תאור של המבני נתונים:

שני המבנים המרכזיים במימוש הם טבלת ערבול $artistsTable$ המכיל את כלל האומנים שנוספו למערכת,
ועץ דרגות $rankedSongsTree\ AVL$ המכיל את כלל השירים של כל האומנים במערכת, מסודר לפי הסדר המתואר בפונקציה $GetRecommendedSongInPlace$ (לפי מספר השמעות, מספר מזהה של אומן, ומספר השיר).

- $artistsTable$:

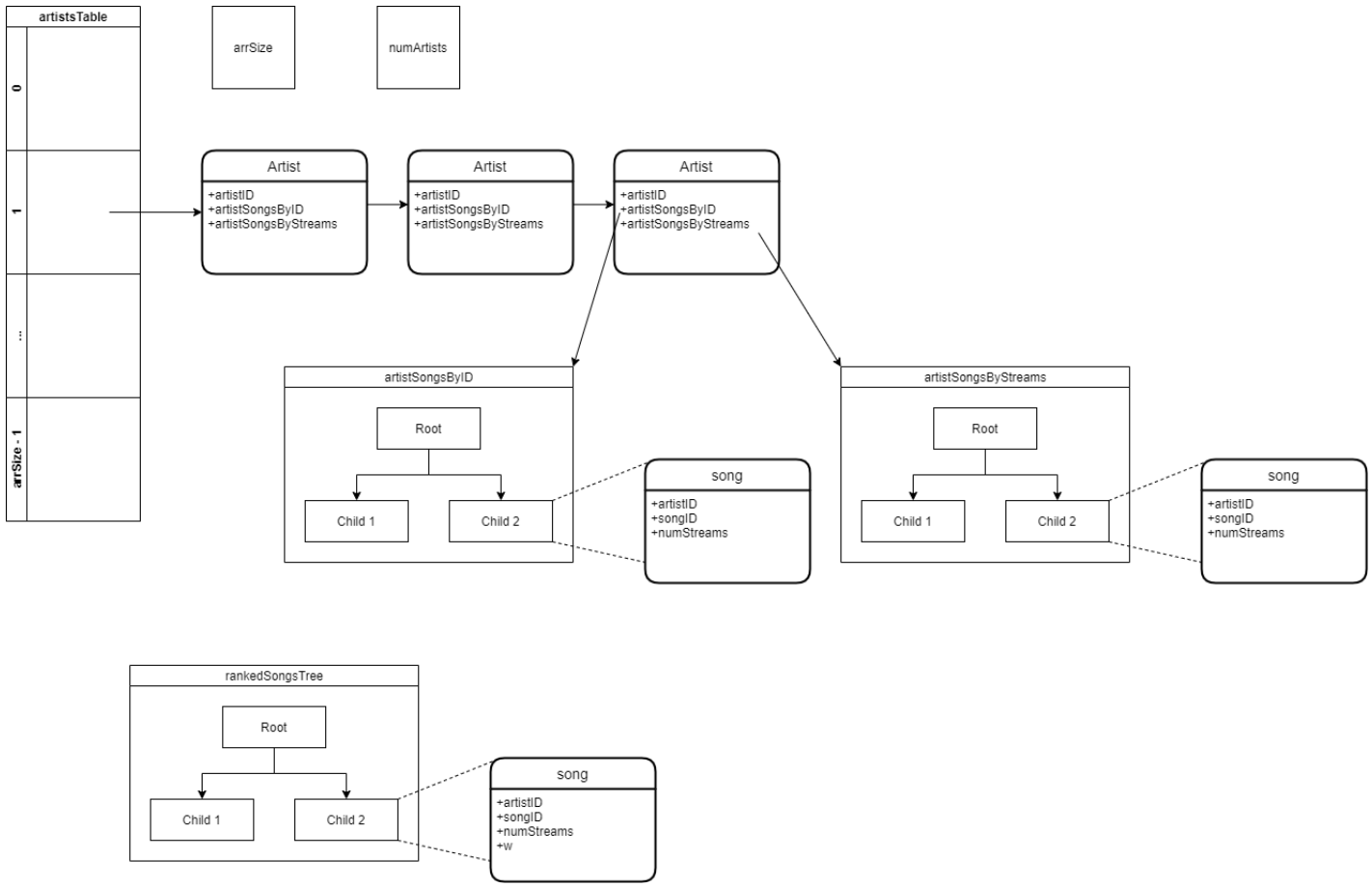
  - טבלת ערבול דינמית מסוג שרשראות המכיל את כלל האומנים שנוספו למערכת. כל אומן מיוצג ע"י אובייקט $artist$ בעל השדות הבאות: מפתח שהוא מספר המזהה של האומן $artistID$, עצי $artistSongsByID\ AVL$ ו-$artistSongsByStreams$ המכילים את כלל השירים של אותו אומן – הראשון ממוין לפי מספר מזהה של השירים בסדר עולה, והשני ממוין בסדר עולה לפי ההשמעות של השירים ולפי מספר השיר אם ההשמעות זהות, ושדה $mostStreamed$ המצביע לשיר עם הכי הרבה השמעות ב-$artistSongsByStreams$. כל שיר בהעצים מיוצג ע"י אובייקט $song$ בעל השדות $songID$ (מספר מזהה של השיר), $artistID$ (מספר מזהה של האומן של השיר), ו-$numStreams$ (מספר ההשמעות של אותה שיר).
  - בנוסף $artistsTable$ מכילה שדה $arrSize$ גודל של המערך כעת, ושדה $numArtists$ המכיל את מספר האומנים כעת במערכת. בנוסף, ל-$artistsTable$ יש פונקציית ערבול h המוגדר כ-

