234218 – מבנה נתונים

אביב תשפ"א 2020-2021

תיעוד של רטוב #1

שם סטודנטים: כריסטין שאהין, גורג סארגי

מספרי תעודות זהות: 207660150 , 211910708

מועד הגשה: 25/05/2021

מספר הגשה: יבש של רטוב #1

במבנה הנתונים שלנו, השתמשנו ב- 4 עצי חיפוש, ושני מצביעים.

עבור כל עץ חיפוש, קיים מבני ייחודי עבורו כך שיענה על דרישת החיפוש באופן שמתאים לו כדי לסדר את האיברים בכל עץ.

נציג את המבנים ואת תוכנם:

1. המבנה הראשון יהיה מבנה ה- CarModel, שהוא מבנה שמכיל 5 שדות של int: sales, complaints, id, type, grade, והוא מבנה הייחודי לדגם מכונית.
המבנה עוקב אחרי כל השדות כפי שתואר בתרגיל כך ש:
   - Type הוא המספר המזהה של סוג הרכב.
   - Id הוא המספר המזהה של דגם הרכב.
   - Sales עוקב על כמות הפעמים אשר מוכרים את דגם המכונית.
   - Complaints העוקב על מספר התלונות עבור כל רכב,
   - Grade העוקב אחרי ציון הרכב.

   המבנה יהיה מסודר בעץ לפי הציון הטוב ביותר, אם הציונים שווים, אז נסדר לפי המספר המזהה של סוג הרכב, כך שהרכב בכל המספר המזהה של סוג הרכב הקטן ביותר תחושה כ"גדולה יותר", ואם גם זה שווה אז נבדוק לפי המספר המזהה של דגם המכונית כך שהרכב בעל המספר המזהה של דגם הרכב הקטן ביותר תחושה כ"גדולה יותר".

2. המבנה השני יהיה מבנה ה- CarType, שהוא מבנה ייחודי עבור כל סוג רכב.
המבנה מכיל את השדות הבאות:
3 שדות int: id, max_sales, num_of_models.
שדה של *CarModel: best_seller.
ושדה של **CarModel: models.
   - Id הוא המספר המזהה של סוג הרכב.
   - Max_sales הוא המספר המקסמלי של ה- sales עבור הדגמים.
   - Num_of_models הוא מספר הדגמים של הרכב הזה.
   - Best_seller שהוא הדגם הנמכר ביותר עבור הדגמים של סוג הרכב הזה.
   - Models שהוא מערך של מצביעים על CarModel, כך שהתא ה- i במערך יהיה מצביע על הדגם בעל המספר המזהה i.

   העץ של המבנה יהיה ממוין לפי המספרים המזהים של מספרי הרכב.

3. המבנה השלישי יהיה מבנה ה- TypeNode, שיעקוב אחרי הדגמים שלא נמכרו עדיין והוא יכיל את השדות הבאים:
   - שדה int type_id המכיל את המספר המזהה של סוג הרכב.
   - שדה Tree<CarModel>* models שיהיה עץ שמכיל את הדגמים שלא נמכרו, מאותו type_id כך שיהיו ממויינים לפי המיון של העץ של המבנה הראשון.

- שדה *<Tree<CarModel smallest_model שהוא מצביע לצומת בעץ models בעל המספר המזהה הקטן ביותר.

הערה חשובה: כל תא בעץ של המבנה הזה מכיל עוד עץ בתוכו.

העץ הזה יהיה ממוין לפי המספרים המזהים של סוגי הרכב כלומר לפי ה- type_id.

4. המבנה האחרון יהיה המבנה SalesNode שיעקוב אחרי המכירות של כל הדגמים במערכת.
המבנה יכיל רק שדה אחד שיהיה *CarModel model, ויעקוב אחרי המכירות של כל דגם קיים.

המבנה יהיה מסודר בעץ לפי ה- sales הגבוהה ביותר, אם המכירות שווים, אז נסדר לפי המספר המזהה של סוג הרכב, כך שהרכב בכל המספר המזהה של סוג הרכב הקטן ביותר תחושה כ"גדולה יותר", ואם גם זה שווה אז נבדוק לפי המספר המזהה של דגם המכונית כך שהרכב בעל המספר המזהה של דגם הרכב הקטן ביותר תחושה כ"גדולה יותר".

