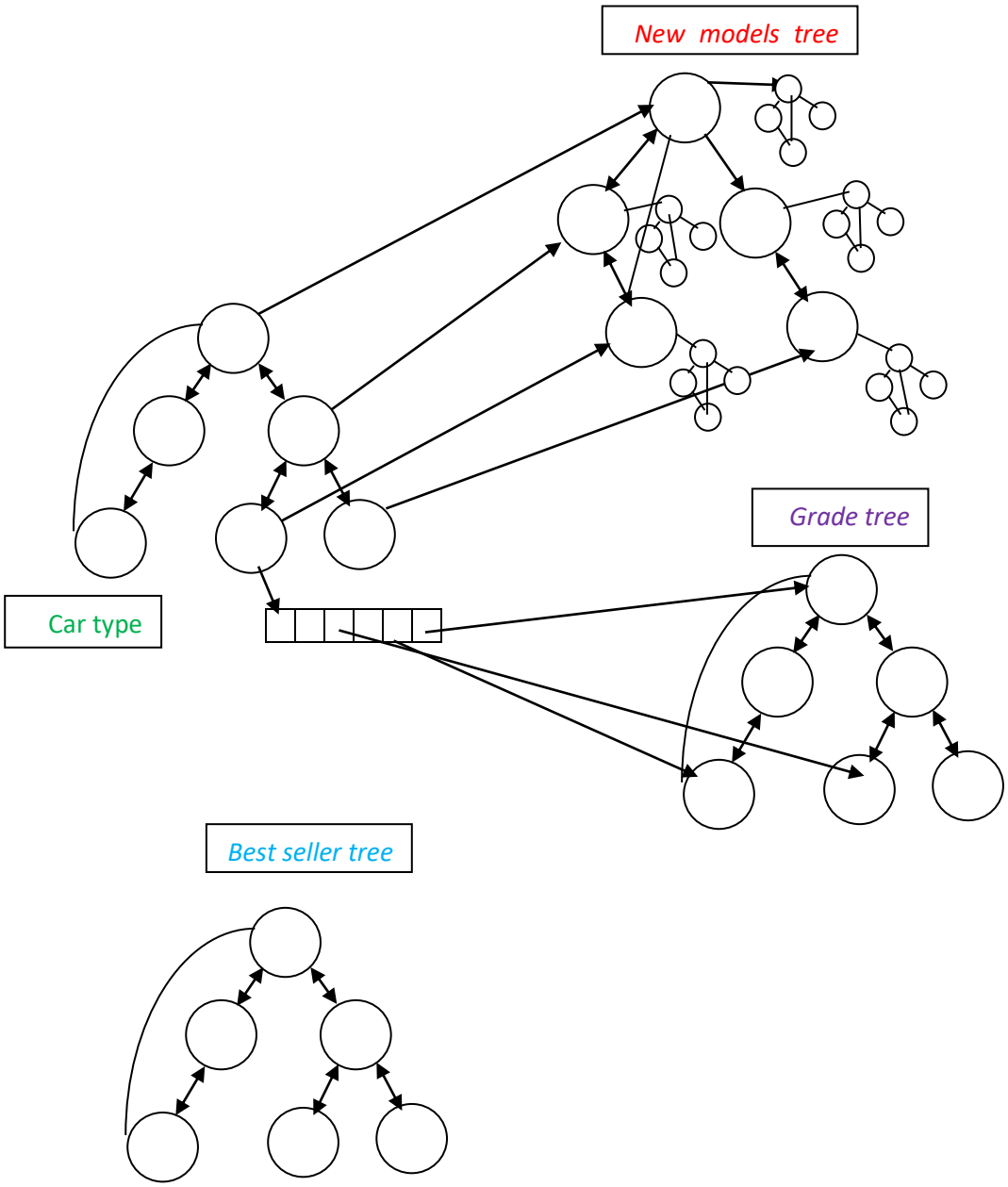

מבני הנתונים:

