מבני נתונים 1

234218

תרגיל רטוב 2

סטודנטים:

בוראן סוויד – ת״ז 211516836.

סוראי סוויד – ת״ז 322827239.

תיאור המבנה:

המבנה שלנו מכיל את מבני הנתונים הבאים:

1. **טבלת ערבול דינמית**- (מערך של אובייקטים מסוג Employee) הממומשת בשיטת ה- chain hashing למניעת התנגשות. כאשר כל תא במערך זה מכיל את המידע הבא על כל עובד בנפרד:
   i. Employee_id- המספר המזהה של העובד.
   ii. Employer_id- זהו מצביע משותף ששמור בתוכו המספר המזהה של החברה.
   iii. Salary- שכר של העובד.
   iv. Grade- הדרגה של העובד.

2. **employeesTree**- זהו עץ RANKED AVL שמכיל מידע על כל העובדים אשר שכרם גדול מאפס בלבד והוא ממוין לפי שכר העובד הגדול בעדיפות ראשונה ולפי מספר הזהות הגדול בעדיפות שניה.
   כל צומת בעץ מכילה את המידע הבא:
   i. company_id- זהו המספר המזהה של החברה שהעובד עובד בה.
   ii. Employee_id - זהו המספר המזהה של העובד.
   iii. Salary- שכר של העובד.
   iv. Grade- הדרגה של העובד.
   v. sumOfGrades - סכום הדרגות של תת העץ הימני של כל צומת, כלומר כל הצמתים הגדולים מצומת זה (במקרה שהוא בן שמאלי הסכום יכיל את האבא של הצומת הנוכחי) בלי הדרגה של הצומת עצמה.
   vi. Rank – הדרגה של כל צומת, כאשר הדרגה מכילה כמה צמתים יש בתת העץ הימני של צומת מסוים לא כולל אותו (כל הצמתים שערכם גדול מצומת מסוים).

3. **companies**- מערך בגודל שנקבע בתחילת התכנית, כל תא במערך מכיל מצביע לחברה. מערך זה מייצג את ה- UF שעובד באמצעות כיווץ מסלולים עם איחוד לפי גודל ועצים הפוכים.
   בכל תא יש את המידע הבא:
   i. Employer _id- מספר הזהות של החברה, מספר התא שנמצאת בה בתחילת התכנית.
   ii. value- ערך החברה.
   iii. numOfEmployeesZeroSalary- מספר העובדים שבחברה ששכרם אפס.
   iv. sumOfGradesOfZeroSalary- סכום הדרגות של כל העובדים שבחברה ששכרם אפס.
   v. Employees – עץ RANKED AVL שמכיל את העובדים ששכרם שונה מאפס, הדרגות הם כפי שהוסבר למעלה.
   vi. All_employees_table: טבלת ערבול דינמית של כל העובדים של החברה כולל אלה ששכרם אפס.
   vii. Acquired_value: זהו הערך שאנו מוסיפים ל-value כאשר חברה קונה חברה אחרת (נפרט יותר בהמשך).
   viii. Owner: מזהה החברה בעלת קבוצת חברות מסוימת.

בנוסף, שמרנו שלושה משתנים למבנה numberOfCompanies, numberOfEmployees ו- sumOfGradesOfZeroSalary. ששומרים את מספר החברות הכולל במערכת, ואת מספר העובדים הכולל במערכת ואת סכום הדרגות של כל העובדים ששכרם הינו אפס בהתאמה.

