תרגיל בית רטוב 2 - חלק יבש

מבני נתונים הבסיסיים שמימשנו

1. רשימה מקושרת המכילה עובדים ותומכת בפעולות הבאות:
   - הכנסה
   - חיפוש
   - הוצאה
2. טבלת ערבול דינמית מבוססת על Chain Hashing המכילה עובדים ותומכת בפעולות הבאות:
   - הכנסה
   - חיפוש
   - הוצאה
3. עץ דרגות מאוזן המכיל עובדים וממוין על פי משכורת ואח"כ על פי id, כך שכל צומת מכיל בנוסף לעובד כלשהו את מספר העובדים בתת העץ שלו ואת סכום הדרגות של כל העובדים בתת העץ שלו. העץ תומך בפעולות הבאות:
   - הכנסה
   - חיפוש
   - הוצאה
   - מציאת איבר מינימלי
   - מציאת איבר מקסימלי
   - מציאת איבר לפי דרגה נתונה
   - מציאת סכום הדרגות של m העובדים עם המשכורת הכי גבוהה בעץ
   - מציאת מספר העובדים שהמשכורת שלהם נמצאת בטווח כלשהו של משכורות, ואת סכום הדרגות שלהם
   - מיזוג עץ דרגות חיצוני לתוך העץ הנתון

4. union find בו האיברים מיוצגים על ידי חברות והקבוצות הן חברות המאוגדות תחת חברה אחת שרכשה אותן. בכל איבר הוספנו שדה נוסף שמטרתו לעזור לנו לחשב שווי של חברה ספציפית שערכה גדל בעקבות רכישות, בדומה לשאלת הארגזים מהתרגול. בכדי לחשב שווי של חברה, נעבור על המסלול ממנה ועד השורש ונסכום את הערכים הללו. המבנה תומך בפעולות:

   - איחוד קבוצות לפי גודל תוך עדכון הערך הנוסף
   - מציאת איבר עם כיווץ מסלולים תוך עדכון הערך הנוסף
   - חישוב שווי של חברה ספציפית
   - החזרת חברה לפי מספר מזהה(ממערך החברות ב- $O(1)$)

מחלקות עזר

- Employee - המחלקה תכיל את השדות הבאים:
  - Id - מזהה של העובד
  - Salary - משכורת של העובד
  - Grade - דרגה של העובד
  - Company - מצביע לחברה שבה הוא עובד

בנוסף המחלקה מכילה getters and setters עבור השדות.

- Company - המחלקה מכילה את השדות הבאים:
  - Id - מזהה של החברה
  - Value - ערך החברה
  - num_of_interns - מספר העובדים עם משכורת 0 בחברה.
  - sum_of_interns_grades - סכום הדרגות של העובדים עם משכורת 0 בחברה.
  - Employees_with_salary - עץ דרגות כפי שתואר בהתחלה, שיכיל את העובדים בחברה שהמשכורת שלהם גדולה מ-0.
  - All_employees – טבלת ערבול המכילה את כל העובדים.

המחלקה תומכת בפעולות הבאות:

  - Getters and setters
  - הוספת עובד לחברה
  - פיטור עובד מהחברה

מבנה הנתונים הכולל שמימשנו CompanySystem

מבנה הנתונים שלנו מורכב מהמרכיבים הבאים:

- employees_with_salary - עץ דרגות (אם מאפיינים כפי שפורט קודם) שיכיל את כל העובדים עם שכר.
- all_employees – Hash Table דינמית (אם מאפיינים כפי שפורט קודם) המכילה את כל העובדים במערכת.
- companies – union find בו האיברים הם חברות והקבוצות הן חברות המאוגדות תחת חברה אחת שרכשה אותן כפי שתואר קודם.
- num_companies – מספר החברות המאותחל בתחילת התוכנית עם ערך שמקבלים.
- num_interns – מספר העובדים עם משכורת 0 בחברה.
- sum_interns_grades - סכום הדרגות של העובדים עם משכורת 0 בחברה.

התייחסות לסיבוכיות משוערכת – נשים לב כי כל הפעולות שמשתמשות במתודות של Union-find משוערכות יחדיו, ולכן שימוש במתודות הללו בתוך הפעולות של מבנה הנתונים יהיה בסיבוכיות משוערכת $O(\log^*(k))$ כאשר k הוא מספר החברות כפי שנלמד בהרצאה.
בנוסף, אנו משתמשים בטבלת ערבול דינמית שתחזיק את העובדים, אבל מפני ש-addEmployee

משוערכת יחד עם removeEmployee, אז כל פעולות ההכנסה וההוצאה יהיו בסיבוכיות משוערכת של $O(1)$ כפי שנלמד בתרגול.

