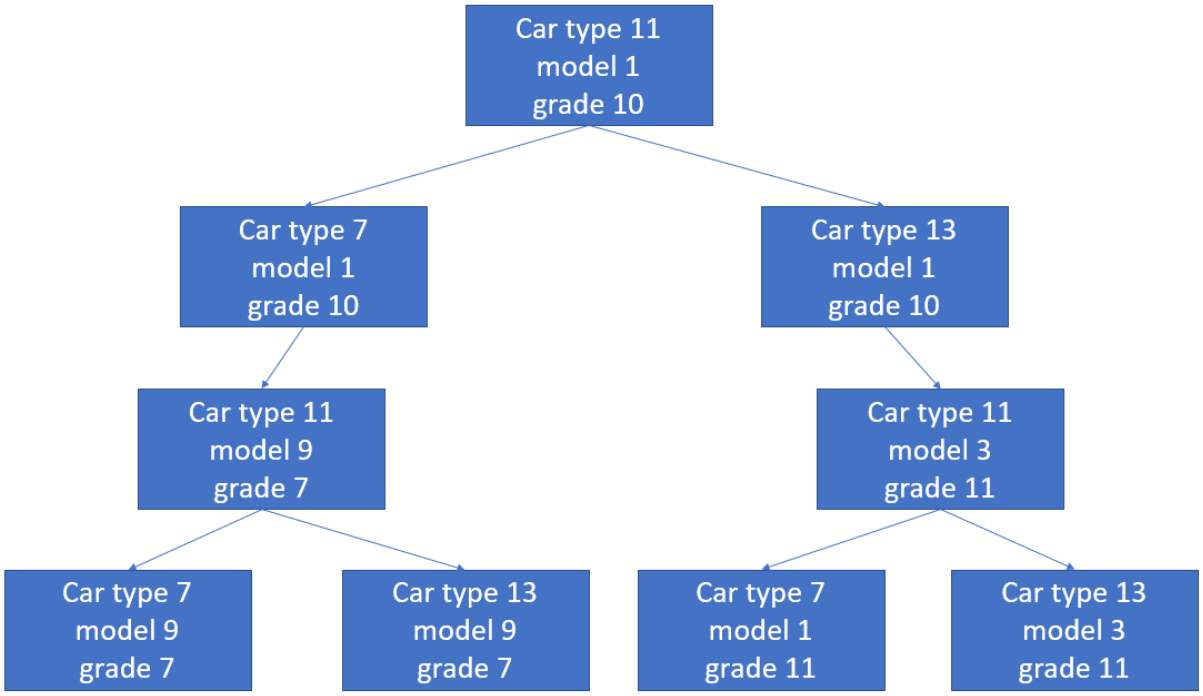
מבנה:

עצים ראשיים:

עץ א': עץ avl דו כיווני ראשי של ציונים, בו יופיעו כל הדגמים של כל סוגי הרכבים(נבחין כי רק דגמים בעלי ציון ייכנסו לעץ כל דגם שטרם נמכר יהיה בעל ציון 0). החוקיות תעבוד באופן הבא:
כל צומת תבנה כך שמימינו יהיו דגמים עם ציון גבוה יותר, ומשמאלו עם ציון נמוך יותר.
במקרה של ציון זהה דגמים עם מס' סוג רכב קטן יותר יהיו משמאל,
ובמקרה של מס' סוג רכב זהה דגמים בעלי מס' דגם קטן יותר יהיו משמאל.
בנוסף, נשמור מצביע לצומת הכי ימני(בעל הציון הגדול ביותר) ולצומת הכי שמאלי (בעל הציון הנמוך ביותר).