סיבוכיות המקום:

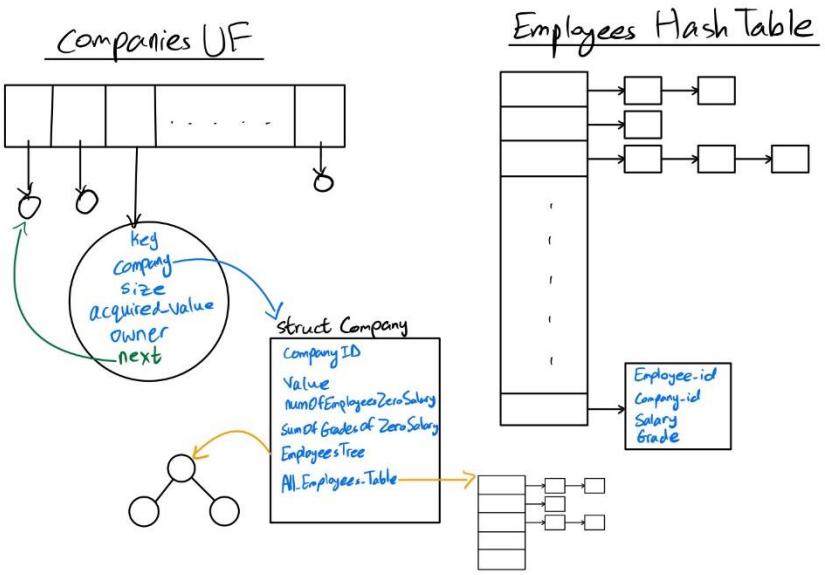
- שמרנו עץ אשר מכיל מידע על העובדים **employeesTree** שמכיל לכל היותר n צמתים (כאשר n הוא מספר העובדים הכולל במערכת), כאשר בכל צומת אנו שומרים 6 שדות, ולכן כל צומת לוקחת O(1) מקום, וסה"כ עבור העץ נקבל O(n) מקום, ולכן נקבל O(n).
- שמרנו מערך של החברות שמהווה UF, מערך זה בגודל k לכן עולה O(k) סביכיות מקום. בנוסף לכך, בכל תא בתוך מערך זה יש טבלת ערבול של העובדים של החברה ועץ של העובדים של החברה וזה עולה O(n) סביכיות מקום. ביחד יעלה O(n)+O(k)=O(n+k).
- שמרנו טבלת ערבול לכל העובדים במערכת. המימוש הינו מערך דינמי שיעלה O(n) סביכיות מקום כאשר עובד לפי שיטת ה- chain hashing למניעת התנגשיות דרך רשימה מקושרת שעולה O(1) בממוצע סביכיות מקום.
  לכן בסה"כ יעלה O(n) סביכיות מקום בלבד.

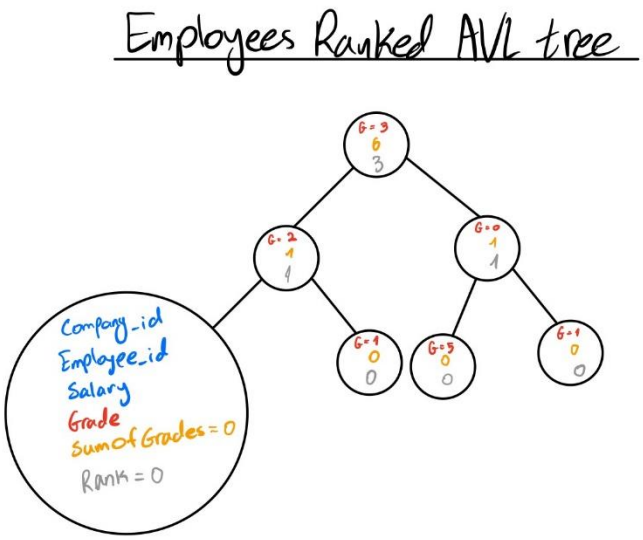
לכן סיבוכיות מקום כוללת עבור כל המבנה:

$$O(n+k)+O(n)+O(n)=O(3n+k)=O(n+k)$$

וזה אכן עומד בסיבוכיות הדרושה.

<u>נצרף תיאור גרפי למבנה:</u>





<u>תיאור הפונקציות והוכחת סיבוכיות:</u>

(נציין שלאורך הוכחת הסיבוכיות עבור הפונקציות- k הוא מספר החברות הכולל במערכת ו- n הוא מספר העובדים הכולל במערכת).

