCCESS$.

<u>סיבוכיות זמן:</u> מס׳ סופי של גישות למצביעים ב-$O(1)$. מס׳ סופי של הסרות והוספות לעץ דרגות, ב-$O(\log n)$, גישה ב-$HashTable$ ב-$O(1)$ בממוצע על הקלט. <u>ולכן סיבוכיות זמן כוללת:</u> $O(\log n)$ בממוצע על הקלט.

❖ $PromoteEmployee(int\ EmployeeeID,\ int\ BumpGrade)$:
נחפש את העובד עם ה-$id$ המתאים ב-$allEmployees$, נקרא לו $employee$, אם לא מצאנו אותו נחזיר $FAILURE$.
נבצע השמה $employee \to grade\ +\ = BumpGrade$, ואז אם $employee \to salary > 0$:

- ניגש לשני העצים $allEmployeesWithSalary$ ו-
  $employee \to company \to employeesWithSalary$ ונמחק את העובד משני עצים אלה.
- ניגש לשני העצים $allEmployeesWithSalary$ ו-
  $employee \to company \to employeesWithSalary$ ונוסיף את העובד משני עצים אלה.

אחרת: נגדיל את $employee \to company \to sumGradesOfZeroSalaryEmployees$ ב-$BumpGrade$.
נגדיל את $sumGradesOfZeroSalaryEmployees$ ב-$BumpGrade$, ונחזיר $SUCCESS$.
<u>סיבוכיות זמן:</u> חיפוש ב-$HashTable$ $O(1)$ בממוצע על הקלט. הוספה והחסרה של עובד מעץ דרגות $O(\log n)$.
<u>ולכן סיבוכיות זמן כוללת:</u> $O(\log n)$ בממוצע על הקלט.

❖ $SumOfBumpGradeBetweenTopWorkersByGroup(int\ CompanyID,\ int\ m)$:
נקצה מצביע לעץ דרגות בשם $toSearchIn$, אם $companyID = 0$: נסתכל על העץ
$allEmployeesWithSalary$, אם מתקיים ש-$allEmployeesWithSalary \to size < m$: נחזיר $FAILURE$.
אחרת נבצע השמה $toSearchIn \leftarrow allEmployeesWithSalary$.
אם $companyID > 0$: נמצא את החברה עם $companyID$ מתאים, נקרא לה $company$.

- אם מתקיים ש-$company \to employeesWithSalary \to size < m$: נחזיר $FAILURE$.
- אחרת נבצע השמה $toSearchIn \leftarrow company \to employeesWithSalary$.

עכשיו נתבונן בעץ הדרגות $toSearchIn$, נאתחל מצביע לחוליות בשם $temp = toSearchIn \to root$.
נאתחל מונים $numOfEmployeesLeft = k$, ו- $sum = 0$.
נתחיל מהשורש באופן הבא, כל עוד $temp \neq null$:

- אם $temp \to right \to numberOfSons > numOfEmployeesLeft$:
  $temp \leftarrow temp \to right$.
- אחרת אם $temp \to right \to numberOfSons < numOfEmployeesLeft$:

- $sum \leftarrow sum + temp \rightarrow grade + temp \rightarrow right \rightarrow gradeOfSubtree$.
- $numOfEmployeesLeft -= temp \rightarrow right \rightarrow numberOfSons + 1$.
- $temp \leftarrow temp \rightarrow left$.

- אחרת:
  - $sum \leftarrow sum + temp \rightarrow right \rightarrow gradeOfSubtree$, וצא מן הלולאה.

נחזיר את $sum$.

<u>סיבוכיות זמן:</u> חיפוש החברה $O(\log^*(k))$ משוערך עם

$companyValue,\ acquireCompany,\ averageBumpGradeBetweenSalaryGroup,$
$sumOfBumpGradeBetweenTopWorkersByGroup,\ removeEmployee$

הסיור בלולאה לוקח לכל היותר גובה העץ של העובדים עם שכר גבוה מ-0, שזה $O(\log n)$.

<u>ולכן סיבוכיות זמן כוללת:</u> $O\left(\left(\log^*(k)\right) + O(\log n)\right)$.