5. המבנה הראשי שיש לנו עם המעטפת הינו CarDealershipManager. זהו המבנה הכללי שכולל כל העבודה והפונקציות.
המבנה מכיל מספר משתנים:

- *CarModel bestModel: מצביע כללי לדגם הטוב ביותר מבחינת מכירות.
- Int total_models: מספר הדגמים בכל המערכת
- *<Tree<CarType types: עץ המכיל כל הסוגים שמאחסנת המערכת
- *<Tree<CarModel sold_models: עץ של כל הדגמים שנמכרו.
- *<Tree<TypeNode non_sold_models: עץ של כל הדגמים שעדיין לא נמכרו
- *<Tree<SalesNode car_sales: עץ שמאחסן הדגמים לפי מספר המכירות (שמאל = מכירות גבוהות, ימין = מכירות נמוכות).

<u>Init():</u>

מאתחלת מבני נתונים ריק:

מאתחלת את כל 4 העצים, כך שיואתחלו כעצים ריקים. בנוסף נאתחל את המצבעים ל-nullptr ואת total_models לאפס.

<u>סיבוכיות:</u> הזמן של פעולות האתחול דורשת $O(1)$.

AddCarType:

הפעולה הזו מוסיפה סוג של רכב חדש למערכת.

תחילה, בודקים תקינות הנתונים, ומאתחלים משתנים מסוג CarType, TypeNode, SalesNode ובודקים אם קיים הסוג כבר במערכת ע"י חיפוש בינארי. אם קיים, משחררים ומחזירים שגיאה.

אחר כך, אני יודעים כי לא קיים סוג כזה, ולכן מוסיפים אותו לעץ הסוגים, ואז מוסיפים את משתנה מסוג TypeNode לעץ המתאים, וכנ"ל למשתנים מסוג SalesNode.

בכל שלב בפונקציה, אם מתרחשת שגיאת זיכרון אז משחררים את מה שהקצאני בפונקציה ומחזירים השגיאה המתאימה.

סיבוכיות: השתמשנו בחיפוש בינארי על עץ בן n איברים, והוספת m איברים ממוינים לעץ ממוין, ולכן קיבלנו: O(log(n) + m), כאשר: n – מספר הסוגים במערכת, m – מספר הדגמים החדשים

RemoveCarType:

בפעולה הזו נצטרך להסיר את סוג הרכב הזה מכל העצים הנמצאים.

קודם נצטרך להסיר אותה מהעץ של types. בעץ הזה יש לנו n תאים בעץ שהוא מספר סוגי הרכב הכלליים. נחפש את התא בעץ של ה- typeID המתאים, שדורש $O(\log(n))$ פעולות. הסרת המערך models מהתא דורש $O(m)$ פעולות, ושאר ההסרה דורשת $O(1)$.

כעת צריך להסיר אותה מעץ ה- sold_models ומעץ ה- car_sales שדורשת $O(\log(M))$ פעולות חיפוש לתא המתאים כי יש M איברים בכל עץ והסרתו שדורשת $O(1)$ פעולות, עבור כל דגם. מכיוון שיש לנו m דגמים אזי בסה"כ לפעולה זו נצטרך ל- $O(mlog(M))$ פעולות.

כעת צריך להסיר אותו מהעץ האחרון, שיש בו n איברים, ולכן מציאת התא המתאים דורשת $O(\log(n))$ פעולות והסרת התא עם העץ שבתוכו, שיש בו לכל היותר m תאים עבור ה- m דגמים, בסה"כ דושרת $O(\log(n) + m)$.

רק נותר עידכון המצביעים הנמצאים במבנה הכללי שדורשת $O(\log(n) + \log(M))$ כי עלינו לקחת את האיברים שהכי שמליים בעצים: *non_sold_models* ו- *car_sales*.

סיבוכיות: *לכן בסה"כ נקבל :*$O(\log(n) + mlog(M))$.

SellCar:

הפונקציה זו מטפלת במכירת דגם רכב מסויים כנתון.

ראשית, אנו בודקים אם הנתונים תקינים, ואם לא, מחזירים שגיאה מתאימה לפי פרוט הפונקציה המבוקשת. אחר כך, אנחנו עושים חיפוש בונארי על עץ הסוגים כדי למצוא הסוג המבוקש. אם לא מוצאים הסוג המבוקש, מחזירים השגיאה המתאימה ומשחררים כל המשתנים שהקצאנו עד כו. אחר כן, אנחנו בודקים אם קיים המזהה הנתון של הדגם לפי מספר הדגמים בסוג. כנ"ל, אם לא מתקיים, מחזירים שגיאה מתאימה ומשחררים כל המשתנים שהקצאנו.

אם הדגם עדיין לא נמכר, אנחנו מסירים את הנתונים שלו מעץ non_sold_models, מעדכינים נתונים שלו ומוסיפים צומת מתאים לעץ sold_models ומעדכינים צומת המתאים בעץ car_sales. אחר כך מעדכנים משתנים גלובליים המתאימים לפי מידת צורך.

