מבנה נתונים יבש של רטוב 1

1. נבנה 2 מחלקות, אחת של עובד בשם Employee ואחת של חברה בשם Company.

Company:

**Key:**

CompanyID - משתנה מטיפוס int שמכיל מספר מזהה של החברה.

**Data:**

Value – משתנה מטיפוס int\* שמכיל את שווי החברה.

numEmployees – משתנה מטיפוס \*int שמכיל את מספר העובדים בחברה.

comEmpBySalary – משתנה מסוג \*void שמכיל עץ avl של משתנים מסוג Employee

שממוין שכר העובד.

comEmpByID – משתנה מסוג void\* משכיל עץ avl של משתנים מסוג Employee שממוין לפי ID של העובד.

HighestEarnerInCom– משתנה מסוג Employee\* שמצביע על העובד בעל השכר הגבוה ביותר.

Employee:

**Key:**

EmployeeID – משתנה מטיפוס int מספר מזהה יחודי לעובד.

**Data:**

EmployerID – משתנה מטיפוס int\* מצביע למספר מזהה החברה שבה העובד מועסק.

Salary – משתה מטיפוס int\* מצביע לשכר של העובד.

Grade – משתנה מטיפוס int\* מצביע לדרגת העובד.

Company – משתנה מסוג NodeCampany\* מצביע לחברה שבה העובד נמצא.

מבנה הנתונים שלנו יכיל 4 עצי avl ראשיים ומשתנה מטיפוס Employee:

1. Companies - עץ כל החברות לפי ID.
2. Employees – עץ כל העובדים לפי ID.
3. CopaniesWithEmp – עץ החברות שמעסיקות עובדים לפי ID.
4. EmployeesBySalary – עץ העובדים ממוין לפי שכר.
5. HighestEarner – משתנה מטיפוס Employee שמכיל את העובד הכי מרוויח בכל החברות ביחד.

בתוך Companies ו- CopaniesWithEmp בכל Node יש עוד 2 עצי avl:

1. comEmpBySalary עץ העובדים באותה החברה לפי שכר.
2. comEmpByID עץ העובדים באותה החברה לפי ID.

מבנה הנתונים שלנו בנוי בצורה הבאה:

ציור:

AVL TREE

AVL TREE

AVL TREE

חלק 2:

// לתקן

מימוש הפונקציות וחישוב הסיבוכיות:

StatusType AddCompany(void \*DS, int CompanyID, int Value)

סיבוכיות זמן:

הכנסת חברה חדשה לעץ avl של Companies ב log(k) כפי שנלמד בהרצאות ועדכון id+val בO(1)

// לתקן

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

סיבוכיות זמן:

חיפוש החברה ב log(k) חיפוש בעץ avl כמו שראינו בהרצאה,

הכנסת עובד לעץ המדורג לפי שכר ב log(n) במקרה הגרוע,

והכנסת העובד לעץ המדורג לפי לפי id בlog(n) במקרה הגרוע,

הכנסת העובד לעץ של כלל העובדים בlog(n) עדכון פרטי העובד בכל העצים O(1) סה״כ קיבלנו:

O(3log(n)+log(k) +1) = O(log(n)+log(k))

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

סיבוכיות זמן:

1. מוסיף את העובד לעץ Employees O(log n)
2. מחפש את החברה בעץ Companies בO(log k) ומוסיף ואת העובד לעצים comEmpByID, comEmpBySalary ב O(2log n).

StatusType RemoveEmployee(void \*DS, int EmployeeID)

סיבוכיות זמן:

1. מסיר את העובד מEmployees הסרה מעץ avl עולה O(log n)

#צריך לבדוק הסרה מ CompaniesWithEmp ומ Companies

במקרה של CompaniesWithEmp לבדוק האם צריך להסיר את החברה במידה ולא נשאר בה עובדים.

בעיה רצינית!!

StatusType RemoveCompany(void \*DS, int CompanyID)

סיבוכיות זמן:

במידה ואפשר, מסירים את החברה מהעץ Companies הסרה מעץ avl עולה O(log k)

#צריך לעדכן גם את CompaniesWithEmp

StatusType GetCompanyInfo(void \*DS, int CompanyID, int \*Value, int \*NumEmployees)

סיבוכיות זמן:

מחפש את החברה לפי ID בעץ Companies בO(log k) ומחזיר את הנתונים בהתאם.