עץ ב': עץ avl דו כיווני ראשי של מכירות, בו יופיעו כל הדגמים של כל סוגי הרכבים(נבחין כי רק דגמים שנמכרו ייכנסו לעץ כל דגם שטרם נמכר יהיה בעל ציון 0). החוקיות תעבוד באותו אופן כמו בעץ ציונים הראשי.
בנוסף, נשמור מצביע לצומת הכי ימני(האחד הכי נמכר)



תחזוקת מצביעי min/max:

הכנסה:

אם הכנסנו בן ימני/שמאלי לצמתים min/max אזי יש לעדכן את המצביע$O(1)$

הוצאה:

נבחין כי ישנם שני מצבים:

אם יש לצומת בן יחיד(עבור min ייתכן רק בן ימני ועבור max רק בן שמאלי) ואז נעביר את המצביע min/max לבן זה,ואת הבעלות עליו לאבא של הצומת min/max לפני המחיקה$O(1)$

אם אין לצומת בנים כלל (עלה) אזי נעביר את המצביע min/max לאבא.$O(1)$

גלגולים:

בגלגולים לא משנים את היחס בין האיברים! רק את צורת הופעתם בעץ כלומר את גובהו,ולכן לא צריך לעדכן את המצביע לmin/max

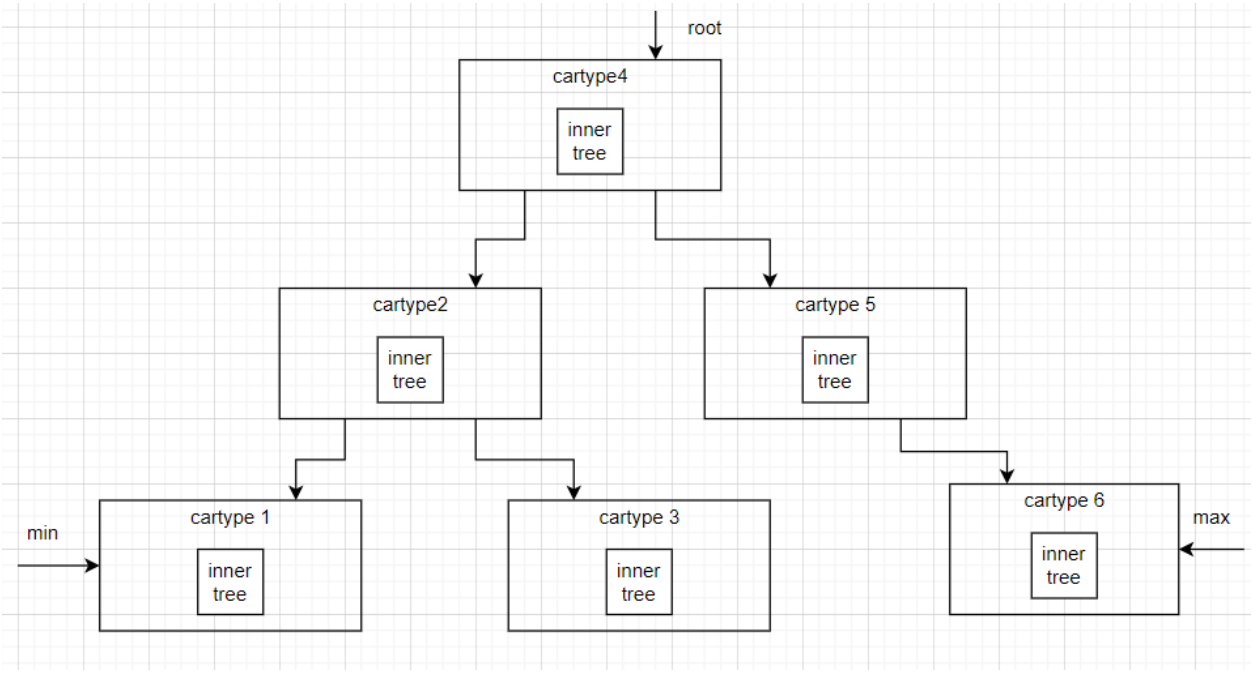
שאר הפעולות ממושות על ידי פעולות אלה ולכן תחזוקת המצביעים בהתאם.

