הסבר על מבני נתונים שלנו :

המבנה הבסיסי:

1. UnionFind:

השתמשנו במבנה זה בכדי לנצל את הסיבוכיות המשוערכת של פעולות חיפוש ומיזוג שהינה O(log*m) (כפי שראינו בהרצאה , השתמשנו באיחוד לפי גודל וכיווץ לפי מסלולים) .
והתפקיד שלו הוא לאחסן את התקליטים השונים שנפרט בהמשך על זה .

2. HashTable:

השתמשנו בו בכדי לנצל את טבלת הערבול הדינמית (שראינו בהרצאה) ולנצל את הסיבוכיות $O(1)$ משוערך על ממוצע הקלט, ששמרנו בה את כל ה costumers . הטבלה מאותחלת ע"י גודל קבוע מראש והגודל שלה משתנה בהתאם לעומס בטבלה ...

הטבלה מערבלת ע"פ מזהה ה costumers וממומשת בפונקציית double hashing (שראינו בהרצאה). והשתמשנו בפונקציית הפיזור האחיד שנלמדה בהרצאה עם הקבוע $(sqrt(5) - 1) / 2$.

3. Avltree: השתמשנו באותו עץ מתרגיל בית 1

עץ AVL , השתמשו בו בכדי לשמור על סיבוכיות זמן של $O(log\, n)$ בעת הכנסה והוצאת איברים. העץ תומך בבניית עץ ריק בסיבוכיות זמן של $O(1)$. העץ מכיל משתנה ששומר את גודל העץ וגם מצביע לשורש העץ מסוג Node שהוא מכיל : מצביע לבן הימני והשמאלי ולאבא. וגם מכיל שני מצביעים (DATA,KEY) שהם מטיפוסיים גנריים.בנוסף מכיל גובה ו BF לכל צומת בשל איזון העץ. וכפי שנלמד בהרצאות סיבוכיות המקום של העץ הוא $O(n)$. שאנו צריכים בכדי לשמור על סיבוכיות של $O(log\, n)$ כשמכניסים member costumer

המבנה העיקרי:

שם של מבנה הנתונים שלנו הוא RecordsCompany שמכיל recording שהוא מטיפוס unionfind , costumerTable מטיפוס hash , ו memberCost שהוא עץ AVL.

1. unionfind :

השתמשנו ב Union של התקליטים, נעשה על מערך ל num recorded תקליטים כך שכל תקליט מכיל ה id שלו ומספר ה copies ומספר הדברים שנמכרו. נאחסן במבנה זה את התקליטים השונים הממוספרים 0 עד למספר התקליטים פחות 1 (num_of_records). ויש לנו עוד שני מערכים שהם מאותו גודל (num_of_records) "מס' התקליטים". שני מערכים אלה משמשים בעיקר לפעולות החיפוש והאיחוד של תקליטים כפי שתואר בהרצאה (השתמשנו במימוש של עצים הפוכים עם כיווץ מסלולים בעת מציאה ...)
ניתוח סיבוכיות זמן : יצירה של תקליט בודד O(1), חיפוש של תקליטים $0(log\, m)$, איחוד של תקליטים O(1). במשוערך, חיפוש ואיחוד לוקחים $O(\log * m)$.

ניתוח סיבוכיות מקום: נניח ויש במבנה m תקליטים. שמרנו במבנה זה 3 מערכים באורך כולל של 3m.
סה"כ סיבוכיות מקום: $3m = O(m)$

2. hash :

השתמשנו בו בכדי לשים בו את ה costumers שכל אחד מהם יש לו מספר מזהה, האם הוא חבר מועדון , מספר הטלפון שלו , ההוצאות שלו . הhash מאפשר לנו למצוא את ה costumer , או להוסיף costumers בזמן משוערך בממוצע על הקלט $O(1)$.

ניתוח סיבוכיות זמן :הוספה, הכנסה וחיפוש ב O(1) משוערך על הקלט כפי שנלד בהרצאה.

