חלק יבש:

<u>הטיפוסים בהם נשתמש :</u>

Employee – מבנה המכיל מידע עבור עובד מסויים:

- Id - מספר מזהה של העובד
- Company id – המספר המזהה של החברה בה הוא עובד
- Grade- דרגת העובד
- Salary – משכורת העובד

Company - מבנה המכיל את המידע על חברה :

- Id– מספר מזהה של החברה
- Num of employees – מספר העובדים בחברה
- Value - שווי החברה
- Employees by salary- עץ avlrank המחזיק nodes עם data = employee
העץ מסודר לפי המשכורות של העובדים בחברה בסדר יורד . ובנוסף יחזיק את דרגת העובד כאשר rankaslead=grade וrank יחזיק את סכום הדרגות של תת העץ מתחת לצומת.
- Employees_with_zero_salary - שדה המחזיק את מספר העובדים בחברה עם משכורת 0.
- Sum_of_zero_employees_grade – סכום הדרגות של כל העובדים עם משכורת 0 בחברה.
- Family –
- Employees – טבלת ערבול המחזיקה מצביעים לכל העובדים בחברה
- val_to_add_them_to_family_members – מספר ממשי שהוא כמה צריך להוסיף הפעולת הכיווץ עבור מישהוא באותה קבוצה
- val_that_i_should_substract_from_myself - מספר ממשי שווה למס' שצריך להחסיר אותו מערך חברה מסוימת כך שהוא ערך החברה שקנתה אותה שהיה לפני שקנתה אותה

מבנה הנתונים **City** יכיל בתוכו:

- Num_of_companies – מספר החברות במבנה
Num_of_employees- מספר העובדים במבנה
- Employees_with_zero_salary - שדה המחזיק את מספר העובדים במבנה עם משכורת 0.
- Sum_of_zero_employees_grade – סכום הדרגות של כל העובדים עם משכורת 0 במבנה.
- Employees_with_zero_salary - שדה המחזיק את מספר העובדים במבנה עם משכורת 0.
- allEmployees – טבלת ערבול המחזיקה מצביעים לכל העובדים במבנה
- Tree of companies – עץ avl המחזיק בו nodes עם data=company
העץ מסודר לפי id של כל החברות בסדר עולה.
- Employees by salary- עץ avlrank המחזיק nodes עם data = employee
העץ מסודר לפי המשכורות של העובדים בחברה בסדר יורד . ובנוסף יחזיק את דרגת העובד כאשר rankaslead=grade וrank יחזיק את סכום הדרגות של תת העץ מתחת לצומת.
- Companies - מבנה unionfind המחזיק מצביעים לכל החברות במבנה.

AVLRankTree – עץ חיפוש בינארי עם דרגות:

- עץ אשר תומך בפעולות הבסיסיות הנדרשות כגון : הוספת צומת , הסרת צומת , גלגולים (כפי שנלמד בהרצאה) איטרציה על צמתי העץ, איחוד עם עץ נוסף ועוד דברים שפורטו בתאור הפעולות הנדרשות
- העץ מבצע את הפעולות הנדרשות בסיבוכיות זמן כפי שנדרש בתרגיל – החישוב וההסבר מפורטים בתאור הפעולות הנדרשות.
- דרגות העץ יהיו הדרגות עצמם של העובדים (grade) וכל צומת בעץ תחזיק את סכום הדרגות בתת העץ של אותה צומת.

HashTable – טבלת ערבול עם תאי מערך המחזיקים מצביעים לרשימות מקושרות:

- המבנה יממש את הפעולות שנלמדו בהרצאה : insert, remove ,find
- בנוסף המבנה יחזיק מערך דינמי של רשימות מקושרות , את גודל המערך ואת כמות האיברים סך הכל במבנה.
- פונקציית הערבול תהיה: h(x)=xmodm כאשר m הוא מספר התאים במערך הדימני ו x הוא המזהה של האיבר שנוסף.
- כל הפעולות מתבצעות בזמן O(1) כפי שנלמד.
- הגדלת המערך מתבצע בפקטור עומס a = 1 כאשר a =n/m כאשר n הוא מספר האיברים במבנה
- הקטנת המערך כאשר a=0.5

LinkedList – מבנה מסוג רשימה מקושרת:

- יחזיק בתוכו שרשרת של nodes שבכל אחד מהם יהיה data ומצביע לאיבר הבא ברשימה
- המבנה יתמוך בהוספת איבר חדש מחיקת איבר ומציאת איבר בשרשרת.

