רטוב 1:

הסבר על מבני הנתונים הבסיסיים שהשתמשנו בהם :

המבנה הבסיסי:

עץ AVL :

, בסיבוכיות זמן log(n)השימוש העיקרי במבנה הזה הוא על מנת לנצל את התכונה של חיפוש . הזנה והסרת איברים בעץ AVLשל

מימשנו מילון גנרי בעל מפתח וערך גנריים באמצעות עץ ה-AVL , נשתמש בהם בטיפוסים שונים לטובת המימוש שלנו .

המבנה העיקרי :

כולל שני עצים עיקרים :

1.עץ של חברות:

שכל מפתח בעץ הוא המספר המזהה של החברה (ID) וה data של כל קבוצה הוא טיפוס בשם COMPANY שכולל :את הid של החברה, מספר העובדים שיש בה ומצביע על העובד עם ה SALARY הכי גבוה , בנוסף כולל עץ של רק העובדים השייכים לחברה לכל עובד בעץ זה המפתח -הוא טיפוס שכולל את ה ID של העובד ואת הSALARY שלו . כלומר העץ מסודר לפי סידור יורד ל SALARY של כל עובד (אם ה SALARY של שני עובדים שונה אז סידור לפי SALARY אם יש להם את אותו SALARY אז מסודרים לפי ה IDS שלהם בסדר עולה) בנוסף לסידור עולה של ה ID שלו.

והDATA-הוא מצביע על טיפוס של EMPLOYEE שכולל מצביע לחברה ששייך אליה העובד את ה ID שלו ואת ה SALARY שלו .

2. עץ של עובדים :

העץ הזה כולל כל ה עובדים שיש בכל החברה המפתח לכל איבר בעץ הוא ה ID של כל העובד וה-DATA הוא מצביע על טיפוס של EMPLOYEE שכולל מצביע לחברה ששיך אליה העובד את ה ID שלו ואת ה SALARY שלו.

3. עץ של החברות הלא ריקות :

מכיל את הקבוצות הלא ריקות , הטיפוס COMPANY מכיל שדה בוליאני שמסמן

אם התווספו עובדים לקבוצה זו , אם כן אנחנו מוספים את החברה לעץ החברות הלא ריקות . המפתח וה DATA של העץ הזה בדיוק כמו עץ החברות.

נפרט עבור כל פונקציה את הסיבוכיות של המימוש שלה :

Init :

אתחול של מצביע למבנה ריק שכולל עצים ריקים לכן זה o(1).

AddCompany(void *DS, int CompanyID, int Value); :

תחילה נצור אובייקט של COMPANY ריק עם **CompanyID** הנתון – O(1)

נבדוק אם האובייקט שיצרנו נמצא בעץ החברות - O(log K) כאשר K מספר האיברים בעץ החברות (חיפוש בעץ AVL).

אם כן נמצא נמחק את האובייקט ונחזיר שגיאה -O(1)

אם לא נמצא אז נוסיף אותו -O(log K) כאשר K מספר האיברים בעץ החברות (הוספת איבר בעץ AVL).

בסה"כ הוספת איבר לעץ החברות, כפי שלמדנו זה O(log k) במקרה הגרוע, כאשר k הוא מספר החברות (האיברים בעץ).

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

תחילה נצור אובייקט של EMPLOYEE ריק עם**EmployeeID** **Salary** ו**Grade** -אתחול מצביעם וערכים O(1)

נבדוק אם קיימת קבוצה עם **CompanyID** הנתון -O(1)

נבדוק אם האובייקט של EMPLOYEE שיצרנו נמצא בעץ העובדים - O(log N) כאשר N מספר האיברים בעץ העובדים (חיפוש בעץ AVL).

אם כן נמצא נמחק את האובייקט ונחזיר שגיאה -O(1)

אם לא נמצא אז נוסיף אותו -O(log N) כאשר N מספר האיברים בעץ העובדים (הוספת איבר בעץ AVL).

נוסיף אותו לעץ העובדים של הקבוצה -O(logK)+ O(*S* log *NUMOFEMPLOYEE*) כאשר K מספר האיברים בעץ החברות (חיפוש הקבוצה בעץ הקבוצות-O(logK) ו*NUMOFEMPLOYEE מספר* העובדים *השייכים לחברה* (הוספת איבר בעץ AVL-O(log *NUMOFEMPLOYEE*).