<u>הערה:</u>

1. **בפונקציות הבאות בדקנו את ערכי השגיאה ב- O(1) זמן במקרה הגרוע והחזרנו INVALID_INPUT כנדרש ומסיימים את הפונקציה.**
2. **חיפוש החברה במערך יעלה O(1) סביכיות זמן. כי כל תא במערך מהווה את המזהה של החברה כלומר התא החמישי של המערך מהווה חברה מספר 5.**
3. **קבוצה ב- UF מהווה קבוצה של חברות השייכות לאותה חברה אם.**

Void* init():

נאתחל מבנה נתונים ריק שמכיל:

- אתחול מערך שמהווה UF לחברות שבכל תא יש מצביע לחברה ב- O(k) סביכיות זמן.
- אתחול עץ העובדים ששכרם גדול מאפס ב- O(1) סביכיות זמן.
- אתחול טבלת ערבול ריקה של כל העובדים ב- ב- O(1) סביכיות זמן.
- נאתחל את השדות numberOfCompanies, numberOfEmployees ו sumOfGradesOfZeroSalary לערך 0 ב- O(1) זמן.

ולכן סה"כ סיבוכיות זמן- O(k) כנדרש.

StatusType AddEmployee(void *DS, int EmployeeID, int CompanyID, int Salary, int Grade):

נבדוק אם המזהה של העובד נמצא בטבלת הערבול של כל העובדים של המערכת בסביכיות O(1) ממוצע בשיטת chain hashing. אם קיים נחזיר שגיאה. אחרת נוסיף עובד חדש למערכת באופן הבא:

- נגדיר shared pointer עבור מספר המזהה של החברה. O(1) סביכיות זמן.
- נוסיף את האובייקט של העובד אל טבלת הערבול של כל העובדים. O(1) סביכיות זמן.
  כמו כן, נוסיף אובייקט זה לטבלת הערבול של העובדים של החברה הספציפית.
  סה"כ יעלה O(log*k) ממוצע כפי שצוין למעלה בהערות.
- נוסיף 1 ל- number_of_employees. O(1) סביכיות זמן.
- נוסיף את ערך הדרגה למשתנה sumOfGradesZeroSalary. O(1) סביכיות זמן.
- נחפש את החברה בסביכיות O(log*k) ממוצע. ואז נבצע את הפעולות הבאות:
  1. נוסיף 1 למשתנה מספר העובדים ששכרם אפס.
  2. נוסיף את הדרגה למשתנה שמכיל סכום הדרגות של העובדים ששכרם אפס.

ולסיום נחזיר SUCCESS.

סה"כ סיבוכיות זמן:

$O(1) + O(1) + O(log^*k) + O(1) + O(1) + O(log^*k) =$ משוערך בבמוצע על הקלט $O(log^*k)$ כנדרש.

StatusType RemoveEmployee(void *DS, int EmployeeID):

נבדוק אם מזהה העובד נמצא בטבלת הערבול של כל העובדים בזמן O(1) בממוצע על הקלט. אם אינו קיים נחזיר שגיאה בהתאם. אם כן קיים נמחק אותו מטבלת הערבול בזמן O(1). כשמצאנו את אובייקט העובד בטבלת הערבול, נשמור במשתנה עזר את מזהה החברה שעובד בה.

נחפש חברה זו בזמן O(1), ניגש לטבלת הערבול של העובדים של החברה ונמחק את העובד הנ"ל. בזמן O(1).

בטבלת הערבול שמרנו את החברה שהעובד עובד בה- במקרה ועשינו acuireCompany אז יישמר במקום זה את השורש של הקבוצה של החברות.

נחפש

בנוסף לכך, נחלק לשני מקרים לפי השכר של העובד:

1. אם השכר של העובד גדול מאפס, אז צריך למחוק את העובד מעץ העובדים בעלי שכר גדול מאפס ומעץ העובדים בעלי שכר גדול מאפס של החברה הספציפית שעובד בה.
   - מחיקת העובד מהעץ של כל העובדים במערכת תעלה לנו O(logn) סביכיות זמן- מחיקת מעץ מאוזן.
   - מחיקת העובד מהעץ של החברה תעלה לנו O(logn) סביכיות זמן- מחיקה מעץ מאוזן.
2. אם השכר של העובד הינו שווה לאפס, אז נבצע את הפעולות הבאות:
   - נחסיר מהמשתנה sumOfGradesZeroSalary את הדרגה של העובד שנמחק. בזמן O(1).
   - נחסיר 1 מהמשתנה שמכיל את מספר העובדים ששכרם אפס של החברה שעובד בה העובד. בזמן O(1).