UnionFind –

- המבנה יממש מה שלמדנו כמובן union,find , כך שהמימוש המתאים ישתמש בפעולת כיווץ כדי להגיע לסיבוכיות log*k בפונקציות המתאימות לזה ,
- במימוש יהיה מצביעים להורה של קבוצת חברות , וצערך מצביעים לחברות
- בפעולה find ו union יתבצע בנוסף עדכונים לval של החברה כך שבאיחוד הקבוצות נעדכן את val_that_i_should_substract_from_myself של החברה שקנינו אותה
- ובפעולת find בפעולת הכיווץ בעיקרון דומה לזה שהוצג בתרגול בשאלה הארגזים נעדכן את הval של company

<u>הפעולות הנדרשות למימוש :</u>

void* Init(int k) – יצירת מבנה נתונים מטיפוס City

אתחול המבנה city על ידי יצירת מבנה מסוג unionfind של k חברות יצירת עץ דרגות ריק של עובדים המסודר לפי שכר העובדים , יצירת טבלת ערבול ריקה שתחזיק את כלל העובדים שנוסיף למבנה. סיבוכיות זמן לאתחול כל שדה מלבד unionfind של החברות יהיה O(1) , סך הכל מספר סופי וקבוע של פעולות. בנוסף אתחול המבנה של החברות ידרוש k פעולות כאשר k הוא מספר החברות הדרושות במבנה .

סך הכל פעולות : c*O(1) + c*O(k)

סיבוכיות הזמן : O(k) במקרה הגרוע

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

הוספת עובד חדש למערכת

ראשית ניצור אובייקט חדש מסוג shared_ptr<Employee> עם הנתונים אותם קיבלנו,

עבור יצירת האובייקט החדש ויצירת הnode הפעולות יתבצעו ב O(1) .

לאחר מכן נוסיף את העובד החדש לטבלת הערבול המחזיקה את כל העובדים במערכת , כפי שנלמד בהרצאות סיבוכיות זמן הריצה של פעולת Insert היא O(1) כאשר בכל m פעולות הכנסה נצטרך להגדיל את טבלת הערבול ולהעביר את כל העובדים לטבלה החדשה , סך הכל n פעולות כאשר n/m=1 בזמן ההגדלה.

כפי שנלמד בתרגול סיבוכיות הזמן המשוערכת על פעולות insert O(1).

לאחר מכן נמצא את החברה אליה עתיד להשתייך העובד על ידי חיפוש במבנה המחזיק את כל החברות מסוג UnionFind כאשר סיבוכיות הזמן לחיפוש בו היא log* k כאשר k הינו מספר החברות הכולל במבנה , לאחר שנמצא את החברה נוסיף לתוך טבלת הערבול שמחזיקה את העובדים בה את העובד .

סך כל הפעולות : c*1 +log*(k)

סך הכל סיבוכיות זמן משוערך על הקלט: O(log*k)

– StatusType RemoveEmployee(void *DS, int EmployeeID)

ראשית נמצא את העובד בטבלת הערבול המחזיקה את כל העובדים במערכת סך הכל פעולות למציאת איבר בטבלת ערבול כפי שנלמד : O(1)

לאחר שמצאנו את העובד ניגש לשדה בו המחזיק מצביע לחברה בה הוא נמצא , במקרה שהמשכורת של העובד גדולה מ0 הוא יימצא בעץ העובדים של החברה ושל המבנה הכללי וכדי להסיר עובד מעץ AVL נצטרך לעשות חיפוש בעץ שדורש c*logn פעולות ולאחר מכן להסיר את העובד ולתקן את העץ במקרה הצורך : logn פעולות במקרה הגרוע .