עצי cartype עם עצים פנימיים:

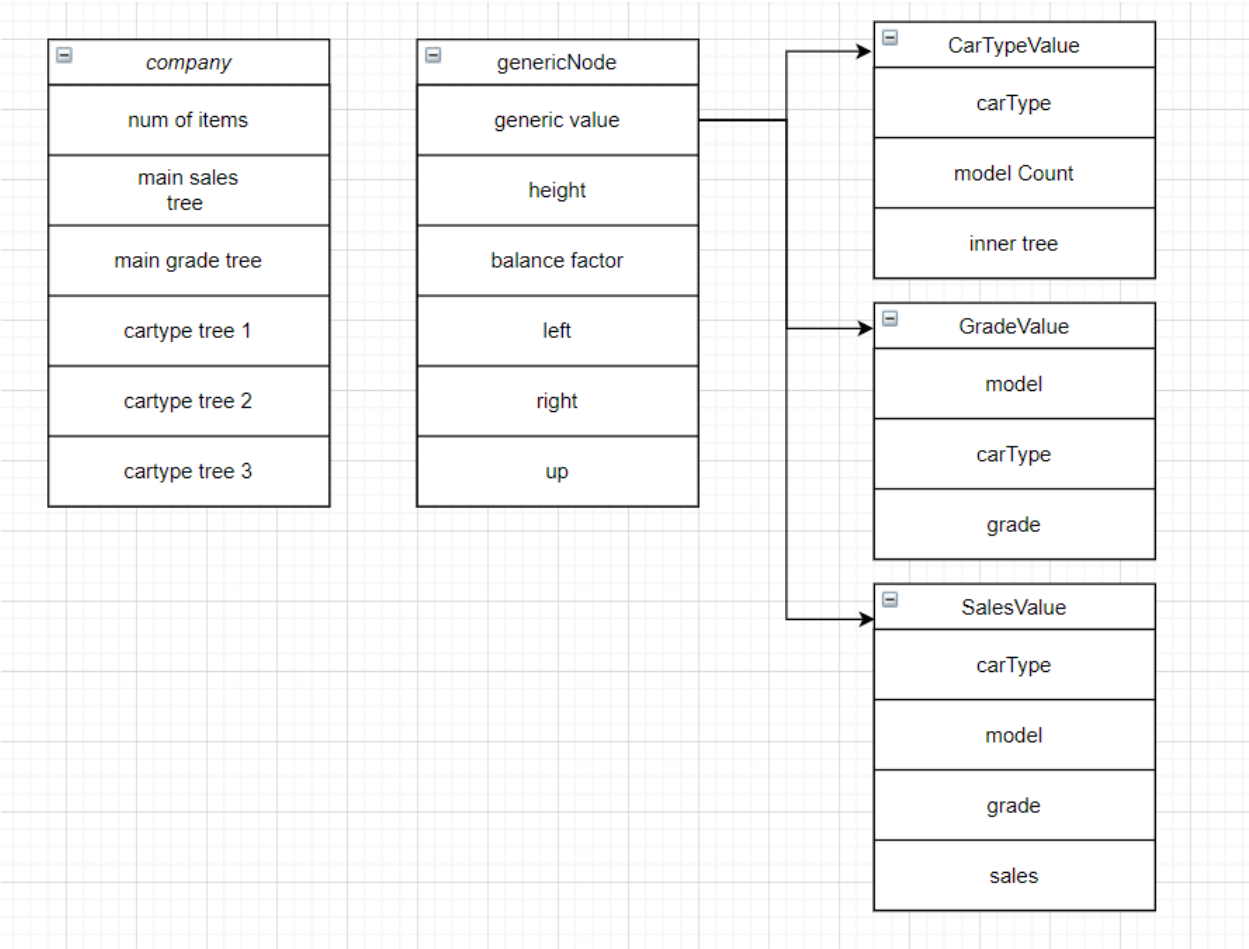
עץ ג': עץ avl דו כיווני של סוגי רכבים כאשר כל צומת תכיל עץ מכירות נוסף של כל הדגמים של סוג רכב זה.
העץ הפנימי יעבוד בחוקיות דומה לעץ א'.