❖ $AverageBumpGradeBetweenSalaryByGroup(int\ CompanyID,$
$intLowerSalary,\ int\ HigherSalary)$:

תחילה ניצור 4 פונקציות עזר בתוך ה-$RankTree$ שכולן מקבלות מצביע לעובד:

$getNumOfNodesUntilNotIncluding,\quad getSumGradesNotIncluding$
$,getNumOfNodesUntil\quad ,\quad getSumGradeUntil$

נציע מימוש בסיום הפונקציה $AverageBumpGradeBetweenSalaryByGroup$.

כעת נתחיל במימוש הפונקציה: נאתחל שני עובדים דמה, שיש להם $HigherSalary$ ו- $LowerSalary$, נקרא להם $high$ ו- $low$ בהתאמה, ונאתחל שני משתנים $sumGrades$ ו- $numOfEmployees = 0$.

אם $companyID = 0$: $sumGrades \leftarrow allEmployeesWithSalary.getSumGradeUntil(high)$.
$numOfEmployees \leftarrow allEmployeesWithSalary.getNumOfNodesUntil(high)$.

- אם $HigherSalary > 0$:
  - $sumGrades -= allEmployeesWithSalary.getSumGradeUntilNotIncluding(low)$.
  - $numOfEmployees -= allEmployeesWithSalary.getNumOfNodesUntilNotIncluding(low)$.
- אם $LowerSalary = 0$: $sumGrades += sumGradesOfZeroSalaryEmployees$.
  - $numOfEmployees += numOfZeroSalaryEmployees$.

אחרת אם $companyID > 0$: נמצא את $company$ עם ה-$id$ המתאים ב- $allCompanies$.

- $sumGrades \leftarrow company \rightarrow allEmployeesWithSalary.getSumGradeUntil(high)$.
- $numOfEmployees \leftarrow company \rightarrow allEmployeesWithSalary.getNumOfNodesUntil(high)$.
- אם $HigherSalary > 0$:
  - $sumGrades -= company \rightarrow allEmployeesWithSalary.getSumGradeUntilNotIncluding(low)$.
  - $numOfEmployees -= company \rightarrow allEmployeesWithSalary.getNumOfNodesUntilNotIncluding(low)$.
- אם $LowerSalary = 0$:
  - $sumGrades += company \rightarrow sumGradesOfZeroSalaryEmployees$.
  - $numOfEmployees += company \rightarrow numOfZeroSalaryEmployees$.

אם $numOfEmployees = 0$: נחזיר $FAILURE$, אחרת נדפיס את $\frac{sumGrades}{numOfEmployees}$. ונחזיר $SUCCESS$.

---

נציע מימוש לפונקציות העזר:

$getSumGradeUntil(employee)$:

נאתחל משתנה $sum = 0$, נתחיל מהשורש באופן הבא, $temp = root$, כל עוד $temp \neq null$:

- אם $temp \to salary = employee \to salary$:
  - $sum += temp \to grade + temp \to left \to gradeOfSubtree$
  - $temp \leftarrow temp \to right$.
- אחרת אם $temp \to salary > employee \to salary$: $temp \leftarrow temp \to left$.
- אחרת: $sum += temp \to grade + temp \to left \to gradeOfSubtree$, $temp \leftarrow temp \to right$.

החזר את $sum$.

$getSumGradesNotIncluding(employee)$:

נאתחל משתנה $sum = 0$, נתחיל מהשורש באופן הבא, $temp = root$, כל עוד $temp \neq null$:

- $temp \leftarrow temp \to left$.

אחרת: $sum += temp \to grade + temp \to left \to gradeOfSubtree$, $temp \leftarrow temp \to right$.

החזר את $sum$.

$getNumOfNodesUntil(employee)$:

נאתחל משתנה $sum = 0$, ונתחיל מהשורש באופן הבא, $temp = root$, כל עוד $temp \neq null$:

- אם $temp \to salary = employee \to salary$:
  - $sum += 1 + temp \to left \to size$, $temp \leftarrow temp \to right$.
- אחרת אם $temp \to salary > employee \to salary$: אז: $temp \leftarrow temp \to left$.
- אחרת: $sum += 1 + temp \to left \to size$, $temp \leftarrow temp \to right$.

נחזיר את $sum$.

$getNumOfNodesUntilNotIncluding(employee)$:

נאתחל משתנה $sum = 0$, ונתחיל מהשורש באופן הבא, $temp = root$, כל עוד $temp \neq null$:

- אחרת אם $temp \to salary \geqslant employee \to salary$: $temp \leftarrow temp \to left$.
- אחרת: $sum += 1 + temp \to left \to size$, $temp \leftarrow temp \to right$.