עץ AVL לסוגי הרכבים Car_types הממוין לפי typeID שבו כל איבר מכיל:

- typeID- מזהה סוג הרכב.
- Models_array- מערך מצביעים כאשר גודלו כמספר הדגמים וכל מצביע מצביע לצמת בעץ Grades כאשר כל צמת בעץ Grades מייצגת ציון של דגם שנמכר לפחות פעם אחת, אחרת מצביע לnullptr
- New_model מצביע לצמת בעץ New_models אשר מכיל צמתים שכל צמת מכיל תת עץ של דגמים שלא נמכרו בעבר.
- Best_seller ערך שנתחזק בפעולות השונות וייצג את הדגם הנמכר ביותר מאותו סוג
- שדות שייסעו בתנועה על העץ: אב בן ימני, בן שמאלי ומצביע לעלה השמאלי ביותר שיתוחזק בכל פעולת הכנסה והסרה.
- Num_of_models מספר הדגמים מאותו הסוג

עץ AVL ציונים Grades המשמש כמילון כאשר המפתחות הם ציונים של הדגמים השונים אשר נמכרו לפחות פעם אחת בעבר וכל צמת בו מכיל:

- Grade – ציון הדגם.
- typeID- מזהה סוג הרכב.
- Model- מס מודל הרכב
- Num_of_sells מס הפעמים שנמכר אותו הדגם
- שדות שייסעו בתנועה על העץ: אב בן ימני, בן שמאלי ומצביע לעלה השמאלי ביותר.
*העץ ימוין כאשר חשיבות הערך שלפיו ממינים היא בסדר יורד של השדות הנ"ל.

עץ AVL דגמים חדשים New_models (זהו בעצם עץ של עצים) הממוין לפי typeID שבו כל איבר

מכיל:

- Model- מס מודל הרכב.
- typeID- מזהה סוג הרכב.
- תת עץ של דגמים חדשים המאותחל ב O(n) לפי אלגוריתם שכל צמת בו תייצג דגם ותכיל המזהה הסוג שלו.
- שדות שייסעו בתנועה על העץ: אב בן ימני, בן שמאלי ומצביע לעלה השמאלי ביותר.

עץ AVL של מכירות Bestseller הממוין לפי מספר מכירות של אותו הדגם שמכיל דגמים שנמכרו פעם אחת לפחות שבו כל איבר

מכיל:

- Num_of_sells מספר מכירות של אותו הדגם
- typeID- מזהה סוג הרכב.
- Model- מס מודל הרכב.
-
- שדות שייסעו בתנועה על העץ: אב בן ימני, בן שמאלי ומצביע לעלה השמאלי ביותר.

העץ ימוין כך שהעלה השמאלי ביותר יכיל את הדגם הכי נמכר המערכת כלומר צמת שמאלי יותר יכיל מספר מכירות גבוה יותר אמ מספר מכירות שווה ומזהה סוג קטן יותר או מספר מכירות ומזהה סוג שווים ומס מודל קטן יותר.

בכל העצים נתחזק מצביע לעלה השמאלי ביותר שיאותחל לשורש ויעודכן לאחר כל הוספה והסרה פשוט נסייר שמאלה בעץ ככל האפשר ובכך נמצא לאן להצביע מה שלוקח $O(\log n)$ ועל כן סיבוכיות פעולות ההוספה וההסרה נשארת זהה.

Models_counter – מייצג מספר הדגמים הכולל במערכת.

:void * Init()

נאתחל אובייקט DM (deals manager)

נאתחל את 4 עצי הAVL המתוארים במבנה הנתונים ריקים ב O(1).

נאתחל Models_counter = 0 בזמן O(1).

סה"כ ביצוע של פעולות בזמן O(1).