עץ ד': עץ avl דו כיווני של סוגי רכבים כאשר כל צומת תכיל עץ נוסף של דגמים אשר טרם נמכרו מסוג רכב זה.(ציונם יהיה 0)
העץ הפנימי יעבוד בחוקיות דומה לעץ א'.

עץ ה': עץ avl דו כיווני של סוגי רכבים כאשר כל צומת תכיל עץ פנימי נוסף של כל הדגמים של סוג רכב זה.
העץ הפנימי יעבוד בחוקיות שונה מהאחרים - למרות שכל דגם מכיל בתוכו את המידע על ציון\מכירות, המיון יתבצע רק לפי מספר דגם. בצורה זו נוכל בקלות לחפש מידע על דגם ספציפי כאשר אנחנו לא יודעים מה הציון שלו.



תיאור סכמטי:



ניתוח סיבוכיות:

void * Init()
$O(1)$

אין צורך לאתחל את העצים

StatusType AddCarType (void *DS, int typeID, int numOfModels)
$O(log(n) + m)$

הכנסה של סוג רכב לעץ סוגי הרכב $O(log(n))$
לא נכניס לתוך עצי הציונים! רק במכירה נכניס $O(1)$
נבנה את תת העץ של דגמים שלא נמכרו ואת תת העץ הכללי לפי השיטה לבניית עץ כמעט שלם מההרצאה.
$O(m)$.
סך הכל:

$O(log(n) + m)$

StatusType RemoveCarType(void *DS, int typeID)
$O(log(n) + mlog(M))$

נמצא את סוג הרכב בעץ סוגי הרכב ובעץ הפנימי הכללי של הדגמים.
נעזר בנתונים ששם כדי להסיר את המודלים בעצי המודלים הגדולים.
נוציא את הרכב מעץ סוגי הרכב $O(log(n))$
נמחק את תת עץ דגמים שלא נמכרו על ידי מחיקת השורש של העץ m פעמים $O(mlog(m))$
נמחק את תת עץ המכירות $O(mlog(m))$
נמחק את תת עץ הדגמים הכללי $O(mlog(m))$

נוציא את המודלים שנמכרו מעץ הציונים הראשי כל הוצאה תהיה $O(log(M))$ לכן עבור m הוצאות נקבל
$O(mlog(M))$
נבחין כי: $O(mlog(m)) < O(mlog(M))$
לכן סך הכל:

$O(log(n) + mlog(M))$

StatusType SellCar(void *DS, int typeID, int modelID)
$O(log(n) + log(M))$

נמצא את סוג הרכב $O(log(n))$ ונחפש את הדגם בעץ הדגמים הכללי.
בעזרת המידע המלא נחפש בעץ הדגמים שטרם נמכרו אם המודל קיים נוסיף לציון שלו 10.
אם המודל לא קיים נוסיף צומת חדש ונמחק את הדגם מהעץ $O(log(m))$כמובן מתקיים:
$O(log(m)) < O(log(M))$
את סדר ההוספה נבצע כך שאם יש כבר דגם בעל אותו ציון, מספר הדגם הנמוך יהיה משמאל למספר הדגם הגבוה.
כעת נחפש את הדגם בעץ הציונים הראשי. אם נמצא נוסיף לציון שלו 10 ואם לא נכניס צומת חדש $O(log(M))$ באותו אופן שהוספנו לעץ הציונים הקטן.
כעת נחפש את הדגם בעץ המכירות הראשי. אם נמצא נוסיף לציון שלו 10 ואם לא נכניס צומת חדש $O(log(M))$ באותו אופן שהוספנו לעץ הקטן.
לכן סך הכל:

$O(log(n) + log(M))$

StatusType MakeComplaint(void *DS, int typeID, int modelID, int t)
$O(log(n) + log(M))$