נחזיר את $sum$.

<u>סיבוכיות זמן:</u> כל אחת מפונקציות העזר הולכת מקסימום כגובה עץ הדרגות. מכאן נובע כי הן רצות בסיבוכיות $O(\log n)$, ונוסף על כך חיפוש של חברה ב-$UnionFind$ שלוקח $O(\log^*(k))$ משוערך עם $companyValue,\ acquireCompany,\ averageBumpGradeBetweenSalaryGroup,$ $sumOfBumpGradeBetweenTopWorkersByGroup,\ removeEmployee$

<u>ולכן סיבוכיות זמן כוללת:</u> $O(\log^*(k) + \log n)$ משוערך.

❖ $CompanyValue(int\ CompanyID)$:

ניקח את $companyID$ ונחפש ב- $UnionFind$ של החברות על החברה בעלת המזהה $companyID$, ואז ניגש לצומת של החברה בעץ ההפוך שבה היא נמצאת, נגדיר משתנה זמני $totalValueExtra$ מאתחלים אותו ל- $0$, ונבצע את הבא: נסכום את שדה ה- $valueExtra$ של כל חברה ונשים את הסכום ב- $totalValueExtra$, בסוף נוסיף ל- $totalValueExtra$ את ה- $value$ המקורי של החברה, השווה ל- $companyID$.

אז נדפיס את הסכום המייצג את הערך הכולל החברה $companyID$, ונחזיר $SUCCESS$.

<u>סיבוכיות זמן:</u> מבצעים פעולת החיפוש $Find$ על $CompanyID$, ותוך כדי החיפוש סוכמים את ה- $valueExtra$ של כל צומת, ולכן סיבוכיות זמן כוללת: היא $O(\log^* k)$ משוערך עם:

$$acquireCompany,\ addEmployee,\ sumOfBumpGradeBetweenTopWorkersByGroup,\ averageBumpGradeBetweenSalaryByGroup$$

❖ $Quit()$:

נעבור על כל חברה וחברה במבנה של $UnionFind$ ששמרנו בו את החברות שלנו, וניגש משם לטבלת הערבול של העובדים שלה, וניגש גם לעץ דרגות שלה, ונרוקן את העובדים משניהם, ואז נשחרר אותם מהזיכרון, ובסוף נמחוק את החברה, כל המעברים והמחיקות הוא לפי מספר החברות והעובדים במבנה, כלומר $O(n+k)$.

נעבור על טבלת הערבול של מבנה הנתונים, בנוסף נעבור על עץ הדרגות של במבנה ונרוקן את שניהם מהעובדים, שלוקח סיבוכיות זמן כמספר העוסדים במבנה, כלומר $O(n)$.

<u>ולכן סיבוכיות זמן כוללת:</u> במקרה הגרוע : $O(n+k)+O(n)=O(n+k)$

## <u>סיבוכיות מקום כוללת:</u>

1. $HashTable$ של כל העובדים במבנה, בעל גודל מקסימלי של $2n$ השווה למספר העובדים במבנה הנתונים, כלומר לוקח סיבוכיות מקום של $O(n)$ במקרה הגרוע.
2. $RankTree$ של העובדים בעלי שכר $salary > 0$, שגודלו במקרה הגרוע הוא $n$ השווה למספר העובדים הכולל במבנה הנתונים, כלומר לוקח סיבוכיות מקום של $O(n)$.
3. $UnionFind$ של כל החברות הנמצאות במבנה הנתונים, בעל גודל השווה ל- $k$, המייצג את מספר החברות במבנה הנתונים.
4. $HashTable$ של העובדים הנמצאים בכל חברה, סכום סיבוכיות המקום של כל טבלאות הערבול לכל החברות הוא לכל היותר $O(n)$ במקרה הגרוע שהוא מספר העובדים בכולל במבנה הנתונים.
5. $RankTree$ של העובדים בעלי שכר $salary > 0$, סיבוכיות המקום הכוללת של כל עצי הדרגות של כל החברות היא $O(n)$, המייצג מספר העובדים במבנה הנתונים.

<u>ולכן סיבוכיות מקום כוללת:</u> במקרה הגרוע : $O(n+k)$.