<u>פעולות מבנה הנתונים</u>

init(k) – באתחול מבנה הנתונים אנו מאתחלים שני משתנים, טבלת ערבול ריקה ועץ דרגות ריק שכל אחד מהם מאותחל בסיבוכיות זמן של $O(1)$. בנוסף, אנו מאתחלים את ה-union-find של החברות ולכן צריכים לאתחל כל חברה (ואתחול כל חברה נעשה בסיבוכיות $O(1)$) לכן אתחול ה-union-find נעשה בסיבוכיות זמן של $O(k)$. לכן, סה"כ סיבוכיות הזמן של האתחול תהיה $O(k)$.

Quit() - בהפעלת ה destructor של מבנה הנתונים עוברים על כל העובדים במערכת הנמצאים ב- hash table ומוחקים אותם, סה"כ עוברים על n עובדים.
לאחר מכן נקראים כל ה- destructors של מבני הנתונים ומוחקים את מה שהקצנו בהם, בהם משחררים n איברים ב hash table, k חברות ועצי הדרגות שלהן ו- $4k$ איברים, קבוצות ושני מערכים ב- union-find.
סה"כ שיחררנו מספר קבוע של פעמים $n+k$ איברים לכן סיבוכיות הזמן היא $O(n+k)$

<u>addEmployee(int employee_id, int company_id, int grade)</u>- מקצים מקום לעובד חדש, מוצאים את החברה של העובד ב union-find -שומרים לו מצביע אליה, מכניסים אותו ל-hash table ומעדכנים את num_interns ואת sum_interns_grades של המבנה, וגם של החברה אליה הוכנס. מכניסים את העובד לתוך טבלת הערבול של העובדים בחברה.
החיפוש של החברה ב-union-find נעשה בסיבוכיות משוערכת של $O(\log^*(k))$ כפי שנלמד בהרצאות, וההוספה של העובד לתוך שתי טבלאות הערבול נעשית בסיבוכיות $O(1)$ משוערך, בממוצע על הקלט כפי שנלמד בהרצאות. לכן, סה"כ הפעולה נעשית בסיבוכיות $O(\log^*(k))$ משוערך, בממוצע על הקלט.

<u>removeEmployee(int employee_id)</u> - נמצא את העובד אותו אנו רוצים להוציא בטבלת הערבול. לאחר מכן נוציא אותו מהחברה שבה עבד יש לנו מצביע אליה לכן אפשר לגשת אליה ב $O(1)$, ובמידה והמשכורת שלו 0 נעדכן את השדות הנוספים. אם המשכורת שלו היא לא 0 אז נוציא אותו מהעץ בחברה שבה הוא נמצא, ומהעץ הכללי במבנה. לבסוף נוציא אותו מטבלת הערבול ונמחק אותו.
ההוצאה של העובד בטבלת ערבול נעשה בסיבוכיות של $O(1)$ משוערך, בממוצע על הקלט והחיפוש שלו בטבלת הערבול נעשה בסיבוכיות $O(1)$ בממוצע על הקלט כפי שנלמד בהרצאות, וההוצאה של העובד מעץ העובדים בחברה נעשית בסיבוכיות $O(\log n)$ כי עץ חיפוש מאוזן. לכן, סה"כ הפעולה נעשית בסיבוכיות $O(\log(n))$ משוערך, בממוצע על הקלט.

<u>employeeSalaryIncrease(int employee_id, int salary_increase)</u> - נמצא את העובד לו מעלים שכר בטבלת הערבול. אם היה לו שכר 0 נעדכן את השכר, מכניסים את העובד לעץ העובדים בחברה ולעץ העובדים הכללי במבנה ומעדכנים את השדות הרלוונטיים במערכת, ובחברה.
אחרת כדי לעדכן את מיקום העובד בעצים נוציא אותו מעץ העובדים הכללי ומעץ העובדים בחברה, לאחר מכן נעדכן את השכר של העובד ונכניס אותו חזרה לעצים.
החיפוש של העובד בטבלת ערבול נעשה בסיבוכיות של $O(1)$ בממוצע על הקלט כפי שנלמד בהרצאות, ההוצאה והכנסה של העובד לעץ העובדים בחברה ולעץ הכללי נעשות בסיבוכיות $O(\log n)$ כי אלו עצי חיפוש מאוזנים. לכן, סה"כ הפעולה נעשית בסיבוכיות $O(\log(n))$ בממוצע על הקלט.