   בנוסף לכך, נחסיר מהמשתנה שמכיל סכום הדרגות של העובדים ששכרם אפס את הדרגה של העובד הנמחק. בזמן O(1).

נקבל סה"כ סיבוכיות O(2logn) = O(logn) משוערך בממוצע על הקלט.

StatusType AcquireCompany(void *DS, int AcquirerID, int TargetID, double Factor):

אם הפרמטרים שקיבלנו אינם חוקיים נחזיר שגיאה בהתאם, אחרת:

נחפש את שורש הקבוצה של שבה נמצאת כל קבוצה אם שתי החברות שייכות לאותה הקבוצה נחזיר תוצאה הצלחה. סביכיות החיפוש במערך החברות הינה O(1) וחיפוש הקבוצה עבור כל חברה בנפרד עולה $O(log^*k)$ משוערך. (זמן פעולת find במבנה של ה- UF הינו $O(log^*k)$).

נעדכן את המשתנה owner של החברה בשורש הקבוצה בסיבוכיות זמן של O(1)- גישה למערך.

כעת נבצע איחוד לפי גודל בין שתי הקבוצות בזמן $O(log^*k)$ משוערך כפי שנלמד. כך שבזמן האיחוד נבצע את הפעולות הבאות (כל הפעולות הללו מדברות על השורש של העץ ההפוך):

1. תיקון הערך acquiered_value של החברה לפי פקטור באופן הבא:
   - לערך הזה של החברה שהופכת להיות השורש של הקבוצה נוסיף את ה value של ה- owner של ה- target אש מוכפל בפקטור.
   - לחברה שמהווה שורש של הקבוצה שאנו משייכים אותה לשורש הקבוצה הראשונה נחסיר אותו ערך שהוספנו בנקודה הקודמת.
2. נוסיף למשתנה שמכיל את סכום הדרגות של העובדים ששכרם אפס את המשתנה הזה אבל של החברה הנקנית. O(1) סביכיות זמן.
3. נוסיף למשתנה שמכיל את מספר העובדים ששכרם אפס את המשתנה הזה אבל מהחברה הנקנית. O(1) סביכיות זמן.
4. נתקן את המשתנה owner של החברה הנקנית כך שנשים אותו ה- owner של השורש. O(1).

אחרי זה, נעשה את שתי הפעולות החשובות:

1. במקרה שיש עובדים שיש להם שכר שונה מאפס בעץ הנקנה, נבצע merge של שני עצי העובדים של שתי הקבוצות. אופן המימוש: נשים את שני העצים בשני מערכים כאשר גודל כל מערך הינו בגודל מספר הצמתים של כל עץ בהתאמה ע"י סיור inorder.
סה"כ: סיבוכיות זמן $O(n_{acquire} + n_{target})$. כאשר n הינו מספר העובדים בכל עץ בהתאמה.
נאחד את שני המערכים הממוינים באלגוריתם merge בסביכיות $n_{acquire} + n_{target}$.
ניצור עץ חדש ומערך חדש בגודל $n_{acquire} + n_{target}$ ונעבור על העץ בסיור inorder ונכניס אליו את האיברים מהמערך הממוזג.
בסה"כ כל שלב 1 יעלה לנו: סיבוכיות זמן $O(n_{acquire} + n_{target})$.

2. ביצוע מיזוג של שתי טבלאות הערבול של שתי החברות. נעבור על כל תא במערך הקבוצה (קבוצה של חברות) ובכל תא נעבור על הרשימה המקושרת שיש בו ונוסיף כל עובד בקבוצה זו לקבוצה הקונית. נקבל סביכיות זמן של $O(n_{acquire} + n_{target})$.