ע"י בדיקה לשדה הבוליאני של הקבוצה ,נעדכן אותו בהתאם , אם זיהו העובד הראשון המתווסף לקבוצה זו נעדכן אותו לtrue ונוסיף את הקבוצה לעץ החברות הלא ריקות –

O(log *NUMOCOMPANYS*)

*** נשים לב שלעץ החברות הלא ריקות מתקיים : (K מספר כל החברות במערכת (ריקות+לא ריקות))(*NUMOCOMPANYS השייכים לעץ הקבוצות הלא ריקות):*

NUMOCOMPANYS <=K

השוואה ועדכון במידת הצורך האיבר בעל הSalary_ הכי גדול בעץ של החברה ועדכון מספר האיברים בעץ-O(1)

*** נשים לב שלכל חברה מתקיים : (N מספר כל העובדים)(*NUMOFEMPLOYEE S השייכים לקבוצה):*

NUMOFEMPLOYEE<= N

בסה"כ הוספת עובד, זה

O(log k) + O(log N)+O (log *NUMOFEMPLOYEE*)+ O(log *NUMOCOMPANYS*)<=

O(log k) + O(log N) + O(log N)+ O(log k) <= O(log k) + O(log N)

במקרה הגרוע, כאשר k הוא מספר החברות(האיברים בעץ)ו N הוא מספר כל העובדים .

:RemoveCompany(void *DS, int CompanyID);

מחפשים את האיבר שיש להסיר בעץ של החברות -O(log N) כאשר N מספר האיברים בעץ החברות (חיפוש בעץ AVL).

אם לא נמצא נחזיר שגיאה -O(1)

נבדוק אם יש עובדים בחברה זו אם כן נמצא נחזיר שגיאה -O(1)

מוחקים את האיבר מעץ החברות של החברה ומעדכנים מספר החברות במערכת (log O(*NUMOFCOMPANYS*

סכ"ה:

O(log N)+ O(log N)+ O(1)+ O(1)+ O(log *NUMOFCOMPANYS*)<= O(log N)

במקרה הגרוע, כאשר N הוא מספר כל החברות.

RemoveEmployee(void *DS, int EmployeeID);

מחפשים את האיבר שיש להסיר בעץ של העובדים -O(log N) כאשר N מספר האיברים בעץ העובדים (חיפוש בעץ AVL).

אם לא נמצא נחזיר שגיאה -O(1)

ניגשים לעץ של העובדים השייכים לחברה של השחקן דרך המצביע שמחזיק העובד בעץ העובדים לחברה שלו -O(1) (כי לא עשינו חיפוש לקבוצה בעץ אלא שמרנו מצביע ונגשנו באמצעתו)

מוחקים את האיבר מעץ העובדים של החברה ומעדכנים מספר העובדים בחברה (log O(*NUMOFEMPLOYEE*

השוואה ועדכון במידת הצורך האיבר בעל הSalary_ הכי גדול בעץ של החברה ועדכון מספר האיברים בעץ של העובדים השייכים לקבוצה -O(1)

מוחקים את האיבר מעץ העובדים של כל המערכת -O(log N) כאשר N מספר האיברים בעץ העובדים

עדכון מספר האיברים בעץ של העובדים -O(1)

סכ"ה:

O(log N)+ O(log N)+ O(1)+ O(1)+ O(log *NUMOFEMPLOYEE*)<= O(log N)

במקרה הגרוע, כאשר N הוא מספר כל העובדים.

GetCompanyInfo(void *DS, int CompanyID, int *Value, int *NumEmployees);

מחפשים את האיבר שיש להחזיר מידע עליו בעץ של החברות -O(log N) כאשר N מספר האיברים בעץ החברות (חיפוש בעץ AVL).

אם לא נמצא נחזיר שגיאה -O(1)

נעדכן במצביעים הנתונים המידע המבוקש -O(1)

סכ"ה:

O(log N)+ O(1)+ <= O(log N)

במקרה הגרוע, כאשר N הוא מספר כל החברות.