$$h(x) = x \cdot \mod arrSize$$

אם $numArtists$ שווה ל-$arrSize$ אז נגדיל את המערך פי 2 ואם $numArtists$ שווה ל$arrSize/4$ נקטין את המערך פי 2. כך תמיד $numArtists = O(arrSize)$, ולכן במקרה שלנו $\alpha = O(1)$. ובנוסף לפי התירגול ה-$h(x)$ שלנו מקיים את הנחת פיזור האחיד הפשוט ולכן פעלולות ההוספה ומחיקה מהמערך לוקחים $O(\alpha) = O(1)$ זמן בממוצע על הקלט משוערכת ופעולת החיפוש לוקח $O(\alpha) = O(1)$ זמן בממוצע על הקלט.

- $rankedSongsTree$:

  - עץ דרגות AVL המכיל את כלל השירים של כל האומנים במערכת,
  מסודר לפי הסדר המתואר בפונקציה $GetRecommendedSongInPlace$ (לפי מספר השמעות, מספר מזהה של אומן, ומספר השיר). כל שיר מיוצג ע"י אותו אובייקט $song$ המתואר למעלה , כאשר כל השדות של $song$ ביחד הם המפתח של אותה שיר.



איור 1: מבנה הנתונים

חישוב סיבוכיות מקום של המבנה נתונים:

נשים לב לעובדה שבכל עת, קיים בדיוק עותק אחד של כל אומן במערכת, וקיים בדיוק שלושה עותקים של כל שיר במערכת (אחד ב-$artistSongsByStreams$, אחד ב-$artistSongsByID$, ואחת ב-$rankedSongsTree$). לכן, אם m מספר האומנים כעת במערכת ו-n מספר השירים הכולל כעת במערכת, אז עבור סיבוכיות המקום הכולל של המערכת נקבל:

$$O\left(3n+m\right)=O\left(n+m\right)$$

תאור מימוש הפונקציות:

עבור כל הפונקציות שמקבלות פרמטרים, נבצע בדיקת תקינות הפרמטרים. במקרה שאחד או יותר מהפרמטרים לא תקינים, נחזיר $INVALIDINPUT$. הבדיקה הזאות היא תמיד מספר קבוע של פעולות.
בנוסף, עבור כל הפונקציות שיש להם ערך החזרה, במקרה שהפונקציה מסיימת בהצלחה נחזיר $SUCCESS$, שזה מספר קבוע של פעולות.
בכל חישוב של סיבוכיות, n_1 מסמן מספר השירים של אומן מסוים, ו-n מסמן מספר השירים בכללי במערכת

$Init()$:

1. ניצור $artistsTable$ ריקה עם מערך ערבול בגודל התחלתי 10 ($O(1)$ זמן לפי התירגול).
2. נאתחל את $arrSize$ ל-10 ו-$numArtists$ ל-0 ($O(1)$ זמן).
3. ניצור $rankedSongsTree$ ריקה ($O(1)$ זמן לפי התירגול).