StatusType GetEmployeeInfo(void \*DS, int EmployeeID, int \*EmployerID, int \*Salary, int \*Grade)

סיבוכיות זמן:

מחפש את העובד לפי ID בעץ Employee בO(log n) ומחזיר את הנתונים בהתאם.

StatusType IncreaseCompanyValue(void \*DS, int CompanyID, int ValueIncrease)

סיבוכיות זמן:

1. מחפש את החברה בCompanies ומעדכן בהתאם O(log n)

#צריך לעדכן גם את CompaniesWithEmp

StatusType PromoteEmployee(void \*DS, int EmployeeID, int SalaryIncrease, int BumpGrade)

סיבוכיות זמן:

1. מחפש את העובד בעץ Employees ב O(log n) ומקדם בהתאם.

#צריך לקדם גם את העובדים בתוך העץ company וCompaniesWithEmp לדבר עם יגל על איך הכי כדאי לעשות, מצביע לכל עובד או שכולם יהיה מצביעים

StatusType HireEmployee(void \*DS, int EmployeeID, int NewCompanyID)

סיבוכיות זמן:

1. נחפש את העובד בעץ Employees בO(log n) ונשמור את המשתנים של העובד במשתנה מקומי .
2. נסיר את העובד מהעץ Employees בO(log n).
3. נחפש את החברה הישנה, בעץ Companies, בO(log k).
4. נסיר את העובד מ2 העצים comEmpByID, comEmpBySalary בO(log n) במקרה הגרוע.
5. נעשה AddEmployee על הנתונים של המשתנים המקומיים שאספו מהעובד שהוסר, O(log n + log k)

סה״כ O(4log(n) + 2log(k)) = O(log n +log k)

#צריך לעדכן גם את CompaniesWithEmp

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

סיבוכיות זמן:

מעדכן את שווי החברה רוכשת בO(1)

1. ניצור 2 מערכים של Employee, אחד בגודל החברה הרוכשת, ואחד בגודל החברה הנרכשת.
2. נעבור על העץ של Companies בO(log k) נמצא את החברה הנרכשת.
3. נעבור על העץ של Companies בO(log k) נמצא את החברה הרוכשת.
4. נכניס את 2 עצי הavl של comEmpBySalary של הרוכשת והנרכשת למערכים ממוינים בעזרת מעבר על העץ in-order.
5. נאחד את 2 המערכים הממוינים שקיבלנו בסיבוכיות ב (n + ntarid )O בעזרת merge-sort.
6. ניצור עץ avl בעזרת המערך הממויין מנקודה 5 בעזרת BST ב (n + ntarid )O (כי מספר התאים במערך המאוחד הוא סכום הצמתים בעצי avl)
7. נשמור את העץ שנוצר בצד כמשתנה זמני mergeEmp.
8. נעשה אותו תהליך לעצים comEmpByID.
9. נצור אובייקט מסוג company כאשר השדות של העצי avl של העובדים מצביעים על העצים שיצרנו.
10. נעדכן את הvalue של האובייקט שיצרנו
11. נעדכן את מספר העובדים של האובייקט שיצרנו
12. נבדוק מי העובד עם השכר הגבוהה ביותר על ידי השוואה בין העובדים עם השכר הגבוהה ביותר בשני החברות ונעדכן בהאובייקט שיצרנו.
13. נמחק את החברה הנרכשת והרוכשת מהעץ Companies.
14. במידה ומספר העובדים בחברה הרוכשת הוא 0 ובחברה הנרכשת לא 0 נמחק את החברה הנרכשת מהעץ CopaniesWithEmp ונכניס את הצומת שיצרנו גם לעץ CopaniesWithEmp.
15. אם מספר העובדים בחברה הרוכשת אינו 0 וגם בנרכשת נמחק את שניהם מהעץ CopaniesWithEmp ונכניס את הצומת.
16. אם החברה הרוכשת לא 0 והנרכשת כן 0 נחמק את הרוכשת מהעץ CopaniesWithEmp ונכניס שוב.

#צריך לעדכן גם את CompaniesWithEmp

StatusType GetHighestEarner(void \*DS, int CompanyID, int \*EmployeeID)

סיבוכיות זמן:

CompanyID<0 יוחזר HighestEarner בO(1)

CompanyID>0 נעבור על העץ Companies בO(log(k)) נמצא את החברה ונחזיר את HighestEarnerInCom.