סה"כ קיבלנו סביכיות זמן של: $O(n_{acquire} + n_{target} + log^{*}k)$ משוערך בממוצע על הקלט כנדרש.

StatusType employeeSalaryIncrease (void *DS, int EmployeeID, int , int SalaryIncrease):

קודם כל נחפש אם העובד נמצא בטבלת הערבול של כל העובדים שבמערכת בסיבכיות זמן של O(1), אם לא, נחזיר שגיאה בהתאם. אם כן, נבצע את הפעולות הבאות:

1. נחפש את העובד בטבלת הערבול בסביכות זמן של O(1), ונשנה את ה- salary שלו לשכר החדש שלו.
2. נשמור את החברה שהעובד עובד בה, ניקח מתוך טבלת הערבול (מצאנו בסעיף 1) ונחפש בתוך טבלת הערבול על עובד זה ונתקן לו את השכר לשכר החדש. סה"כ: *O(1)* זמן.

כעת, נחלק לשני מקרים:

1. אם ה- old_salary של העובד היה 0 אז אנחנו צריכים להוסיף אותו לעץ הראשי של כל העובדים שבמערכת עם new_salary = salary_increase וזה יעלה O(logn) סביכיות זמן- הוספת צומת חדשה לעץ מאוזן.
בנוסף לכך, ניגש לחברה שהעובד עובד בה בסביכיות של O(1) ונוסיף עובד זה לעץ העובדים עם שכר שונה מאפס. הוספת צומת חדש לעץ מאוזן עולה O(logn).
אחרי ההוספה לשני העצים, נעדכן את המשתנה sumOfGradesZeroSalary של כל המערכת כך שיהיה שווה ל- sumOfGradesZeroSalary -= grade ותוך כדי הגישה לחברה נחסיר 1 ממספר העובדים ששכרם אפס בחברה, ונחסיר את הדרגה של העובד מן המשתנה שמכיל את סכום הדרגות של העובדים ששכרם שונה מאפס בתוך החברה. שתי פעולות אלו עולות O(1) סביכיות זמן.
2. אם ה- old_salary שונה מאפס, אז נמחק את העובד עם השכר הישן מתוך העץ שמכיל כל העובדים ששכרם שונה מאפס של כל המערכת, וגם נמחק את העובד מתוך העץ של העובדים ששכרם שונה מאפס שנמצא בתוך החברה שהעובד עובד שם. זה יעלה O(logn) עבור כל מחיקה.
כעת, נוסיף לשני העצים הנ"ל את העובד בסביכיות O(logn) עבור כל עץ אבל עם השכר המעודכן שהוא new_salary = old_salary + salaryIncrease.

בסה"כ קיבלנו סביכיות זמן של O(logn)בממוצע על הקלט.

StatusType PromoteEmployee(void *DS, int EmployeeID, int BumpGrade):

נחפש את העובד עם המזהה שקיבלנו בתוך טבלת הערבול של כל העובדים שבמערכת, אם לא המצא נחזיר שגיאה בהתאם. זה יעלה O(1) סביכיות זמן.

אם BumpGrade גדול מאפס אז נבצע את הפעולות הבאות:

נחפש את העובד בתוך טבלת הערבול שמכליה כל עובדי המערכת, ונתקן בה את הדרגה שלו לדגה החדשה. זה יעלה O(1) סביכיות זמן. ניגש לחברה בסביכיות זמן של O(1). ניגש לטבלת הערבול של העובדים של החברה ונתקן בה את הדרגה. זה יעלה O(1).