סה"כ קיבלנו שסיבוכיות הזמן של כל הפונקציה היא $O(1)$

$AddArtist()$:

1. נבדוק אם כבר קיים האומן עם המספר המזהה הנתון ב-$artistsTable$. אם כן, נחזיר $FAILURE$. (סה"כ $O(1)$ זמן בממוצע על הקלט כי לפי התירגול פעולת חיפוש על טבלת ערבול שרשראות לוקחת $O(1)$ זמן בממוצע על הקלט)
2. אחרת ניצור $artist$ חדש עם $artistID$ השווה למספר מזהה הנתון ונוסיף אותו ל-$artistsTable$ באותו אופן שהוגדר הוספה לטבלת ערבול שרשראות בתירגול, ואז נקדם את $numArtists$ ב-1. אם אחרי ההוספה של האחד $numArtists$ שווה ל-$arrSize$ אז נכפיל את גודל המערך פי שתיים כמו שהוגדר בתירגול לגבי מערכים דינמיים, ואז נעדכן את $arrSize$ לגודל החדש (סה"כ $O(1)$ זמן בממוצע על הקלט משוערכת כי לפי התירגול פעולת הוספה לטבלת ערבול שרשרות דינמית לוקחת $O(1)$ זמן משוערכת בממוצע על הקלט).
3. נאתחל את $artistSongsByID$, ו-$artistSongsByID$ להיות ריקים (סה"כ $O(1)$ זמן כי לפי התירגול ייצור של עץ AVL ריק לוקחת $O(1)$ זמן)
4. נצביע את $mostStreamed$ ל-$NULL$ ($O(1)$ זמן)

סה"כ קיבלנו שסיבוכיות הזמן של כל הפונקציה היא $O(1)$ זמן בממוצע על הקלט משוערכת.

$RemoveArtist()$:

1. נבדוק אם קיים האומן עם המספר המזהה הנתון ב-$artistsTable$. אם לא קיים או ש-$mostStreamed$ של האומן היא לא $NULL$, נחזיר $FAILURE$. (סה"כ $O(1)$ זמן בממוצע על הקלט כי לפי התירגול פעולת חיפוש על טבלת ערבול שרשראות לוקחת $O(1)$ זמן בממוצע על הקלט)

2. אחרת נמחוק את האומן מ-$artistsTable$ ונוריד את $numArtists$ ב-1 . אם אחרי ההורדה $numArtists$ קטן או שווה ל-$arrSize/4$ וגדול מ-3 נחלק את גודל המערך של $artistsTable$ בחצי כמו שהוגדר בתירגול לגבי מערכים דינמיים, ואז נעדכן את $arrSize$ לגודל החדש. (סה"כ $O(1)$ זמן בממוצע על הקלט משוערכת כי לפי התירגול פעולת מחיקה מטבלת ערבול שרשראות דינמית לוקחת $O(1)$ זמן משוערכת בממוצע על הקלט , והמחיקה של העצים הפנימיים הריקים של $artist$ לוקחים $O(1)$ זמן בגרוע).

סה"כ קיבלנו שסיבוכיות הזמן של כל הפונקציה היא $O(1)$ זמן בממוצע על הקלט משוערכת.

$AddSong()$:

1. נבדוק אם קיים האומן עם המספר המזהה הנתון ב-$artistsTable$. אם לא קיים, נחזיר $FAILURE$. (סה"כ $O(1)$ זמן בממוצע על הקלט כי לפי התירגול פעולת חיפוש על טבלת ערבול שרשראות לוקחת $O(1)$ זמן בממוצע על הקלט). אם האומן קיים אבל כבר קיים לו שיר עם אותו מספר מזהה ב-$artistSongsByID$, נחזיר גם כן $FAILURE$. (סה"כ $O(log(n_1))$ זמן בגרוע כי לפי התירגול פעולת חיפוש בעץ AVL לוקחת $O(log(n))$ זמן בגרוע).

