מבני נתונים (234218) רטוב #2 – חלק יבש

מגישים:

- **שם:** דניאל מודריק **ת.ז:** 212243257
- **שם:** ניקיטה ליסוקון **ת.ז:** 332684190

תאריך הגשה:

13/6/2021

תכנון המערכת:

- בשביל לממש את המערכת תוך כדי עמידה בתנאי סיבוכיות, נבחר להשתמש במבנה הנתונים **UnionFind** בכדי לשמור את סוכנויות הרכב, ואז נוכל לחפש ולאחד אותן ב- log*(n) משוערך.
- כל סוכנות רכב בUnionFind תהיה מטיפוס **Dealership** שמכיל **עץ דרגות** אשר ממוין לפי מספר המכירות של כל סוג רכב, ו**עץ AVL** רגיל אשר ממוין לפי מספר המזהה של כל סוג רכב (ובעזרתו נוכל למצוא כמה מכירות בוצעו עבור טיפוס רכב מסויים).
- את UnionFind נממש בעזרת מערך דינאמי שיכיל מצביע לכל איבר וגודל קבוצתו, ואת הקבוצות עצמן נממש עם עצים הפוכים שמצביעים לאבותיהם (עם כיווץ מסלולים).
- את עץ הדרגות נממש בעזרת עץ הAVL ע"י שמירה בתוך כל צומת את מספר הצמתים בתת העץ שצומת זה מהווה בו שורש (בדומה לאופן שבו אנחנו שומרים את גובה הצומת).

תיאור המערכת:

מבנה הנתונים שלנו יכיל את המידע הבא:

CarDealershipManager
UnionFind<Dealership> dealerships – מבנה נתונים מסוג UnionFind אשר יכיל קבוצות שאיבריהן מטיפוס Dealership המייצג סוכנות רכבים.

כל סוכנות רכב תיוצג לפי הטיפוס:

Dealership
int id – מספר המזהה של סוכנות הרכב.
AvlTree<int, int> car_types – עץ AVL אשר מכיל בכל צומת: • **key-** מספר המזהה של סוג הרכב. • **data-** מספר המכירות של סוג הרכב. ישמש בעיקר בשביל שנוכל לעדכן את עץ המכירות (ולכן אנחנו שומרים את מס' המכירות).
AvlTree<int, SaleKey> sales_tree – עץ דרגות אשר מכיל בכל צומת: • **key-** טיפוס מסוג SalesKey אשר מכיל את מספר המזהה של סוג הרכב וכמות מכירותיו. • **data-** מספר המזהה של סוג הרכב. ישמש בעיקר בשביל שנוכל למצוא את סוג הרכב ה-i הנמכר ביותר ב(log(m.

מימוש הפונקציות:

1. **void* Init():**

נשתמש בבנאי ברירת המחדל הריק של המחלקה CarDealershipManager, אשר משתמשת בבנאי הריק של UnionFind שיוצר מערך ריק באורך 10 ומאתחל את המונה הפנימי ל- 0.

סיבוכיות זמן: O(1) **סיבוכיות מקום:** O(1)

2. **StatusType AddAgency(void* DS):**

נוסיף לUnionFind איבר חדש- נוסיף סוכנות רכב ריקה באינדקס של המונה הפנימי בתוך המערך הדינאמי שבעזרתו מימשנו את UnionFind, סהכ: **O(1) זמן**. אם המערך התמלא, נגדיל אותו ב10 תאים ע"י יצירת מערך חדש שיעתיק את כל האיברים במערך הקודם לחדש, ולכן נקבל **O(n) זמן**, שזה **O(1) משוערך עבור n קריאות AddAgency (לבד).**

סיבוכיות זמן: O(1) משוערך לבד **סיבוכיות מקום:** O(1)

3. **StatusType SellCar(void* DS, int agencyID, int typeID, int k):**

ראשית נבדוק האם הקלט חוקי, ונוודא כי agencyID קטן ממש ממספר הקבוצות בUnionFind, סהכ: **O(1) זמן**.

