רטוב 2 – הסבר ופירוט עבודה

הסבר על מבני הנתונים הבסיסיים שהשתמשנו בהם:

Generic Union Find:

השימוש העיקרי באלגוריתם זה הוא ניצול העובדה שלפי המימוש הרביעי שראינו בהרצאה (איחוד לפי גודל וכיווץ מסלולים) הסיבוכיות המשוערכת לכל פעולה היא $O(\log^* k)$, מה שמאפשר לנו לקיים את דרישת סיבוכיות הזמן ברוב הפונקציות למשל בmerge נשתמש בUnion ונבצע זאת ב-$O(1)$ או כשרוצים לחפש קבוצה מסוימת אז משתמשים בFind.

Hash Table :

טבלת ערבול דינאמית של כל הלקוחות במערכת, שמכילה אובייקטים מסוג לקוח (המזוהה ע"י ת.ז ומספר טלפון השמור כמידע נוסף). טבלת הערבול מאותחלת עם גודל קבוע מראש וגדלה/קטנה בהתאם לעומס בטבלה כפי שלמדנו בתרגול והרצאות. טבלת הערבול של השחקנים מערבלת ע"פ מזהה הלקוח וממומשת בשיטת שכל תא מחזיק עץ AVL במקום רשימה מקושרת .

Tree Ranked AVL:

הסתמכנו על המימוש של ה-AVL מהתרגיל הקודם וניצלנו את זה שהחיפוש נעשה בסיבוכיות $O(log\ n)$, אבל בנוסף עכשיו יש לנו את התשלום החודשי של כל עובד שנכנס לתמונה והוספת PRIZE שנשמור כמידע נוסף.

המבנה העיקרי:

1. Union Find אחד של התקליטים:

נעשה על מערך דינאמי של מספרים מ-1 ועד number_of_records שמייצגים את התקליטים שנמצאות אצלנו במערכת,. כל תקליט במערך החברות שמורה בתא שהוא המספר המזהה שלה, ובנוסף החזקנו בכל תא מספר העותקים מכל תקליט ו מספר הרכישות של כל תקליט שמשמשים לחישובים כפי שנפרט בהמשך בתיאור הפונקציות.
כל צומת בUF שומרת שדה של מידע נוסף שיעזור לנו לחישוב הגובה והעמודה של כל צומת החישוב של הגובה כפי שנלמד בדיוק בתרגול UF והעמוד הוא שדה שמתעדכן בכל פעולה UNION לפי מה התקליט שנערם על התקליט האחר.

2. Hash Table אחד של הלקוחות:

טבלת ערבול דינמית שמכילה את כל הלקוחות ובכל תא בטבלה שומרת אותם בעץ AVL Ranked Tree במקום רשימה מקושרת , העץ ממוין לפי המזהה של כל

לקוח ושומר מידע בדיוק כמו עץ חברי המועדון המוסבר למטה בנוסף לשדה בינארי שמעיד אם הוא חבר מועדון או לא . מה שמאפשר לנו למצוא אם הלקוח נמצא או לא במימוש הפונקציות (ב-O(1) בממוצע על הקלט משוערך ובמקרה הגרוע $O(log\ n)$).

3. עץ AVL Ranked Tree:

**** כל הוספה/הסרה של אחד הצמתים בעצים אלה נעשית בסיבוכיות של $O(logn)$ כאשר n הוא מספר הלקוחות בכל המערכת****

העץ מכיל את כל הלקוחות שהם חברי מועדון , העץ הזה ממוין לפי המזהה של כל לקוח ושומר את התשלום החודשי של הלקוח וערך PRIZE בנוסף מצביע דו-כיווני למקום שלו בHASHETABLE.

נפרט עבור כל פונקציה את הסיבוכיות של המימוש שלה:

**** כל הבדיקות של תקינות הקלט נעשית בסיבוכיות $O(1)$ לכן לא נזכיר אותם בניתוח הסיבוכיות בפונקציות, כנ"ל לגבי עדכון מצביעים ואובייקטים בפונקציות ****

1. RecordsCompany()

אתחול של מצביע למבנה ריק שכולל העץ שמותחלים ב-O(1), hash ומבנה ה-Union Find ריק(בזמן $O(1)$).
אתחול של העצים וה-hash הוא בסיבוכיות של $O(1)$, לכן בסה"כ האתחול הוא בסיבוכיות $O(1)$.

2. ~RecordsCompany()

מאחר והקצנו $O(n+m)$ (n לקחות ו m תקליטים) מקום, עלינו לעבור על כל המקום ולשחרר אותו. לכן סיבוכיות הזמן היא בהתאם $O(n+m)$.

3. newMonth(int *records_stocks, int number_of_records)

מקצים מערך דינאמי שמהווה המערך של מבנה הUF שלנו ממלאים את המידע הנוסף של כל תא בnumber_of_records , עוברים על ה- HASHETABLE בכל תא מאפסים את סך הכסף שכל אדם חייב במידה והוא חבר מועדון, לבסוף עוברים על עץ חברי מועדון מאפסים את סך הכסף שכל אדם חייב.

סכ"ה: $O(n+m)$

4. addCostumer(int c_id, int phone)

בודקים את הפרמטרים תקינים בסיבוכיות של $O(1)$.