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

מחפשים את האיבר שיש להחזיר מידע עליו בעץ של העובדים -O(log N) כאשר N מספר האיברים בעץ העובדים (חיפוש בעץ AVL).

אם לא נמצא נחזיר שגיאה -O(1)

נעדכן במצביעים הנתונים המידע המבוקש -O(1)

סכ"ה:

O(log N)+ O(1)+ <= O(log N)

במקרה הגרוע, כאשר N הוא מספר כל העובדים.

IncreaseCompanyValue(void *DS, int CompanyID, int ValueIncrease); :

מחפשים את האיבר שיש לעדכן בעץ של החברות -O(log N) כאשר N מספר האיברים בעץ החברות (חיפוש בעץ AVL).

אם לא נמצא נחזיר שגיאה -O(1)

מחפשים ומעדכנים את הVALUE של האיבר מעץ החברות של המערכת (O(log *NUMOFCOMPANYES*

סכ"ה:

O(log N)+ O(1) + O(log *NUMOFCOMPANYES*)<= O(log N)

במקרה הגרוע, כאשר N הוא מספר כל החברות.

PromoteEmployee(void *DS, int EmployeeID, int SalaryIncrease, int BumpGrade);

מחפשים את האיבר שיש לעדכן בעץ של העובדים -O(log N) כאשר N מספר האיברים בעץ החברות (חיפוש בעץ AVL).

אם לא נמצא נחזיר שגיאה -O(1)

מחפשים ומעדכנים את הSALARY של האיבר מעץ העובדים של המערכת (O(log *NUMOFEMPLOYEES*

במידה ו**0< BumpGrade** אז מעדכנים בהתאם הדרגה שלו

סכ"ה:

O(log N)+ O(1) + O(log *NUMOFEMPLOYEES*)<= O(log N)

במקרה הגרוע, כאשר N הוא מספר כל העובדים.

HireEmployee(void *DS, int EmployeeID, int NewCompanyID);

מחפשים את העובד שיש להעביר בעץ של העובדים -O(log N) כאשר N מספר האיברים בעץ החברות (חיפוש בעץ AVL).

בודקים אם **NewCompanyID** שווה למזהה החברה שלו אם כן נחזיר שגיאה -O(1)

אחרת מסירם את העובד מעץ העובדים של החרה שלו O(log N כאשר N מספר האיברים בעץ החברה (הסרה בעץ AVL). N חסום ע"י מספר כל העובדים במערכת

מחפשים החברה החדשה בעץ החברות O(log K)

ומוסיפים אותו לעץ העובדים של החברה החדשה O(log N כאשר N מספר האיברים בעץ החברה (הסרה בעץ AVL). N חסום ע"י מספר כל העובדים במערכת

ס"כ:

O(log N)+ O(log K)

GetHighestEarner(void *DS, int CompanyID, int *EmployeeID);

אם CompanyID שלילי אז מעדכנים את הערך שמחזיק השדה של highestSALARYEMPLOYEE במשחק ב EmployeeID אם לא נמצא ערך נעדכן -1 בEmployeeID -O(1)

אחרת מחפשים את החברה המתאימה בעץ החברות של המערכת -O(log K) כאשר K מספר האיברים בעץ החברות (חיפוש איבר בעץ AVL).

אם לא נמצאת נחזיר שגיאה -O(1)

אחרת מעדכנים את הערך שמחזיק השדה של highestSALARYEMPLOYEE בחברה בעלת המזהה **CompanyID ב EmployeeID** -O(1)

סכ"ה:

O(1)+ O(1)+ O(1) + O(log K) <= O(log K)

במקרה הגרוע, כאשר K הוא מספר הקבוצות .

GetAllEmployeesBySalary(void *DS, int CompanyID, int **Employees, int *NumOfEmployees);

אם **CompanyID** שלילי אז בודקים אם יש עובדים במערכת אם לא מעדכנים את המצביעים בהתאם - O(1)

אם יש עובדים אז עושים סיור INORDER (שמחזיר אותם מסודרים) על עץ העובדים ושומרים אותם במערך הנתון . -O(N) כאשר N מספר האיברים בעץ העובדים

אם **CompanyID** איננו שלילי אז מחפשים את החברה המתאימה בעץ החברות -O(logK)

עושים סיור INORDER (שמחזיר אותם מסודרים) על עץ העובדים של החברה ושומרים אותם במערך הנתון (*NUMOFEMPLOYEES*)O- כאשר *NUMOFEMPLOYEES מספר העובדים השייכים לחברה*

במידה ויש שגיאה (אי מציאת ה *חברה* בעלת הID הנתון) מחזירים שגיאה --O(1)

סכ"ה:

אם ***<CompanyID*** 0 אז $O(n)$ במקרה הגרוע, כאשר n הוא מספר העובדים במערכת.