בנוסף בלי קשר למשכורתו של העובד יהיה עלינו להסירו מטבלת הערבול של החברה ושל המבנה המחזיקה את כל העובדים. כאשר מחיקת איבר מטבלת ערבול דורשת O(1) בממוצע על הקלט מכיוון שמציאתו והסרתו דורשים מספר c קבוע של פעולות ובממוצע כל m פעולות מחיקה נצטרך להקטין את הטבלה . כפי שלמדנו בתרגול סיבוכיות הזמן המשוערכת להקטנת טבלת ערבול תהיה גם היא O(1)

סך הכל פעולות : c*logn

סיבוכיות הזמן בממוצע על הקלט : O(log n).

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

ראשית נמצא את העובד אותו יש לעדכן מתוך טבלת הערבול של כל העובדים במערכת, סך הכל O(1) פעולות בממוצע על הקלט , וכעת נסיר את העובד מעץ העובדים הכללי ומטבלת הערבול וניצור עובד חדש עם ערך grade המעודכן. את העובד החדש נוסיף לטבלת הערבול ועץ העובדים , סך הכל c*logn (כאשר n הוא מספר העובדים במערכת) פעולות לפעולות הנדרשות בעץ ועוד c*O(1) עבור הפעולות בטבלת הערבול. לאחר מכן ניגש לחברה אליה שייך העובד בעזרת המצביע של העובד על החברה ונבצע את אותן הפעולות עבור טבלת הערבול של עובדי החברה ועץ העובדים של החברה.

סך כל הפעולות ב:c*logn +c*1

סיבוכיות הזמן בממוצע על הקלט : O(logn)

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

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

ראשית נמצא את העובד אותו יש לעדכן מתוך טבלת הערבול של כל העובדים במערכת, סך הכל O(1) פעולות בממוצע על הקלט , לאחר שמצאנו את העובד נעדכן את שכרו ונוסיף אותו לעץ העובדים הכללי של המבנה, סך כל הפעולות הנדרשות בממוצע להכנסת עובד לעץ הדרגות: c*logn כאשר n הוא מספר העובדים במערכת.

לאחר שהוספנו את העובד לעץ העובדים של המבנה ניגש לחברה אליה שייך העובד על ידי המציע לחברה אותו העובד מחזיק. ובחברה של העובד נבצע הוספה לעץ העובדים של החברה באופן דומה לפעולה שעשינו בעץ העובדים של המבנה.

סך כל הפעולות : c*logn

סיבוכיות הזמן בממוצע על הקלט : O (log n)

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

עבור CompanyID=0 :

ניגש לעץ העובדים של החברה ונבדוק האם מספר העובדים עם משכורת גדולה או שווה לm

אם כן נקרא לפונקציית עזר רקורסיבית שתחזיר את סכום הדרגות של m העובדים עם המשכורת הגבוהה בעץ. מכיוון שלכל צומת בעץ יש את הgrade של העובד ובנוסף דרגת הצומת היא סכום כל grades בתת העץ של אותה צומת. כל שנותר לעשות הוא לסכום את הסכומים הנדרשים. הפונקציה הרקורסיבית תהיה עם תנאי עצירה כזה שאם תת העץ של הצומת הנתון שווה לm תחזיר את דרגתו אחרת תבדוק האם תת העץ של הבן השמאלי גדול מm אם כן תקרא לפונקציה שוב עם הבן השמאלי של הצומת הנוכחי ומ , אחרת תחזיר את הערך של דרגת תת העץ השמאלי יחד עם הgrade של הצומת הנוכחי ועוד ערך ההחזרה של הפונקציה הרקורסיבית שתקרא עם הבן הימני של הצומת ומ חדש שיהיה m המקורי פחות כמו הצמתים בתת העץ השמאלי של הצומת הנוכחי +1 .

סך הכל קריאות רקורסיביות log n משוערך כגובה העץ כאשר n הוא מספר העובדים בעץ ובכל קריאה c פעולות קבועות .

סיבוכיות הזמן המשוערכת : O(logn)

עבור CompanyID>0 :

נבצע חיפוש במבנה unionfind המחזיק את החברות , סך הכל log*k פעולות כפי שנלמד , כאשר k הוא מספר החברות במבנה. לאחר שמצאנו את החברה נבצע את אותן פעולות שביצענו על עץ העובדים של המבנה עבור עץ העובדים של החברה.