בעץ סוגי הרכבים:
נמצא את סוג הרכב $O(log(n))$ ובתת העץ הכללי של דגמים נקבל את המידע המלא $O(log(m))$,נמצא את הדגם בתת העץ של דגמים שנמכרו$O(log(m))$,וכן בעץ הציונים הראשי $O(log(M))$,נוריד את הציון לפי החישוב $\left[\frac{100}{t}\right]$ (חשוב להדגיש כי נמחק את העבר ונכניס אותו מחדש כך שיישמר הסדר בעץ.)
נזכיר כי נשמור על חוקיות שבה דגמים עם אותו ציון יסודרו כך שמספר הדגם הנמוך יותר יהיה בצד שמאל.

לכן סך הכל:

$O(log(n) + log(M))$

StatusType GetBestSellerModelByType(void *DS, int typeID, int *modelID)
$typeid = 0: O(1)$
$typeid \neq 0: O(log(n))$
אם $typeid = 0$ נתחיל מהצומת הכי ימני של עץ המכירות(יש לנו מצביע אליו ולכן $O(1)$) נוודא כי אין צומת בעל אותו כמות מכירות אך עם דגם קטן יותר. נמצא את הצומת הפוטנציאלי הבא על ידי האלגוריתם הבא:
אם יש לו בן שמאלי נרד אליו וניקח עד הסוף ימינה
אם אין בן שמאלי נעלה במעלה העץ כל עוד היינו בן שמאלי של הצומת שעולים אליו נמשיך לעלות אם עלינו מבן ימני נפסיק ונחזיר אותו.

אם $typeid \neq 0$ נחפש את סוג הרכב בעץ סוגי הרכב,נתחיל מהמצביע למקסימום של תת עץ המכירות של סוג הרכב המדובר ולפי האלגוריתם לעיל נחזיר את הדגם הקטן ביותר המקיים את אותו כמות מכירות $O(log(n))$
לכן סך הכל:

$typeid = 0: O(1)$
$typeid \neq 0: O(log(n))$

StatusType GetWorstModels(void *DS, int numOfModels , int *types, int *models)
$O(m)$
על מנת לקבל את הדגמים הגרועים ביותר גם מהדגמים שנמכרו וגם אלו שלא, נבצע מיזוג (merge) בין עץ המכירות הכללי (עץ ב') לבין עץ סוגי הרכב שלא נמכרו (עץ ד').
נתחיל מהמצביע לאיבר הכי קטן בעץ ב' (שמרנו אותו ולכן $O(1)$) ונבצע הדפסה בסיור inorder החל ממנו. כאשר ניתקל בציון 0 נשמור את סוג הדגם ומספר הרכב המדובר, נעבור לעץ הרכבים שטרם נמכרו(ציונם 0),נדפיס את הצמתים עד סוג רכב וסוג דגם זה, נשמור מצביע לאחרון שהדפסנו מסוג זה,ונחזור להדפסת הצומת בעץ הראשי.
כך נמשיך עד להדפסת m צמתים.
מכיוון ששומרים מצביעים לכל מקום ייקח לנו $O(m)$ להדפסה

void Quit(void **DS)

$O(n + m)$

למחוק את כל העצים.
עץ הציונים הראשי $O(m)$
עץ המכירות הראשי $O(m)$
שלושת עצי ה car Types כאשר במקסימום יהיו בתת עץ אחד M דגמים (של סוג רכב זה לא סך הדגמים! ולכן קטן מ$(O(n + m)$
לכן סהכ:

$O(n + m)$

סיבוכיות המקום:
במקרה הגרוע (עבור כל פעולה אפשרית חוץ מInit)כל העצים יהיו מלאים כלומר הכל נמכר (לכן רק תת העץ שבו דגמים שטרם נמכרו יהיה ריק)
כלומר יש n צמתים בעצי סוגי הרכב וכל הדגמים נמצאים$O(n + m)$
ובעצים הראשיים ציונים ומכירות יש $O(m)$
לכן:

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