אם הדגם כבר נמכר, אז מעדכינים הנתונים שלו, מעדכנים הצומת המתאים בעץ sold_models וגם בעץ car_sales, ואחר כך מעדכינים המשתנים הגלובליים המתאימים לפי מידת הצורך.

סיבוכיות: השתמשנו בחיפוש בינארי והוספה/הסרה על עצים בני n ו-M איברים במקרה גרוע, ולכן מקבלים: O(log(n) + log(M)) כאשר: n – מספר סוגים במערכת, M – מספר הדגמים במערכת.

GetBestSellerModelByType:

תחילה, אנחנו בודקים תקינות המשתנים ומחזירים השגיאה המתאימה אם צריך.

במקרה שנתון כי typeID = 0, אז אנחנו משתמשים במשתנה bestModel ומחזירים בהתאם.

אחרת, אנחנו מחפשים על הסוג המבוקש בעץ הסוגים, ומשתמשים במשתנה פנימי bestseller ומחזירים בהתאם.

סיבוכיות: כאשר typeID = 0, מקבלים O(1). אחרת, עושים חיפוש בינארי על עץ בן n איברים, ולכן מקבלים: O(log(n)), כאשר n הוא מספר הסוגים במערכת

GetWorstModels:

הפונקציה זו מטפלת בהחזרת מספר הדגמים המבוקשים עם הציון הנמוך ביותר בסדר עולה. ראשית, בודקים אם הנתונים תקינים. אם לא, מחזירים שגיאה.

בפונקציה זו מימשנו אלגוריתם מבוסס על inorder עם שינויים, כך שאנחנו מתחילים מסוף העץ ועולים למעלה בתוך קריאה עד שמגיעים למספר המבוקש של דגמים או עד שהעץ הנוכחי נגמר.

השתמשנו במצביעים על האיברים הקטנים ביותר בעצי sold_models, non_sold_models שהגדרנו במבנה כללי. אנחנו לוקחים איברים מתוך עץ sold_models לפי האלגוריתם שלנו עד שאנחנו מגיעים להפיכת סימן, ואז משלימים מעץ non_sold_models לפי האלגוריתם, עד שמגיעים למספר הדגמים המבוקש או עד שעברנו על כלל הדגמים. אם עדיין נותר בקשה לעוד דגמים, אנחנו ממשיכים עם האלגוריתם שלנו בעץ sold_models מנקודה שעצרנו בה.

סיבוכיות: השתמשנו בסיור inorder על m איברים עם שינויים קטנים, ולכן מקבלים O(m) כאשר m הוא מספר הדגמים המבוקשים.

MakeComplaint:

בפונקציה הזו נדרש לעדכן את ציון הרכב עם המספר המזהה המתאים ולבצע תלונה.

לכן עלינו רק לעדכן את נתונים בשני העצים הבאים:

קודם עלינו לעדכן את הנתונים בעץ types, לכן נבצע חיפוש בינרי על העץ לפי המספר מזהה של סוג הרכב, שדורש $O(\log(n))$ פעולות, אחר כך, עלינו רק לגשת לאינדקס ה- modelID במערך ולעדכן את נתונים, לכן בסה"כ הסיבוכיות של הפעולה הזו היא $O(\log(n))$.

כעת נחפש את התא המתאים בעץ sold_models שיש בו M איברים ע"י ביצוע חיפוש בינרי עם המספר מזהה המתאים, אבל במקרה הזה לא נעדכן את הפרטים כי זה יפגע במיון העץ, לכן נסיר את התא הזה מהעץ, ואז נכניס תא חדש לעץ עם הנתונים המעודכנים. סיבוכיות הפעולה הזו היא: $O(\log(M))$.

רק נותר לעדכן את המצביע smallest_sold_model שזו פעולה שדורשת $O(\log(M))$, מכיוון שאנחנו מצביעים על האיבר הכי שמאלי בעץ.

סיבוכיות: בסה"כ יוצא: $O(\log(n) + \log(M))$.

Quit:

בפונקציה הזו נצטרך לשחרר את כל המבנים הנמצאים, לכן על מנת לשחרר את כל העצים, צריכים לעבור על כל האיברים.

נשחרר עצים בני $m + n$ איברים ולכן:

סיבוכיות: $O(n + m)$.

נזכור כי בכל עץ מוקצה מקום לכל היותר כמספר הדגמים והסוגים ולכן סיבוכיות המקום היא: $O(n + m)$.