<u>promoteEmployee(int employee_id, int bump_grade)</u> - נמצא את העובד לו מעלים שכר בטבלת הערבול. אם היה לו שכר 0 מעדכנים את השדה במערכת ובחברה הסוכמים את דרגות העובדים עם משכורת 0, ולאחר מכן נעדכן את הדרגה שלו.

אחרת, העובד נמצא בעץ הדרגות, ולכן נרצה לשנות את הערכים של הצמתים במסלול החיפוש שלו שסוכמים את דרגות העובדים בתת העץ שלהם. לצורך כך נוציא אותו מהעץ של החברה ומהעץ הכללי, נעדכן את הדרגה שלו, ונחזיר אותו חזרה (במהלך ההוצא וההכנסה עץ הדרגות מטפל בעדכון השדות הרלוונטיים.
החיפוש של העובד בטבלת ערבול נעשה בסיבוכיות של $O(1)$ בממוצע על הקלט כפי שנלמד בהרצאות, ההוצאה והכנסה של העובד לעץ העובדים בחברה ולעץ הכללי נעשות בסיבוכיות $O(\log n)$ כי אלו עצי חיפוש מאוזנים. לכן, סה"כ הפעולה נעשית בסיבוכיות $O(\log(n))$ בממוצע על הקלט.

<u>acquireCompany(int acquirer_id, int target_id, double factor)</u> – נמצא ב union-find את החברות שתחתן נמצאות החברות הנ"ל אם הן שוות נזרוק שגיאה. נוסיף את כל העובדים בטבלת הערבול של target לתוך טבלת הערבול של העובדים ב-aquirer (גודל הטבלה של target חסום ע"י $c \cdot n_{target}$ כאשר c קבוע כלשהו שהגדרנו, ולכן פעולה זו נעשית ב-$O(n_{target})$ פעולות בממוצע על הקלט), נמגז את עצי הדרגות של העובדים עם השכר של החברות, נעדכן את המשתנים שסוכמים מספר עובדים עם משכורת 0 ואת דרגתם בחברה הרוכשת. לבסוף נאחד את החברות ב- union-find תוך עדכון ערכי ה- fixer לחישוב שווי חברה כפי שנלמד בתרגול בבעיית הארגזים.
החיפוש של החברות ב-union-find נעשה בסיבוכיות משוערכת של $O(\log^*(k))$ כפי שנלמד בהרצאות.
מיזוג העצים: *הסבר על אלגוריתם המיזוג: נבצע סיור inorder על שני העצים ונקבל שני מערכים ממוינים. לאחר מכן נמגז את המערכים הממוינים למערך ממוין אחד, וממנו נרכיב עץ מאוזן באופן ריקורסיבי:
ניקח את האיבר באמצע המערך להיות השורש, ונרכיב עץ מאוזן עבור הבן השמאלי שלו עם כל האיברים שלפניו, ונרכיב עץ מאוזן עבור הבן הימני שלו עם כל האיברים שאחריו. בכל שלב, כתוצאה מזוגיות מספר האיברים במערך, הפרשי הגבהים בין העץ בבן השמאלי לבין העץ בבן הימני יהיה לכל היותר 1, לכן נקבל עץ מאוזן. לבסוף, נבצע סיור post-order על העץ שבו אנו מעדכנים את הגבהים של כל צומת ואת הדרגות המתאימות.
סה"כ נעבור על כל האיברים בעצים ב-inorder, במיזוג המערך נעבור על סך כל האיברים ואת העץ נרכיב רקורסיבית מהשורש, ולבסוף נבצע סיור post-order לעדכון השדות בצמתים, לכן אם $n_{acquirer}$ הוא מספר העובדים ב-acquirer ו-n_{target} הוא מספר העובדים ב-target, אז הסיבוכיות של המיזוג תהיה: $O(n_{acquirer} + n_{target})$.
עדכון המשתנים נעשה ב $O(1)$ ואיחוד החברות ב- union-find מתבצע בסיבוכיות משוערכת של $O(\log^*(k))$ כפי שנלמד בהרצאות.
לכן, סה"כ הפעולה נעשית בסיבוכיות $O(\log^*(k) + n_{acquirer} + n_{target})$ משוערך, בממוצע על הקלט.

