מבני נתונים כללים שממישנו

HashTable: מחלקה של טבלת עירבול אשר תחזיק את העובדים לפי תעודות הזהות שלהם, תערבל לפי פונקציית מודולו כמו שלמדנו בהרצאה. כל איבר הוא shared_ptr לרשימה מקושרת של עובדים.
בטבלה נתחזק שדה בשם internSumGrades שיחזיק את סכום הgrades של כל מי שמשכורתו היא 0.

RankTree: עץ דרגות שיכיל את כל העובדים שמשכורתם גדולה מ-0 ממויין לפי המשכורת ובעדיפות שניה לפי המזהה.

בכל צומת v נחזיק 2 שדות נוספים:

- sumGrade - סכום הgrade של כל העובדים בתת העץ ששורשו הוא v.
- rank - סכום כמות הצמתים בתת העץ ששורשו הוא v.

UnionFind: מחלקה של UF בשיטת עצים הפוכים וכיווץ מסלולים כמו שלמדנו בהרצאה. המחלקה תכיל מערך של UFNode (חוליה עבור כל חברה) ואת גודל המערך, כלומר את כמות החברות הקיימות
בכל חולייה v נשמור:

- מצביע לUFNode בשם father: יצביע לחוליה הבאה בעץ ההפוך.
- מצביע לחברה בשם company: לכל חוליה נשמור את החברה שלה.
- מצביע לחברה בשם master: אם v הוא השורש של העץ ההפוך אז נשמור את חברת האם.
- משתנה מסוג double בשם r: נשתמש בו בשביל לחשב את הvalue של חברה בסיבוכיות $log * k$ כפי שלמדנו בתרגול (בעית גובה הארגזים).

טיפוסים

EmployeeManager – מחלקת האב של כלל מבנה הנתונים, עם השדות הפנימיים הבאים:

- companies - מבנה מבוסס UF שיחזיק את החברות כפי שתואר למעלה.
- employeesTree - עץ דרגות שיכיל את כל העובדים שהמשכורת שלהם גבוהה מאפס.
- employees - טבלת ערבול שתכיל את כל העובדים הקיימים במערכת

Company - מחלקה עם השדות הפנימיים הבאים:

- id - מזהה החברה
- paidEmployees - עץ דרגות שיכיל את כל העובדים בחברה שהמשכורת שלהם גבוהה מאפס
- allEmployees - טבלת ערבול שתכיל את כל העובדים של החברה.

Employee – מחלקה עם השדות הפנימיים הבאים:

- **id** – מזהה העובד.
- **salary** – המשכורת של העובד.
- **grade** - הדרגה של העובד.
- **employer** – מצביע לחברה המעסיקה את העובד.

מימוש הפונקציות כולל ניתוח סיבוכיות

$init(k)$:

יצירת המבנה EmployeeManager, ואתחול השדות על ידי יצירה של:

- UnionFind בשם companies עם k איברים , נקרא ל- $makeset()$ לכל חברה $O(k)$
- עץ דרגות ריק בשם employeesTree $O(1)$
- טבלת ערבול בשם employees $O(1)$

סיבוכיות זמן: סך הכל $O(k)$

$addEmployee$:

- בדיקה האם הקלט תקין וזריקת חריגה אם צריך $O(1)$
- חיפוש האם כבר קיים עובד בעל מזהה EmployeeID ב- Employees $O(1)$
- נמצא את המזהה של חברת האם בעזרת $find()$ של UnionFind ונשמור מצביע לחברה בשדה employer של העובד $O(log * (k))$
- יצירת אובייקט Employee חדש עם שכר 0 ודרגה grade והכנסה שלו ל-employees $O(1)$
- נוסיף את העובד לטבלת ערבול של החברה בה הוא עובד. $O(1)$

סיבוכיות זמן: סך הכל $O(log * (k))$

$removeEmployee$:

- בדיקת תקינות הקלט וזריקת חריגה במקרה הצורך $O(1)$
- מציאת העובד בטבלת הערבול לפי מזהה העובד, אם לא נמצא נזרוק חריגה $O(1)$
- הסרת העובד מטבלת הערבול של <u>המערכת</u> $O(1)$
- אם לעובד הייתה משכורת 0 נחסיר את דרגתו מהשדה internSumGrades של <u>המערכת</u> $O(1)$
- ניגש אל המעסיק שלו דרך המצביע ונסיר את העובד מטבלת הערבול שבתוך החברה $O(1)$
- אם המשכורת שלו גדולה מ0 נסיר את העובד גם מעץ העובדים שבתוך החברה $O(log\ n)$
- אם המשכורת שלו גדולה מ0 נסיר העובד מעץ הרמות employeesTree $O(log\ n)$

סיבוכיות זמן: סך הכל $O(log\ n)$

$acquireCompany$:

- נבדוק את תקינות הקלט ובמקרה הצורך נזרוק חריגה $O(1)$
- נמצא את השורש של החברה <u>הרוכשת</u> בCompanies ונסמנו בa $O(log\ *\ (k))$
- נמצא את השורש של החברה <u>הנרכשת</u> בCompanies ונסמנו בt $O(log\ *\ (k))$
- נבדוק האם החברות נמצאות תחת אותה חברת אֵם, אם כן נזרוק חריגה. $O(1)$
- נבצע $union(a, t)$, הפעולה זוכרת כי הפרמטר הראשון הוא החברה הרוכשת ותעדכן את השדה master בשורש החדש להצביע לחברת האם. $O(log\ *\ k)$
- בזמן האיחוד, נעדכן את שדות ה-R של החברות כפי שראינו בתרגול עם בעיית הגובה של הארגזים $O(1)$
- נבצע מיזוג של עצים מאוזנים (אלגוריתם והוכחת סיבוכיות בעמוד האחרון) עבור עץ הדרגות שבכל חברה ונשמור את העץ המאוחד בחברה הרוכשת. תוך כדי האלגוריתם של מיזוג העצים, נעדכן את השדה employer של כל עובד להיות החברה הרוכשת. $O(n_{AcquirerID} + n_{TargetID})$.
- נבצע סיור $postOrder$ על כל העץ הממוזג ונעדכן בהתאם את הrank ואת הsumGrade בצמתים, סיבוכיות זמן סיור זה היא כמספר האיברים בעץ כלומר $O(n_{AcquirerID} + n_{TargetID})$.
- נעבור על טבלת הערבול בחברה ה<u>נרכשת</u> ונסיר כל עובד משם ונבצע הכנסה לטבלת הערבול של החברה ה<u>רוכשת</u> $O(n_{TargetID})$

סיבוכיות זמן: סך הכל $O(log\ *\ k\ + n_{AcquirerID} + n_{TargetID})$

$employeeSalaryIncrease$:

- נבדוק את תקינות הקלט ובמקרה הצורך נזרוק חריגה $O(1)$
- נמצא את העובד בטבלת הערבול, אם לא קיים נזרוק חריגה $O(1)$
- נעדכן את שדה ה-salary של העובד להיות $salary\ =\ salary\ +\ SalaryIncrease$ $O(1)$
- אם לפני ההעלאה היה לעובד שכר גדול מ-0:
  - נסיר אותו מעץ העובדים <u>בחברה</u>. $O(log\ n)$
  - נסיר אותו מעץ העובדים <u>במערכת</u>. $O(log\ n)$

  (נוסיף אותו שוב לאחר מכן וזאת על מנת לעדכן את מיקומו בעצים)
- ניגש לחברה המעסיקה דרך השדה employer ונכניס את העובד לעץ העובדים <u>בחברה</u>, נשים לב כי מספר העובדים בחברה חסום ע"י מספר העובדים הכולל במערכת ולכן $O(log\ n)$
- נכניס את העובד לעץ העובדים <u>במערכת</u> $O(log\ n)$

סיבוכיות זמן: סך הכל $O(log\ n)$

$promoteEmployee$:

- נבדוק את תקינות הקלט ובמקרה הצורך נזרוק חריגה $O(1)$
- נמצא את העובד בטבלת הערבול, אם לא קיים נזרוק חריגה $O(1)$
- אם הפרמטר bumpGrade אינו חיובי ממש, לא נבצע כלום ונחזיר SUCCESS $O(1)$
- נחלק למקרים: אם המשכורת של העובד גדולה מ-0:
  - הסרה של העובד לעץ הרמות employeesTree $O(log\ n)$
  - ניגש למעסיק דרך העובד ונבצע הסרה של העובד מעץ הרמות paidEmployees $O(log\ n)$
  - נעדכן את השדה grade של העובד להיות $grade = grade + BumpGrade$ $O(1)$
  - הוספה של העובד לעץ הרמות employeesTree $O(log\ n)$
  - ניגש למעסיק דרך העובד ונבצע הוספה של העובד לעץ הרמות paidEmployees $O(log\ n)$
- אם העובד הוא בתקופת חפיפה כלומר המשכורת שלו היא 0:
  - נעדכן את השדה grade של העובד להיות $grade = grade + BumpGrade$ $O(1)$
  - נוסיף את bumpGrade לשדה internSumGrade של טבלאת הערבול של <u>המערכת</u> וגם של <u>החברה</u> $O(1)$

סיבוכיות זמן: סך הכל $O(log\ n)$

$sumOfBumpGradeBetweenTopWorkersByGroup$:

- נבדוק את תקינות הקלט ובמקרה הצורך נזרוק חריגה $O(1)$
- במידה ו- $CompanyID > 0$:
  - נמצא את חברת האם של החברה ב- Companies $O(log\ *\ (k))$
  - נבדוק האם מספר העובדים בחברה עם משכורת גדולה מ-0 קטן מ-m, אם כן נזרוק חריגה. נעשה זאת ע"י שדה הגודל שאנו מתחזקים בעץ paidEmployees $O(1)$
  - נחפש בעץ הדרגות paidEmployees של החברה המעסיקה, אחר העובד בעל $rank = n - m$ (בהכרח קיים כי $m \leq n$ ואם נחפש rank=0 אז נסכום את כל העץ בדרך) ובמהלך הדרך אל הצומת הזו, בכל פעם שנרצה לרדת שמאלה נסכום את הערך sumGrade של הצומת עצמה ושל הבן הימני שלה. בסוף התהליך, נקבל בסכום את הערך המבוקש $O(log\ n)$
- ואם $CompanyID = 0$ נבצע את אותו התהליך הנ"ל על employeesTree של <u>המערכת</u> $O(logn)$

סיבוכיות זמן: סך הכל $O(\ log\ *\ k + log\ n)$

$averageOfBumpGradeBetweenTopWorkersByGroup$:

- נבדוק את תקינות הקלט ובמקרה הצורך נזרוק חריגה $O(1)$
- נאתחל משתנה מסוג double בשם totalSum עם הערך 0 $O(1)$
- נאתחל משתנה מסוג int בשם numOfEmployees עם הערך 0 $O(1)$
- במידה ו- $CompanyID > 0$:
  - נמצא את חברת האם של החברה ב- Companies $O(log * (k))$
  - נבדוק האם מספר העובדים בחברה הוא אפס בעזרת המשתנה count אותו אנו מתחזקים לכל חברה. ואם אכן זה המצב נזרוק חריגה $O(1)$
  - אם הגבול התחתון של התחום הוא 0, כלומר צריך להתחשב בעובדים שבתקופת החפיפה:
    - נוסיף ל-totalSum את הערך internSumGrades השמור בטבלת הערבול של החברה. $O(1)$
    - נוסיף לnumOfEmployees את כמות העובדים בתקופת חפיפה (ניתן לחישוב על ידי ההפרש בין גודל הטבלת ערבול לבין עץ העובדים בעלי משכורת גדולה מ0). $O(1)$
  - נחשב את הRank של העובד בעל המשכורת המינימלית בתחום הנתון ע"י סיור בינארי בעץ הדרגות (האלגוריתם מתואר אחרי הפונקציה) ונסמן את דרגתו ב- minRank $O(log\ n)$
  - באותו אופן נחשב את ה-Rank של העובד בעל המשכורת המקסימלית בתחום ונסמן את דרגתו ב- maxRank $O(log\ n)$
- במידה ו- $CompanyID = 0$: נבצע את אותם הפעולות על העץ דרגות וטבלת הערבול של המערכת
- נגדיר 2 משתנים מסוג int, בשם sum1, sum2 ונבצע השמות:

$$sum1 = SumOfBumpGradeBetweenTopWorkersByGroup(companyID,\ size - minRank)$$
$$sum2 = SumOfBumpGradeBetweenTopWorkersByGroup(companyID,\ size - maxRank)$$

כלומר נקרא לפונקציה הקודמת שסוכמת את הgrades של m העובדים המרוויחים ביותר. סיבוכיות הזמן של פונקציה זו היא $O(\ log\ *\ k + log\ n)$

- נוסיף לtotalSum את סכום הgrades של העובדים בעלי שכר גדול מ-0 שנמצאים בתחום הנתון, כלומר:

$$O(1)\ totalSum\ += sum1 - sum2$$

- נוסיף לnumOfEmployees את כמות העובדים בעלי שכר גדול מ-0 שנמצאים בתחום הנתון, כלומר

$$O(1)\ numOfEmployees\ += maxRank - minRank$$

- נבדוק האם $numOfEmployees = 0$, כלומר לא קיימים עובדים בחברה או במערכת בתחום הנתון ולכן נזרוק חריגה $O(1)$
- נחשב את הממוצע $\frac{totalSum}{numOfEmployees}$ ונחזיר אותו $O(1)$

סיבוכיות זמן: סך הכל $O(\ log\ *\ k + log\ n)$

האלגוריתם למציאת הדרגה של העובד בעל המשכורת המינימלית (אנלוגי למציאת המקסימלי):

באופן רקורסיבי נבצע את הפעולות הבאות עבור צומת נתונה בעץ:

- אם הצומת היא NULL - נחזיר אפס

- אם השכר של העובד הנוכחי גבוה או שווה ל-lowerSalary נחזור על האלגוריתם עבור הבן השמאלי
- אחרת נגדיר nRank =1

ונבדוק האם קיים בן שמאלי, אם כן נוסיף ל- $nRank$ את המשתנה Rank ששמור בבן השמאלי.
ונחזיר nRank ועוד הערך המוחזר מהפעלת האלגוריתם עבור הבן הימני

<u>הסבר לסיבוכיות:</u> מכיוון שבכל צומת האלגוריתם בוחר אם להמשיך בתת העץ של הבן ימני או בתת העץ של הבן השמאלי מתקיים כי האלגוריתם מתבצע מספר פעמים כגובה העץ. כלומר $log\ n$ פעמים , ומכיוון שהאלגוריתם מבצע מספר קבוע של פעולות עבור כל צומת מתקיים סך הכל כי הסיבוכיות היא $O(log\ n)$.

$CompanyValue$:

האלגוריתם שהשתמשנו בו מבוסס על האלגוריתם שהוצג בתרגול של עידו - בבעיה של חישוב גובה הארגזים. הפתרון כלל שמירה של ערך R בכל חוליה של UF כך שאם נבצע סכימה של ערכים אלו מהחוליה המבוקשת ועד השורש נקבל את הגובה של הארגז מהרצפה, בנוסף דאגנו לעדכן בהתאם ערכים אלו כאשר מתבצע כיווץ מסלולים.