עושים חיפוש ב HASHETABLE אם הלקוח נמצא מחזירים שגיאה בסיבוכיות של $O(1)$ בממוצע על הקלט.

אחרת מוסיפים את הלקוח ומעדכנים את שדות המידע הנוסף שלו ב$O(1)$ בממוצע על הקלט משוערך.

סכ"ה: $O(1)$ בממוצע על הקלט משוערך.

5. getPhone(int c_id)

בודקים את הפרמטרים תקינים בסיבוכיות של $O(1)$.

עושים חיפוש ב HASHETABLE אם הלקוח לא נמצא מחזירים שגיאה בסיבוכיות של $O(1)$בממוצע על הקלט.

אחרת מחזיר את מספר טלפון השמור בתא ב$O(1)$.

סכ"ה: $O(1)$ בממוצע על הקלט.

6. makeMember(int c_id)

בודקים את הפרמטרים תקינים בסיבוכיות של $O(1)$.

עושים חיפוש ב HASHETABLE אם הלקוח לא נמצא מחזירים שגיאה בסיבוכיות של $O(1)$בממוצע על הקלט.

אחרת מעדכנים השדה שהוא הפך לחבר מועדון ב$O(1)$.

מוסיפים אותו לעץ חברי המועדון ומעדכנים את המצביע הדו כיווני O(log n)

סכ"ה: O(log n) במקרה הגרוע.

7. isMember(int c_id)

בודקים את הפרמטרים תקינים בסיבוכיות של $O(1)$.

עושים חיפוש ב HASHETABLE אם הלקוח לא נמצא מחזירים שגיאה בסיבוכיות של $O(1)$בממוצע על הקלט.

אחרת מחזיר את הערך השמור המעיד אם הוא חבר מועדון או לא ב$O(1)$.

סכ"ה: $O(1)$ בממוצע על הקלט.

8. buyRecord(int c_id,int r_id)

בודקים את הפרמטרים תקינים בסיבוכיות של $O(1)$.

עושים חיפוש ב HASHETABLE אם הלקוח לא נמצא מחזירים שגיאה בסיבוכיות של $O(1)$בממוצע על הקלט.

עושים חיפוש ב UF אם התקליט לא נמצא או מספר הרכישות קטן שווה מהמזהה שלו מחזירים DOESNT_EXISTS .

אחרת מעדכנים מספר הרכישות , ובמידה והלקוח הוא חבר מועדון עוברים לעץ של חברי מועדון מחפשים אותו ומעדכנים את ההוצאות החודשיות שלו.(O(log n

סכ"ה :(O(log n במקרה הגרוע.

9. addPrize(int c_id1, int c_id2, double amount)

בודקים את הפרמטרים תקינים בסיבוכיות של $O(1)$.

מחפשים בעץ חברי המועדון את הלקוח הקטן ביותר הגדול ל id1 וכמו כן מחפשים את הלקוח הגדול ביותר הקטן ל id2, לאחר מכן נמצא את האב המשותף שלהם בעץ, ונשתמש במידע הנוסף שהוא PRIZE ,כאשר עולים מהבן השמאלי ועד האב המשותף נעדכן את שדה ה-PRIZE לכל האיברים מימין, וכאשר עולים מהצומת הימני לאב המשותף נעדכן את שדה ה-PRIZE לכל האיברים משמאלו.

סיבוכיות זמן: כיוון ששמרנו את המידע הנוסף בשני העצים, כל החישובים בעצים מבוצעים אך ורק במסלול החיפוש, כפי שלמדנו בהרצאה / תרגולים ולכן סיבוכיות חישובים אלה היא *O(log n)*.

10. getExpenses(int c_id)

בודקים את הפרמטרים תקינים בסיבוכיות של $O(1)$.

מחפשים את הלקוח בעץ חברי המועדון , במידה ולא מוצאים מחזירים שגיאה ,

אחרת סוכמים את ערך ה- PRIZE מהצומת של הלקוח עד השורש כלומר במסלול בחיפוש -*O(log n)* ומחזירים הערך

סכ"ה:*O(log n)* .

11. putOnTop(int r_id1, int r_id2)

בודקים את הפרמטרים תקינים בסיבוכיות של $O(1)$.

מוצאים את התקליטם בפעולת Find לפי המזהה שלהן O(log*m), ואז משתמשים בפעולת Union של הUnion Find כדי לאחד את שתי התקליטם בנוסף לעדכון השדות הנוספים של גובה ועמודה כפי שנלמד בתרגול O(1), שתי הפעולות האלה במימוש שלנו הן בסיבוכיות משוערכת של $O(\log^* \mathrm{m})$

12. getPlace(int r_id, int *column, int *hight)

בודקים את הפרמטרים תקינים בסיבוכיות של $O(1)$.

בעזרת המידע הנוסף ששמרנו ב-UF, אנחנו יכולים לחשב ערך של התקליט פשוט ע"י קריאה ל-find וסכום הערכים לאורך המסלול בפעולת find יניב את הערך הנכון של החברה (כפי שפורט בהתחלה רעיון זה דומה לתרגיל שהועבר בתרגולים) .

סכ"ה:O(log*m)