סך הכל פעולות : c*log n + log*k

סיבוכיות הזמן המשוערכת : O(logn +log*k)

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

תחילה נבדוק הפרמטרים , אם מזהה החברה שלילי או המשכורת הקטנה גדולה מהמשכורת הגדולה אז נחזיר INPUT_INVALID , אחרת:

אם המזהה של החברה הוא 0 אז העץ הרלוונטי כאן הוא עץ העובדים הכללי , אם לא אז העץ הוא העובדים של החברה הרצויה ואז צריך למצוא את החברה דרך FIND של UNIONFIND , אם אין עובדים בכלל או בחברה המסוימת אז נחזיר FAILURE(מסתתרת כאן פעולת FIND לחברה שהיא LOG*(K)) אחרת :

(כרעיון נסתכל על העץ כמערך מסודר באופן יורד לפי המשכורת (אם המשכורת שווה אז בסדר יורד לפי המזהים))נמצא את האינדקס של העובד בעל המשכורת הכי גבוהה שנצא בתחום הרצוי ונסמן אותו ב NUMFIRST(נשתמש בשדות של AVLRANK ואז הסיבוכיות LOG(N) כאשר N מס' צמתי העץ ,נמצא את האינדקס של העובד בעל המשכורת הכי קטנה שנמצא בעץ (אותו דבר מבחינת סיבוכיות כי יש כאן סוג של סימטריות) , נחסיר אותו ממס' הצמתים הכללי בעץ ונסמן את תוצאת החיסור ב NUMLAST

אם NUMLAST קטו מNUMFIRST צריך להחזיר FAILURE כי אז אין עובדים בתחום)

ניקח את סכום דרגות העובדים היותר גדולים מ NUMFIRST מבחינת משכורת ונסמן אותו בFIRST

ניקח את סכום דרגות העובדים היותר קטנים מ NUMLAST מבחינת משכורת ונסמן אותו בlast

גם הפעולות האלה כל אחת מתבצעת ב LOG(N)

נסמן SUM תוצרת חיסור FIRST מ LAST וDIVIDER את תוצאת חיסור NUMFIRST מ NUMLAST- O (1)

, אם LOWERSALARY שווה 0 צריך להוסיף סכום דרגות של העבודים בעלי משכורת 0 ומספרם גם- O (1)

ואז רק נשאר לחלק כדי לקבל הממוצע , להדפיס אותו ולחזיר SUCCESS

בסה"כ : O(LOG(N)+O(1)+O(LOG*K) ולכן LOG(M)+LOG*(K) (משוערך עם הפונקציות שיש בהן פעולות הFIND וה UNION)כאשר M הוא מס' העובדים הכללי ו K הוא מס' החברות שזה מה שרצוי

StatusType companyValue(void *DS, int CompanyID)

נבדוק אם המזהה לא חיובי או גדול ממס החברות אז נחזיר INPUT_INVALID

נבדוק אם החברה שהמזהה שלה companyID אם נמצאת נמשיך אחרת נחזיר FAILURE

אם נמצאת אז דרך פעולת ה FINDשל ה UNIUNFIND הערך של החברה התעדכן וניקח את החברה עצמה המקורית (ולא שחוזרת דרך FIND בד"כ שהיא יכולה להיות האב של קבוצת חברות מסוימת)

נדפיס את ערכה ונחזיר SUCCESS

פעולת ה FIND כפי שיודעים מההרצאה והתרגול פי מה שהיה בשאלת הארגזים (פעולת הכיווץ) תתבצע בסיבוכיות (K)*LOG וK הוא מס' החברות בנוסף למס' קבוע של פעולות ולכן בסה"כ מה שרצוי

(K)*LOG משוערך

void Quit(void **DS)

נשחרר את מצביע המבנה כולו שהיה (ואחר-כך תתבצע קריאה אחרי סגירת המערכת לכל ההורסים ויתבצע שחרור של כל המבנים)

בסוף עבור סיבוכיות המקום יכולים לראות שעבור כל עובר שמרנו מבנה שלו בחברה וגם באופן כללי כך שיכול להיות שיהיה שמור שתי פעמים פעם בטבלת הערבול ופעם בעץ ולכן יש לנו עבור כל *N* העובדים בסה"כ *O(4N)* ועבור חברות כל חברה בעלת מבנה אחד ולכן *K*

לכן : ***O(4N)+O(K)<=O(4(K+N))<=O(K+N)***

כפי שרצוי