<u>companyValue(int company_id)</u> – נפעיל את המתודה שמחשבת ערך של חברה מסוימת ב--union find המתודה עוברת על המסלול מהאיבר אל שורש העץ וסוכמת בדרך את שווי החברה ההתחלתי והערכים לחישוב השווי.
היא פועלת באופן זהה לפעולה שמחשבת גובה של ארגז מהקרקע בבעיית הארגזים מהתרגול.
לכן, כפי שראינו בתרגול סיבוכיות הפעולה תהיה $O(\log^*(k))$ משוערך.

<u>sumOfBumpGradeBetweenTopWorkersByGroup(int company_id, int m)</u>

מימשנו פונקציית עזר המוצאת איבר בעץ בהינתן הדרגה שלו מהאיבר העליון, כלומר בהינתן מספר m נחזיר את האיבר ה- m מלמעלה. נשמור משתנה sum הסוכם את מספר הצמתים שיש מעל הצומת הנוכחית ונרוץ בלולאה עד שהוא יגיע ל- m, תוך מעבר על צמתי העץ.
אם מספר הצמתים בעץ קטן מ-m נזרוק שגיאה.

בכל צומת בעץ יש לנו את מספר הצמתים בתת העץ שלה, נרד ימינה בעץ כל עד מספר הצמתים תחת הבן הימני קטן מ- m.כאשר נגיע לצומת כזו נגדיל את sum במספר הצמתים תחת הבן הימני ועוד 1 על הצומת הנוכחית. נלך לבן השמאלי ונבדוק את הבן הימני שלו, אם יש לו בן ימני נבדוק אותו, אם אין לו נוסיף את הצומת הנוכחית ל sum ונלך שמאלה. נמשיך בתהליך זה עד שנגיע לצומת בה מספר הצמתים תחת הבן הימני שלה + 1(הצומת הנוכחית) + sum יהיו שווים ל- m ואז נחזיר את הצומת הנוכחית. בהכרח נמצא את האיבר ה- m מלמעלה כיוון שהתחלנו לסכום את האיברים ב- sum מצומת בה מספר הצמתים בתת העץ גדול או שווה ל-m. במקרה הגרוע הפונקציה יורדת עד לעלה ולכן סיבוכיות הזמן תהיה $O(\log(n))$

בנוסף, מימשנו בעץ הדרגות פונקציה שמוצאת את m העובדים עם המשכורת הכי גבוהה (לפי התנאים שמפורטים). הפונקציה מוצאת את האיבר בדרגה ה-m מהסוף (כלומר האיבר המקסימלי מדרגה 1 מהסוף), ולאחר שמצאנו אותו אנחנו עוברים על מסלול החיפוש שלו ובכל שלב מבצעים את הצעד הבא: אם אנחנו הולכים ימינה, אז אנחנו ממשיכים לבן הימני שלו בלי לעשות כלום, אבל אם אנחנו הולכים לבן השמאלי שלו אנחנו סוכמים את סכום הדרגות שנמצא בבן הימני של השורש הנוכחי, ואת הדרגה של השורש הנוכחי, ואנחנו עושים זאת ב-$O(1)$ כי הפרטים הללו נמצאים בתוך הצמתים שהגדרנו.
מפני שאנחנו עוברים על מסלול החיפוש של האיבר שדרגתו m מהסוף, אנו נגיע אליו תוך כדי שסכמנו את כל הדרגות של העובדים עם משכורת גדולה יותר ממנו או משכורת שווה לו ו-id גדול יותר. מציאת האיבר בדרגה ה-m מהסוף נעשה בסיבוכיות של $O(\log(n))$, ולאחר מכן אנו עוברים על מסלול החיפוש בעץ מאוזן ומבצעים בכל שלב מספר פעולות קבוע שהסיבוכיות שלהן הוא $O(1)$ ולכן החיפוש של האיבר בעל הדרגה ה-m מהסוף יעשה בסיבוכיות של $O(\log(n))$ כפי שנלמד בהרצאות. לכן, סה"כ הסיבוכיות של פונקציה זו תהיה $O(\log(n))$.