StatusType GetAllEmployeesBySalary(void \*DS, int CompanyID, int \*\*Employees, int \*NumOfEmployees)

סיבוכיות זמן:

**אם CompanyID<0:** נעבור בinorder על העץ, EmployeesBySalary בO(n) סה״כ נכניס למערך ונדפיס.

**אם CompanyID>0:** נעבור על העץ של Companies בlog(k) נמצא את החברה, ואז נעבור על comEmpBySalary בinorder O(n subscript C o m p a n y end subscript) נכניס למערך תוך כדי המעבר ונחזיר את המערך.

סה״כ: O(n subscript C o m p a n y end subscript+k)

StatusType GetHighestEarnerInEachCompany(void \*DS, int NumOfCompanies, int \*\*Employees)

סיבוכיות זמן:

נעבור על העץ של CompaniesWithEmp – בO(log k) במקרה הגרוע בו כל החברות מעסיקות עובדים, ונמצא את החברה עם הID הנמוך ביותר (הכי שמאלית). נתקדם ממנה לפי האלגוריתם הבא:

1. יהי v הצומת עם הId המינימלי.
2. count=0
3. הכנס את highSalaryEmployee לEmployee[count]
4. אם count==NumOfCampanies-1 עצור.
5. אם לצומת v יש בן ימני, עבור על תת העץ שהוא השורש שלו בinorder
6. בכל צומת שנתקל בתת עץ הימני של v נקדם את count ונעדכן את Employee[count].
7. אם count==NumOfCampanies-1 עצור.
8. אם עוד לא עברו על מספיק חברות, tmpV =v v=parent(v)
9. אם הבן הימני של v==tmpV נחזור ל(6)
10. אם הבן הימני של v!=tmpV נחזור ל(4)

סה״כ האולגוריתם עולה לנו O(NumOfCompanies) כי עברנו רק על החברות הקטנות ביותר עד שעצרנו.

וקיבלנו סה״כ O(NumOfCompanies+log(k)).

StatusType GetNumEmployeesMatching(void \*DS, int CompanyID, int MinEmployeeID, int MaxEmployeeId, int MinSalary, int MinGrade, int \*TotalNumOfEmployees, int \*NumOfEmployees)

סיבוכיות זמן:

**אם :CompanyID<0**  נחפש בעץ של Employees בlog(n) כמו שראינו בהרצאה את העובד עם הMaxEmployeeId נעבור שוב בlog(n) ונחפש את העובד עם הMinEmployeeId, העץ מסודר לפי EmployeeID ולכן נתקדם מהמינימום עד למקסימום ב O(TotalNumOfEmployee) בצורה הבאה:

1. יהי v הצומת עם הId המינימלי.
2. אם v הצומת עם הId המקסימלי, עצור.
3. אם לצומת v יש בן ימני, עבור על תת העץ שהוא השורש שלו בinorder
4. בכל צומת שנתקל בתת עץ הימני של v נבדוק האם היא מתאימה לתנאים ונקדם בהתאם.
5. אם מצאנו את הId המקסימלי, נעצור.
6. אם לא מצאנו, tmpV =v v=parent(v)
7. אם הבן הימני של v==tmpV נחזור ל(4)
8. אם הבן הימני של v!=tmpV נחזור ל(2)

בצורה כזו, ניגשנו אך ורק לעובדים שברשימה ולכל עובד בדיוק פעם אחת בלבד. ולכן סה״כ קיבלנו:

O(log(n)+TotalNumOfEmployees)

**אם CompanyID>0:** נחפש את החברה בעץ של Companies בlog(k) לפי ID ואז נבצע בדיוק את אותו האלגוריתם שביצענו בCompanyID < 0 על העץ comEmpByID. זה יעלה לנו:

O(log(n subscript c o m p a n y end subscript)+TotalNumOfEmploees)

וסה״כ נקבל: O(log(k)+log(n subscript c o m p a n y end subscript)+TotalNumOfEmploees)

// לתקן

:void Quit(void \*\*DS)

סיבוכיות זמן:

מחיקת עץ הEmployees בO(n) מחיקת כל העובדים בתוך הavl של החברות ((cmpEmpBySalary,cmpEmpByID בO(2n) ומחיקת עץ Compenies ב

O(k)

סה״כ O(3n+2k) = O(n+k)