ניתוח סיבוכיות מקום : אם יש במבנה n לקוחות , ע"פ מסקנה מההרצאה והתרגול נזכור כי לפי כלל שינוי הגודל, אנו משנים את גודל הטבלה כאשר היא מלאה (מגדילים פי 2) או רבע ריקה (מקטינים פי 2) ולכן שימוש בטבלה דינמית מבטיח size*0.25 ≤ n ≤size כלומר נקבל שפקטור העומס הוא $O(1)$.

לכן סה"כ סיבוכיות מקום: $O(n * factor\_omess) = O(n * 1) = O(n)$

3.memberCost :

שהוא עץ AVL , המכיל את חברי המועדון והוא ממוין ע"פ מס המזהה של כל חבר מועדון. וכל איבר (צומת / חבר מועדון) מכיל ב DATA שלו את מספר מזהה, מספר הטלפון שלו , ההוצאות שלו , וגם משתנה שהוא חבר מועדון (פשוט את ה Class Customers) .ואת סיבוכיות הזמן והמקום הוסברו מקום (לעיל).

הסבר של הפונקציות :

1. RecordsCompany() :

אתחול שלושה מבנים (hash , Union , Avl עץ)באופן ריק שזה ייקח סיבוכיות $O(1)$.עבור לכל אחד מהמבנים (לפי מה שראינו בהרצאה), כלומר יוצא לנו $3O(1)$

לכן בסה"כ סיבוכיות הזמן היא $O(1)$.

2. ~RecordCompany() :

הריסת העץ Avl, כלומר משחררים את העץ ריקורסיבית בסיור inorder שזה לוקח $O(n)$

הריסת ה Hash ייקח $O(n)$

הריסת ה Union ייקח $O(m)$

כאשר m הוא מספר התקליטים ו n הוא מספר ה Costumers

כלומר ייקח $2O(n) + O(m) = O(m + n)$

לכן בסה"כ סיבוכיות הזמן היא $O(m + n)$.

עמיד חסין [redacted]

3. newMonth(int *records_stocks , int number_of_records) :

בדיקת נכונות $O(1)$

הקצאת ומילוי מערך של תקליטים ב $O(m)$

אחר כך מחקנו את מערך התקליטים ב $O(m)$

הצבענו על המערך החדש ב $O(1)$.

כניסה לכל לקוח ומאפסים את סכום הכסף שכל אדם חייב בסיבוכיות $O(n)$ דרך ה Hash משום שיש לנו n לקוחות .

וגם אפסנו עבור אלו חברי המועדון שנמצאים בעץ ב $O(log\, n_{member\_club}) \leq O(log\, n)$.

$$O(1) + 2O(m) + 2O(1) + O(n) + O(log\, n_{member\_club}) \leq 2O(m) + 2O(n) \leq O(m+n)$$

לכן בסה"כ סיבוכיות הזמן היא $O(m+n)$.

4. addCostumer(int c_id , int phone) :

בדיקת נכונות $O(1)$

בדיקת אם לקוח קיים דרך חיפוש על לוקח בטבלת הערבול (Hash) שלוקח סיבוכיות משוערך על קלט $O(1)$.

הוספת customer ל Hash בסיבוכיות $O(1)$משוערכת כפי שראינו בהרצאה .

לכן בסה"כ סיבוכיות הזמן משוערכת בממוצע על הקלט $O(1)$.

5. getPhone(int c_id) :

בדיקת נכונות $O(1)$

בדיקת אם לקוח קיים דרך חיפוש על לוקח בטבלת הערבול (Hash) שלוקח סיבוכיות משוערך על קלט $O(1)$.

מחפשים על הלקוח בעל מזהה הנתון תוך כניסה ל Hash בסיבוכיות משוערך על קלט $O(1)$

לכן בסה"כ סיבוכיות הזמן משוערכת בממוצע על הקלט $O(1)$.

6. makeMember (int c_id) :

בדיקת נכונות $O(1)$

בדיקת אם לקוח קיים דרך חיפוש על לוקח בטבלת הערבול (Hash) שלוקח סיבוכיות משוערך על קלט $O(1)$, ובמקרה הגרוע ביותר $O(\log n)$

בדיקת אם לקוח הוא כבר חבר מועדון דרך חיפוש על לוקח בטבלת הערבול (Hash) שלוקח סיבוכיות משוערך על קלט $O(1)$ ובמקרה הגרוע ביותר $O(\log n)$.