כעת נמצא את הקבוצה של agencyID בעזרת פעולת find, שזה **O(log*(n)) זמן משוערך** וניגש לסוכנות שמכילה את 2 העצים של הקבוצה. קודם נבדוק האם typeID כבר נמכר בעבר, אם לא, נוסיף אותו לעץ car_types עם 0 מכירות, ולאחר מכן נסיר אותו מעץ הדרגות sales_tree ונוסיף אותו בחזרה עם מספר המכירות החדש, סה"כ: **O(log(m)) זמן**.

הערה- סיבוכיות מציאת הקבוצה של agencyID היא O(log*(n)) זמן משוערך עם הפעולות UniteAgencies ו- GetIthSold, מכיוון שראינו בהרצאה כי פעולת find ו-union בUnionFind עם עצים הפוכים וכיווץ מסלולים נעשים בסיבוכיות זמן O(log*(n)) משוערכת, לכן מכיוון שבפעולה UniteAgencies אנחנו משתמשים בunion ו-find ובפעולת GetIthSold אנחנו משתמשים בfind, נקבל שאכן סיבוכיות זו נכונה.

סיבוכיות זמן: O(log*(n) + log(m)) משוערך עם UniteAgencies ו- GetIthSold
סיבוכיות מקום: O(1)

4. StatusType UniteAgencies(void* DS, int agencyID1, int agencyID2):

נבדוק קודם האם agencyID1 ו- agencyID2 קטנים ממש ממספר הקבוצות בUnionFind.
לאחר מכן, נמצא את הקבוצה של כל אחת מהסוכנויות, סהכ: **O(log*(n)) זמן משוערך** כפי שהסברנו בהערה לעיל, ואז נאחד את העצים של שתי הסוכנויות באופן הבא:

- נמיר כל עץ ל2 מערכים בגודל מספר הצמתים הממוינים לפי המפתח של הצומת, מערך אחד עבור המפתחות ומערך אחר עבור המידע, **O(m1+m2) זמן, O(m1+m2) מקום**.
- נאחד את 4 המערכים שבידינו ל2 מערכים ממוינים של מפתחות ומידע, בזמן שנשמור על הסדר היחסי בין מפתח ומידע של אותו צומת (כלומר באינדקס i במערך המפתחות נמצא המפתח של הצומת עם המידע שנמצא באינדקס i במערך המידע), זה גם יקח **O(m1 + m2) זמן, O(m1 + m2) מקום.**
- נמיר את 2 המערכים הממוינים לעץ אחד עם m1+m2 צמתים כפי שראינו בהרצאה, מכיוון שהמערכים ממוינים זה יקח **O(m1 + m2) זמן, O(m1 + m2) מקום.**

נעשה כך עבור כל עץ, ונשמור עצים אלו בקבוצה הגדולה יותר.

סיבוכיות זמן: O(log*(n) + m1 + m2) משוערך עם SellCar ו- GetIthSold
סיבוכיות מקום: O(m1 + m2)

5. StatusType GetIthSoldType(void* DS, int agencyID, int i, int* res):

ראשית נבדוק את הקלט, ולאחר מכן נמצא את הקבוצה של agencyID בUnionFind, שזה סה"כ: **O(log*(n)) זמן משוערך** כפי שהסברנו בהערה לעיל.

כעת, נבדוק האם i קטן ממש ממספר הרכבים בעץ, ואם כן, נמצא את הצומת עם הדרגה i בעץ המכירות sales_tree, סה"כ: **O(log(m)) זמן משוערך**.

סיבוכיות זמן: O(log*(n) + log(m)) משוערך עם UniteAgencies ו- SellCar
סיבוכיות מקום: O(1)

6. void Quit(void DS):**

נקרא להורס של CarDealershipManager שיהרוס את כל האיברים בUnionFind, כאשר כל עץ בDealership ניתן להרוס ב**O(k) זמן, O(k) מקום**, כאשר k הוא מספר הצמתים בעץ,
ולכן עבור מספר כולל של m סוגי רכב נקבל **O(m) זמן, O(m) מקום**, ומכיוון שUnionFind מכיל מערך באורך n (מספר הסוכנויות) נצטרך להרוס אותו ב**O(n) זמן**.

סיבוכיות זמן: O(n + m) **סיבוכיות מקום:** O(m)