אחרת, $O(n_{NUMOFEMPLOYEES} + \log k)$ במקרה הגרוע, כאשר $n_{NUMOFEMPLOYEES}$ הוא מספר השחקנים ששייכים לקבוצה בעלת המזהה ***CompanyID*** ו-k הוא מספר החברות.

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

נשתמש בעץ החברות הלא ריקות , באמצעות סיור INORDER על העץ רק עד קבלת **numOfCOMPANYs** מהאיברים נשמור אותם במערך של COMPANYES בגודל **numOfCOMPANYs**.
O(**numOfCOMPANYs**)

עוד מעבר על המערך לקבלת האיבר בשדה highestSALARYEMPLOYEE ונשמור את ה ID של העובד הזה במערך הנתון (אחרי הקצאה נכונה למערך זה). O(**numOfCOMPANYs**)

בס"ה: O(**numOfCOMPANYs**)

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

חיפוש את שתי החברות בעץ החברות של המערכת -2*O(log K) כאשר K מספר האיברים בעץ החברות (חיפוש איבר בעץ AVL).

אחרי חיפוש נעשה סיור INORDER על כל עץ ונשמור את הערכים במערכים- $O(n_{COMPANY} + n_{replacement})$

סיור ה INORDER מחזיר אותם ממוינים , נעשה מיזוג בין המערכים ממיונים כפי שהוצג בתרגול מספר 5

-O(N*log k) כאשר N שווה לסכ"ה עובדים בשני העצים ו K הוא מספר המערכים שעושים להם מיזוג כלומר 2 לכן זה

$O(n_{COMPANY} + n_{replacement})=O(N)$

בסכ"ה נקבל: $O(\log k + n_{COMPANY} + n_{replacement})$ במקרה הגרוע.

StatusType GetNumEmployeesMatching(void *DS, int CompanyID, int MinEmployeeID, int MaxEmployeeId,

int MinSalary, int MinGrade, int *TotalNumOfEmployees, int *NumOfEmployees);

אם **CompanyID** שלילי אז בודקים אם יש עובדים במערכת אם לא מעדכנים את המצביעים בהתאם - O(1)

אם יש עובדים אז עושים סיור INORDER (שמחזיר אותם מסודרים) על עץ העובדים ושומרים אותם במערך הנתון . -O(N) כאשר N מספר האיברים בעץ העובדים

אם **CompanyID** איננו שלילי אז מחפשים את החברה המתאימה בעץ החברות -O(logK)

עושים סיור INORDER (שמחזיר אותם מסודרים) על עץ העובדים של החברה ושומרים אותם במערך הנתון (*NUMOFEMPLOYEES*)O- כאשר *NUMOFEMPLOYEES מספר העובדים השייכים לחברה*

במידה ויש שגיאה (אי מציאת ה *חברה* בעלת הID הנתון) מחזירים שגיאה --O(1)

בשני המקרים לעיל אנחנו נעבור על המערך ונספור את העובדים המקיימים התנאם לעיל נעדכן המספר במצביעים הנתונים .

סכ"ה:

אם ***<CompanyID*** 0 אז $O(n)$ במקרה הגרוע, כאשר n הוא מספר העובדים במערכת.

אחרת, $O(n_{NUMOFEMPLOYEES} + \log k)$ במקרה הגרוע, כאשר $n_{NUMOFEMPLOYEES}$ הוא מספר השחקנים ששייכים לקבוצה בעלת המזהה ***CompanyID*** ו-k הוא מספר החברות.

:Quit(void **DS)

מאחר והקצנו $O(k+n)$ מקום, עלינו לעבור על כל המקום כדי לשחרר אותו. לכן סיבוכיות הזמן היא בהתאם $O(k+n)$.