כאשר A רוכשת את T:

נסמן את ערכה של החברה T ב- $TValue$ ואת ערכה של החברה A ב- $AValue$ ואת הפקטור הנתון ב-$factor$.
כתלות בגודל הקבוצות המתאימות ל-A ול-T, בדקנו האם מדובר **במקרה רגיל** - בו החברה הנרכשת תצביע לחברה הרוכשת (בעץ ההפוך של הUF) או **במקרה ההפוך** שבו החברה הרוכשת תצביע לחברה הנרכשת. ועידכנו את ערכי ה-R באופן הבא:

מקרה רגיל:

$$R_A = R_A + factor * TValue$$

$$R_T = R_T - R_A$$

מקרה הפוך:

$$R_A = R_A + factor * TValue - R_T$$

- ניגש לUFNode של החברה המבוקשת, נסכום את השדה R בכל צומת עד שנגיע לשורש ונחזיר את הסכום בתור ה-value של החברה.

סיבוכיות זמן: סך הכל $O(log * k)$

$Quit$:

- נעבור על כל העובדים בטבלת הערבול ונשחרר אותם, לאחר מכן נשחרר את כל הרשימות המקושרות ולבסוף את המערך הדינאמי. $O(n)$
- נעבור על כל העובדים בעץ העובדים של המערכת בסיור PostOrder ונשחרר אותם $O(n)$
- באותו אופן נעבור על מערך החברות ב- companies ועבור כל חברה נעבור על עץ העובדים שלה ועל טבלת הערבול שלה ונשחרר את כל העובדים שלה ולאחר מכן נשחרר את החברה.
מכיוון שכל עובד מופיע בעץ ובטבלת ערבול של חברה <u>אחת בלבד</u> (דאגנו לכך כאשר עושים Acquire) ומספר העובדים בכל החברות חסום ע"י מספר העובדים בכל המערכת מובטח שסיבוכיות הזמן של שחרור <u>כל</u> החברות יהיה $O(k + n)$.
- נשחרר את מחלקת האב של כלל מבנה הנתונים. $O(1)$

סיבוכיות זמן: סך הכל $O(k + n)$

סיבוכיות מקום של כל המערכת:

מבני הנתונים במערכת שמוקצים דינאמית:

- $companies$: UF המכיל k חברות:
  לכל חברה i מבין k החברות:
  - עץ דרגות עם n_i עובדים לכל היותר (אם כולם לא בחפיפה)
  - טבלת ערבול עם n_i עובדים

מכיוון שדאגנו שכל עובד מופיע רק בחברה אחת מתקיים: $\sum_{i=1}^{k} n_i \leq n$ ולכן סיבוכיות המקום $O(n + k)$

- employeesTree: עץ דרגות המכיל n עובדים $O(n)$
- employees: טבלת ערבות המכילה n עובדים $O(n)$

סך הכל נקבל כי סיבוכיות מקום של כל המערכת היא $O(n + k)$ כנדרש.

אלגוריתם למיזוג עצים מאוזנים, מקבל 2 עצים ומחזיר עץ ממוזג (שומש ב-AcquireCompany)

- נבצע סיור inorder על עץ א' ונשמור במערך בגודל n_1 , $O(n_1)$
- נבצע סיור inorder על עץ ב' ונשמור במערך בגודל n_2 , $O(n_2)$
- נבצע מיזוג מערכים ממויינים לתוך מערך בגודל $n_1 + n_2$, $O(n_1 + n_2)$
- נבצע אלגוריתם רקורסיבי הבונה עץ מאוזן בהינתן מערך ממוין $O(n_1 + n_2)$:

האלגוריתם קובע את מרכז המערך להיות השורש ואז שולח לריקורסיה אחת את חצי המערך הימני ולריקורסיה נפרדת את החצי השמאלי, לכן אם הוא סיבוכיות הזמן אז מתקיים:

$$T(n) = 2T\left(\frac{n}{2}\right) + C$$

C הוא הזמן הקבוע שלוקח לנו לחשב את אמצע המערך ולחבר את השורש לתת העץ השמאלי ולתת העץ הימני. לפיכך אם נחשב סיבוכיות בעזרת הצבה חוזרת נקבל:

$$T(n) = 2^{log_2 n} \cdot T(0) + C = n + C$$

לכן הסיבוכיות הזמן היא $O(n)$ וסיבוכיות המקום היא $O(n)$ כאשר n הוא גודל המערך כנדרש.

לסיום, נחזיר את העץ הממוזג.