נחלק לשני מקרים:

1. אם השכר של העובד הינו שווה לאפס: אז ניגש לחברה ונעדכן את האיבר (נדגיש כי משתנה זה של החברה) sumOfGradesZeroSalary כך שנוסיף לו את הערך הבא: newGrade – oldGrade בסביכיות זמן של O(1). בנוסף לכך, נעדכן את המשתנה (נדגיש כי משתנה זה הוא לכל המערכת) sumOfGradesZeroSalary +=newGrade-oldGrade.
2. אם השכר אינו אפס: אז נמחק את העובד משני העצים של העובדים כמו בפונקציה שעברה, עץ העובדים של כל העובדים ועץ העובדים של החברה. פעולה זו תעלה O(logn). אחר כך נוסיף לשני עצים אלו את העובד עם הדרגה החדשה. פעולה זו תעלה O(logn).

בסה"כ קיבלנו סביכיות זמן של O(logn) בממוצע על הקלט.

StatusType sumOfBumpGradeBetweenTopWorkersByGroup(void *DS, int CompanyID, int m, void *sumBumpGrade):

נחלק לשני מקרים:

1) *companyID > 0*:
נמצא את השורש של החברה בעלת המזהה הנתון כדי לגשת לעץ העובדים של הקבוצה שהחברה הנתונה שייכת אליה O(log*k), נבדוק אם מספר העובדים בעלי שכר שונה מאפס גדול שווה ל-m, אם זה לא מתקיים אז נחזיר FAILURE ונסיים. אחרת, כדי לחשב את הסכום נבצע את הבא ברקורסיה:
   - נתחיל מהשורש של עץ העובדים, נבדוק אם מספר העובדים בתת עץ הימני שלו שווה ל-*m* אז נחזיר את סכום הדרגות של תת העץ הימני ב-*O(1)* ונסיים.
   - אחרת, מספר הצמתים בתת העץ הימני יותר גדול מ-*m* , אז נקרא לפונקציה עם הבן הימני של הצומת.
   - אחרת, אם מספר הצמתים בתת עץ הימני יותר קטן מ-*m* אז נוסיף לסכום את סכום הדרגות של תת העץ הימני פלוס הדרגה של הצומת עצמה, נחסיר מ-*m* את מספר הצמתים של תת העץ הימני ועוד 1, ונקרא לפונקציה בחזרה על הצומת השמאלי (כל זה ב-*O(1)*).
2) *CompanyID = 0*:
נבצע את אותה פונקציה רקורסיבית שתוארה למעלה אך כעת נפעיל אותה על עץ העובדים ששומר את כל העובדים במבנה.

סיבוכיות כוללת: הפונקציה הרקורסיבית שמחשבת את סכום הדרגות עובדת בסיבוכיות *O(logn)* במקרה הגרוע, כי אנחנו עובדים על הצמתים בגובה של העץ.

סה"כ נקבל סיבוכיות *O(logn) + O(log*k) = O(logn + log*k)* כנדרש.

StatusType averageBumpGradeBetweenSalaryByGroup(void *DS, int CompanyID, int lowerSalary, int higherSalary, void *averageBumpGrade):

נחלק לשני מקרים:

1) *companyID > 0* : במקרה זה אנו צריכים לגשת לעץ העובדים השייך לחברה זו או לקבוצת החברות שהיא שייכת אליהם ולחשב את הממוצע הדרוש, ולכן נבצע את הבא:

- נבצע find על החברה הנתונה במבנה ה- *UF* כדי למצוא את שורש הקבוצה שהיא שייכת אליה, שם נשמר עץ העובדים הכולל של הקבוצה. פעולה זו עולה *O(log*k)* סיבוכיות זמן.
- נבדוק אם העובדים בעלי שכר אפס שייכים לתחום ב- *O(1)*, אם כן אז נוסיף אותם לממוצע ואז נחפש את השכר הכי נמוך בעץ העובדים ב- *O(logn)* ונתחיל לחשב את הממוצע בתחום המתחיל בשכר זה ומסתיים בשכר הגבוה הנתון.
- נמצא את הצומת הראשון שהוא בתוך התחום הנתון נקרא לו *lowNode* ואת הצומת האחרון שהוא שייך לתחום נקרא לו *highNode*, למציאתם נצטרך *2O(logn)* סיבוכיות זמן.
- אז נחשב את מספר וסכום דרגות כל העובדים הגדולים מ- *highNode* ו- מ *lowNode* בסיבוכיות *4O(logn)* באופן הבא:
  לחישוב מספר העובדים, בה"כ נרד במסלול החיפוש של *LowNode* מהשורש, אם אנו פונים שמאלה אז מוסיפים את מספר העובדים בתת עץ הימני של הצומת פלוס 1 עבור הצומת עצמו , אם אנו פונים ימינה לא נוסיף כלום, כאשר אנו מגיעים לצומת עצמה, אז נוסיף את מספר העובדים בתת עץ ימני שלה ונוסיף 1 עבורה.
  כנ"ל עבור חישוב סכום הדרגות ואותם חישובים עבור *HighNode* אך במקרה זה לא נוסיף 1 ואת הדרגה של ה- *HighNode* בחישובים כי אנו מחשבים את הסכום הגדול ממנו שלא שייך לתחום.
- לבסוף חישוב הממוצע הוא על ידי חילוק ההפרש של סכום הדרגות *LowSum-HighSum* חלקי הפרש מספר העובדים *LowNum-HighNum* ב- *O(1)* ונחזיר *SUCCESS*.

נקבל סיבוכיות סה"כ: *O(log*k) + O(logn)*.

2) אחרת, אם *companyID = 0* : נבצע את אותו הדבר כמו במקרה הראשון אך כעת אנו מתייחסים לעובדים של כל המבנה ולא רק לחברה ספיציפית.
נקבל סיבוכיות *O(logn)*.

StatusType companyValue(void *DS,int CompanyID, void *standing):

קודם כל נמצא את השורש של הקבוצה של החברה בעלת מזהה *companyID* ב- *O(log*k)* זמן חיפוש ב*UF* עם כיווץ מסלולים ואיחוד לפי גודל.

לחישוב השווי של החברה ניקח את השווי של החברה המקורי שמבקרה שלנו הוא האינדקס של החברה, נוסיף לו את ערך ה- *acquired_value* ששמרנו לו בעץ ההפוך של הקבוצה ,ונבדוק אם חברה זאת אינה השורש של הקבוצה שלה, אז נוסיף לערך זה גם את שדה ה- *acquired_value* של צומת השורש, אחרת לא צריך להוסיף אותוד. (חישובים אלה מבוצעים ב- *O(1)*)

סה"כ סיבוכיות זמן *O(log*k)* משוערך.

void Quit(void **DS):

נקרא ל*destructor* של כל המבנים השמורים במערכת שלנו, כאשר אנו צריכים למחוק את המבנים הבאים:

1) עץ העובדים הגדול שמכיל את כל העובדים במערכת – נצטרך לעבור על כל צומת ולעשות לו *delete* אז נצטרך סיבוכיות זמן במקרה הגרוע כאשר כל העובדים המערכת הם בעלי שכר גדול מאפס $O(n)$ כמספר העובדים במערכת.
2) *hashTable* של כל העובדים במערכת- הטבלה שמורה כמערך דינמי ששומר על גודל המערך להיות במקרה הגרוע ביותר $O(2n)$, נצטרך בנוסף לשחרר את הרשימות המקושרות ששמורות בכל צומת מטבלה זו, לשחרור כל הרשימות נצטרך $O(n)$ זמן כי גודל הרשימות הכולל חסום על ידי מספר העובדים. סה"כ לטבלת העובדים נצטרך $O(3n) = O(n)$.
3) מבנה ה- *UF* ששומר את החברות במבנה- יש לנו k חברות, לכל חברה אנו צריכים לשחרר את טבלת העובדים ואת עץ העובדים השייכים אליה, השחרור של כל טבלה וכל עץ יהיה כמו שתואר למעלה, נשים לב שכעת מספר העובדים הכולל במערכת שווה לסכום כל העובדים השייכים ל- k החברות, ולכן לשחרורם נצטרך $O(n)$. ולשחרור המערך ששומר את k החברות נצטרך ל- $O(k)$.

נקבל סה"כ סיבוכיות זמן : $2O(n) + O(k) = O(n + k)$ כנדרש.