הוספת אותו לעץ (עץ של חברי מעדון)שזה ייקח $O(log\, n)$.

$$O(1) + 3O(log\, n) \leq O(log\, n)$$

לכן בסה"כ סיבוכיות הזמן היא $O(log\, n)$.

עמיד חסין [redacted]

7. isMember (int c_id) :

בדיקת נכונות $O(1)$

בדיקת אם לקוח קיים דרך חיפוש על לוקח בטבלת הערבול (Hash) שלוקח סיבוכיות משוערך על קלט $O(1)$.

בדיקת האם הוא חבר מועדון (ב Hash) , כל לקוח יש לו תכונה האם הוא חבר מועדון , אז נחפש הלקוח ב hash ונראה האם הוא חבר מועדון. שזה לוקח סיבוכיות בממוצע על הקלט $O(1)$ (כפי שראינו בהרצאה).

לכן בסה"כ סיבוכיות הזמן היא $O(1)$ בממוצע על הקלט .

8. buyRecord(int c_id, int r_id) :

בדיקת נכונות $O(1)$

בדיקת אם לקוח קיים דרך חיפוש על לוקח בטבלת הערבול (Hash) שלוקח סיבוכיות משוערך על קלט $O(1)$, ובמקרה הגרוע ביותר $O(\log n)$

בדיקת אם לקוח הוא כבר חבר מועדון דרך חיפוש על לוקח בטבלת הערבול (Hash) שלוקח סיבוכיות משוערך על קלט $O(1)$ ובמקרה הגרוע ביותר $O(\log n)$.

במידה ואם הבדיקה הקודמת (שכן הלקוח הוא חבר מועדון) אז ומוסיפים הוצאות ללקוח ע"י כניסה ללקוח שהוא חבר מועדון שנמצא בעץ Avl בסיבוכיות של $O(\log \mathrm{n})$.

הוספת רכישה לתלקיט $O(1)$

$$O(1) + O(\log \mathrm{n}) + O(\log \mathrm{n}) + O(\log \mathrm{n}) + O(1) \leq 3O(\log \mathrm{n}) \leq O(\log \mathrm{n})$$

לכן נקבל סיבוכיות של $O(\log \mathrm{n})$

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

בדיקת נכונות $O(1)$

כניסה ללקוח שהוא חבר מועדון ונמצא בעץ Avl בסיבוכיות של O(log n) ומורידים מההוצאות שלו את amount.

$$O(1) + O(\log \mathrm{n}) \leq 2O(\log \mathrm{n}) \leq O(\log \mathrm{n})$$

לכן בסה"כ סיבוכיות הזמן היא $O(log\, n)$.

10. geptExpenses (int c_id) :

בדיקת נכונות בסיבוכיות של $O(1)$

בדיקת אם הלוקח קיים בעץ שהוא עץ של חברי המועדון שזה ייקח $O(\log \mathrm{n})$.

החזרת ההוצאות של הלקוח(שהוא בהכרח חבר מועדון) ע"י חיפוש בעץ Avl בסיבוכיות של $O(\log \mathrm{n})$

$$O(1) + O(\log \mathrm{n}) + O(\log \mathrm{n}) \leq 2O(\log \mathrm{n}) \leq O(\log \mathrm{n})$$

לכן נקבל סיבוכיות הזמן של $O(log\, n)$.

11. PutOnTop (int r_id1 , int r_id2) :

בדיקת נכונות בסיבוכיות של $O(1)$

נשתמש ב find של מחלקת Union שזה $O(\log * m)$ (לפי המימוש שלנו + כפי שלמדנו בהרצאה)
אחר כן עושים איחוד ב $O(1)$

$O(1) + O(\log * m) + O(1) \leq 2O(\log * m) \leq O(\log * m)$

לכן נקבל סיבוכיות הזמן של $O(\log * m)$.

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

בדיקת נכונות $O(1)$

נמצא את התלקיט המתאים ב $O(\log * m)$ (כפי שראינו בהרצאה)דרך מחלקת ה Union .

לכן נקבל סיבוכיות של $O(\log * m)$