StatusType AddCarType (void *DS, int typeID, int numOfModels)

- נוודא תקינות קלט ונחפש אם קיים כבר אותו הסוג רכב ונזרוק שגיאה בהתאם כמידת הצורך. חיפוש $\log n$
- נוסיף לעץ New_models צמת חדש נאתחל לו תת עץ לפי האלגוריתם* שגודלו כמספר הדגמים $O(\log n)$ להוספה ו $O(m)$ לאלגוריתם.
- נוסיף לעץ Car_types צמת חדש עם typeID , נאתחל מערך מצביעים בגודל numOfModels ל-nullptr נאתחל את הצמת החדש בעץ הסוגים להצביע לצמת החדשה שהוספנו לעץ New_models $o(\log n)$ להוספה ו $O(m)$ לאתחול המערך
- נוסיף למספר הדגמים במערכת את הדגמים החדשים (נעדכן את הcounter) O(1)
- פירוט האלגוריתם לאתחול תת העץ כאשר ידוע מספר המודלים :
- נחשב את גובה העץ לפי מספר הצמתים נתון לנו נאתחל שדות begin=0 end=numofmodels-1
- החל מהשורש פעל לפי האלגוריתם הבא
- אם begin>end חזור
- חשב אמצע (begin+end) /2

- עדכן שדה האב וגובה (בהתאם לעומק הרקורסיה)
- הפעל את האלגוריתם על בן שמאל אם קיים עם הפרמטרים begin=begin end=mid-1
- הפעל את האלגוריתם על בן ימין אם קיים עם הפרמטרים begin=mid+1 end=end
-

בסוף גם נעדכן מצביע לבן השמאלי היותר בעלות של log(n)

סהכ עברנו על כל צמת מספר קבוע של פעמים לכן הסיבוכיות היא O(m)

סיבוכיות זמן:

$$O(\log n) + O(m) + O(\log n) + O(m) + O(1) = O(\log n + m)$$

StatusType RemoveCarType(void *DS, int typeID)

- נוודא תקינות קלט בהתאם כמידת הצורך $O(1)$

-נחפש את סוג המכוניות בעץ Car_types אם היא לא קיימת נזרוק שגיאה. $O(\log n)$.
-אם כן נחסיר מ Models_counters את מספר הדגמים של הסוג רכב שנמחק.
- נגש למערך Models נעבור על איבריו עם קיימים ונמחק אותם מהעץ Grades ומהעץ BestSeller כל מחיקה $O(\log M)$ ולכל היותר m מחיקות סהכ $O(m\log M)$.
-נגש לצמת בעץ New_models בעזרת המצביע New_model נמחק את תת העץ ואת הצמת. מחיקת תת העץ O(m) (מחיקת עץ מוסברת בפונקצייה Quit) מחיקת הצמת O(logn).
נמחק את הצמת בעץ Car_type O(logn) .

סיבוכיות זמן:

$$O(3\log n) + O(mlogM) + O(1) = O(\log n + mlogM)$$

StatusType SellCar(void *DS, int typeID, int modelID)

- נוודא תקינות קלט והאם סוג הרכב קיים ונזרוק שגיאה בהתאם כמידת הצורך .

-חפש את סוג בעץ Car_types $O(\log n)$.

-בדוק האם Models[modelID]!=Null (כלומר נמכר בעבר) אם כן:

-גש ל-Models[modelID] עדכן את הציון בצמת בהתאם למכירה אחת.

-חפש את המודל בעץ המכירות ועדכן את הצמת המתאים. $O(\log M)$

-אם Models[modelID]==Null: (כלומר לא נמכר בעבר)

-גש לצמת בעץ New_models בעזרת המצביע New_model

-מחק את הדגם מתת העץ $O(\log M)$.

-אם תת העץ ריק כעת נמחק גם את הצמת ונעדכן את המצביע New_model לnullptr
$O(\log n)$.