לכן, בהתאם לערך של company_id, ניגש לעץ המתאים ונפעיל את הפונקציה שמוצאת את m העובדים עם המשכורת הכי גבוהה.
מציאת החברה עם company_id נעשה בסיבוכיות משוערכת של $O(\log^*(k))$ כפי שנלמד בהרצאה, והפונקציה נעשית בסיבוכיות של $O(\log(n))$, לכן הסיבוכיות הכוללת תהיה $O(\log^*(k) + \log(n))$ משוערך.

averageBumpGradeBetweenSalaryByGroup(int company_id, int lowerSalary, int higherSalary)

בתוך עץ הדרגות שלנו כתבנו פונקציה שמבצעת את הפעולה הדרושה באופן הבא:
נשמור את מספר העובדים בעץ ואת סכום הדרגות של כל העובדים בעץ (מידע זה נמצא בשורש לפי האופן שבו הגדרנו את צמתי העץ).
כעת נרצה למצוא העובד בעל המשכורת הנמוכה ביותר בעץ, אך גדולה או שווה מ-lowerSalary.
לכן נתחיל בשורש את התהליך הבא:
אם בצומת הנוכחית המשכורת גדולה או שווה מ-lowerSalary אז נמשיך לבן השמאלי שלו.
אחרת, העובד הזה לא נמצא בתחום, לכן נחסר 1 ממספר העובדים ששמרנו, ונחסר את ה-*grade* שלו מסכום הדרגות ששמרנו, ואם יש לו בן שמאלי נחסר את כמות העובדים בתת העץ השמאלי שלו ממספר העובדים ששמרנו כי הם גם לא נמצאים בתחום, ונחסר את סכום הדרגות שלהם, ולבסוף נמשיך לבן הימני.
נעצור כאשר נגיע ל-*null* ולאחר מכן נבצע את התהליך הסימטרי עבור *higherSalary*. בסיום התהליך, נישאר רק עם כמות העובדים שבתחום וסכום הדרגות שלהם, ונשמור אותם במשתנים אשר נתונים לפונקציה.
כל אחד מהחיפושים הוא כגובה העץ במקרה הגרוע ולכן $O(\log(n))$, ומפני שאנחנו מבצעים מספר קבוע

של חיפושים, ובתוך החיפושים כל הפעולות שאנו מבצעים הן בסיבוכיות של $O(1)$, נקבל כי הסיבוכיות הכוללת של פונקציה זו היא $O(\log(n))$.
לכן, נרצה למצוא את החברה (במידה ו-*CompanyId>0*) ולאחר מכן נפעיל את הפונקציה על העץ שבתוך החברה (או על העץ שמחוץ לחברה במידה ו-*CompanyId==0*) כאשר אנו מזינים לה את שני המשתנים של מספר העובדים וסכום הדרגות, ולבסוף משתמשים בהם על מנת לחשב את הממוצע.
סה"כ מציאת החברה נעשית בסיבוכיות משוערכת של $O(\log^*(k))$ כפי שנלמד בהרצאה, וחישוב הממוצע שלהם יעשה בסיבוכיות של $O(\log(n))$. לכן הסיבוכיות של הפונקציה תהיה:
$O(\log^*(k) + \log(n))$ משוערך.

<u>סיבוכיות מקום:</u>
במבנה הנתונים: מוקצה מקום לעץ דרגות המכיל את העובדים עם שכר, טבלת ערבול המכילה את העובדים, union-find המכיל את החברות , כל העובדים והחברות במערכת מוקצים פעם אחת לכן סה"כ כאשר c_0 הוא גודל איבר בעץ, c_1 הוא גודל עובד, c_2 הוא גודל חברה, c_3 הוא גודל איבר ב-union-find , c_4 הוא גודל קבוצה ב-union-find , בכל חברה יש עץ עובדים עם שכר וטבלת ערבול של העובדים לכן בכל החברות יש סה"כ *n* צמתים של עובדים בעצים ולכל היותר טבלאות ערבול באורך *2n*.
לכן סיבוכיות המקום של כל טבלאות הערבול היא *2n*2* והסיבוכיות הכוללת היא:

$$2n * c_0 + n * c_1 + k * c_2 + k * c_3 + k * c_4 + 2n * 2 = O(n + k)$$

במתודות: בכל המתודות מוקצה מספר קבוע של עצמים שאינו תלוי בקלט. בפונקציה acquireCompany מוסיפים עובדים לטבלת הערבול של החברה הרוכשת מה שיכול להוסיף לסיבוכיות מקום לכל היותר מערך באורך *n*(לאחר מכן בפונקציה מנקים את טבלת הערבול בחברה שנרכשה ומאתחלים אותה לגודל 4).
כמו כן בפונקציות נעשה שימוש ברקורסיה בסריקות על העצים בגודל n. לכן סיבוכיות המקום של המתודות תהיה $O(n)$.

סה"כ קיבלנו כי סיבוכיות המקום של מבנה הנתונים היא $O(n + k)$.