-צור צמת חדש עבור הדגם הנ"ל כאשר הציון הוא לאחר מכירה אחת, ותוסיף אותו לעץ Grades.
$O(\log M)$

-צור צמת חדש עבור הדגם הנ"ל עם מכירה אחת, ותוסיף אותו לעץ המכירות $O(\log M)$

נעדכן את השדה של best_seller במידת הצורך $O(1)$.

סיבוכיות זמן:

$$O(\log n) + O(4logM) + O(1) = O(\log n + logM)$$

.

StatusType MakeComplaint(void *DS, int typeID, int modelID, int t)

-חפש את הקורס בעץ Car_types

-גש ל-Models[modelID] עדכן את הציון בהתאם לתלונה.

-בצע גלגולים לעץ Grades במידת הצורך

סיבוכיות זמן:

StatusType GetBestSellerModelByType(void *DS, int typeID, int *modelID)

אם typeID=0 החזר את ערך Best_seller של השורש של העץ Car_types
אחרת מצא את סוג הרכב בעץ Car_types והחזר את הערך של Best_seller.

StatusType GetWorstModels(void *DS, int numOfModels , int *types, int *models)

נבדוק האם numOfModels קטן או שווה לModels_counter אם לא נזרוק שגיאה.
נחלק את ההדפסה ל3 שלבים : הדפסת ציונים שליליים ציונים שווים לאפס וציונים חיוביים

ציונים שליליים
נגש לעץ Grades נעזר במצביע לעלה השמאלי ביותר ונתחיל להדפיס לפי סיור inorder כל עוד הציון שלילי (אם בכלל)אחר נמשיך לשלב הבא
אפסים
נעבוד כעת על שתי העצים ביחד עץ Grades מאיפה שנפסק השלב הקודם ועץ New_models כאשר אנחנו ממשיכים בסדר inorder בשניהם העדיפות ביניהם תינתן לפי typeID הקטן בניהם ובמקרה של שיוון העדיפות בניהם תינתן לפי מזהה סוג הרכב הקטן מבניהם. (בדומה לפונקציה הממזגת באלגוריתם merge sort)

כל זה אם קיימים צמתים בעלי ציון 0 או כאשר אנחנו במצב שקיימים אפסים רק בעץ אחד אז נדפיס אותם ב Inorder רגיל.

ציונים חיוביים :
נמשיך להדפיס ב inorder על העץ Grades החל מאיפה שהפסקנו בשלב הקודם.

בכל שלב נפסיק את ההדפסה אם הדפסנו כבר numOfModels הדפסות.

void Quit(void **DS)

נבצע מחיקה לכל העצים ואם אם לצמת יש תת עץ או מערך מצביעים נמחק אותם גם המחיקה תתבצע בעזרת פונקציה שונה לא בעזרת פונקציית ההסרה של עץ בכדי לעמוד בסיבוכיות הזמן.
נמחק את המבנה עצמו.
כלומר :
כל עוד עלה שמאלי קיים – delete(left)
כל עוד עלה ימני קיים – delete(right)
מחק צמת זה.
חזור
(בדומה לסיור postorder)
בהתאם לגודל המבנה ותת המבנים בתוכו שמפורט בחישוב המקום למטה
מחיקה תעלה:
סה"כ עברנו על כל צמת בכל עץ ובכל תא המערך מספר קבוע של פעמים מחיקה תעלה מבחינת זמן :

$$O(2m) + O(2n) + O(2m) = O(m + n)$$

סהכ: עץ המכירות והציונים בגודל m כמספר הדגמים לכל היותר כי כל ציון או רשומה של מכירות מייצגת דגם אחד.
עץ הסוגים והדגמים החדשים בגודל n כל אחד לכל היותר כל כל צמת מייצגת סוג רכב אחד כל תתי העצים וכל המערכים במערכת בגודל כולל של 2m כי כל תא במערך או צמת בתת העץ מייצגים דגם אחד

סהכ סיבוכיות מקום:

$$O(2m) + O(2n) + O(2m) = O(m + n)$$