2. אחרת נוסיף את השיר קודם ל-$rankedSongsTree$ (סה"כ $O(log(n))$ זמן בגרוע כי לפי התירגול פעולת הכנסה לעץ דרגות AVL לוקחת $O(log(n))$ זמן בגרוע). אחרי זה נוסיף את השיר ל-$artistSongsByID$ ול-$artistSongsByStreams$ (סה"כ $O(log(n_1))$ זמן בגרוע כל אחד כי לפי התירגול פעולת הכנסה לעץ AVL לוקחת $O(log(n_1))$ זמן בגרוע).

3. אם לשיר החדש יש את הכי הרבה השמעות מכל השירים של האומן אז נצביע את $mostStreamed$ לצומת של השיר ב-$artistSongsByStreams$ (סה"כ $O(log(n_1))$ זמן בגרוע כי סיור בעץ AVL מהשורש לצומת הכי גבוהה לוקחת $O(log(n_1))$ זמן בגרוע).

סה"כ קיבלנו שסיבוכיות הזמן של כל הפונקציה היא $O(log(n))$ זמן בבגרוע, ולכן גם $O(log(n))$ בממוצע על הקלט.

$RemoveSong()$:

1. נבדוק אם קיים האומן עם המספר המזהה הנתון ב-$artistsTable$. אם לא קיים, נחזיר $FAILURE$. (סה"כ $O(1)$ זמן בממוצע על הקלט כי לפי התירגול פעולת חיפוש על טבלת ערבול שרשראות לוקחת $O(1)$ זמן בממוצע על הקלט). אם האומן קיים אבל לא קיים לו שיר עם אותו מספר מזהה ב-$artistSongsByID$, נחזיר גם כן $FAILURE$. (סה"כ $O(log(n_1))$ זמן בגרוע כי לפי התירגול פעולת חיפוש בעץ AVL לוקחת $O(log(n_1))$ זמן בגרוע).

2. אחרת נמחוק את השיר קודם מ-$rankedSongsTree$ (סה"כ $O(log(n))$ זמן בגרוע כי לפי התירגול פעולת מחיקתה מעץ דרגות AVL לוקחת $O(log(n))$ זמן בגרוע). אחרי זה נמחוק את השיר ב-$artistSongsByID$ וב-$artistSongsByStreams$ (סה"כ $O(log(n_1))$ זמן בגרוע כל אחד כי לפי התירגול פעולת המחיקה בעץ AVL לוקחת $O(log(n_1))$ זמן בגרוע).

3. אם לשיר שמחקנו היה את הכי הרבה השמעות מכל השירים של האומן אז נצביע את $mostStreamed$ לצומת שעכשיו יש את הכי הרבה השמעות ב-$artistSongsByStreams$ (סה"כ $O(log(n_1))$ זמן בגרוע כי סיור בעץ AVL מהשורש לצומת הכי גבוהה לוקחת $O(log(n_1))$ זמן בגרוע).

סה"כ קיבלנו שסיבוכיות הזמן של כל הפונקציה היא $O(log(n))$ זמן בבגרוע, ולכן גם $O(log(n))$ בממוצע על הקלט.

$AddToSongCount()$:

1. נבדוק אם קיים האומן עם המספר המזהה הנתון ב-$artistsTable$. אם לא קיים, נחזיר $FAILURE$. (סה"כ $O(1)$ זמן בממוצע על הקלט כי לפי התירגול פעולת חיפוש על טבלת ערבול שרשראות לוקחת $O(1)$ זמן בממוצע על הקלט). אם האומן קיים אבל לא קיים לו שיר עם אותו מספר מזהה ב-$artistSongsByID$, נחזיר גם כן $FAILURE$. (סה"כ $O(log(n_1))$ זמן בגרוע כי לפי התירגול פעולת חיפוש בעץ AVL לוקחת $O(log(n_1))$ זמן בגרוע).

2. אחרת קודם נוריד את השיר מ-$rankedSongsTree$, נוסיף לו $count$ השמעות, ואז נוסיף מחדש ל-$rankedSongsTree$ (ולכן נעדכן את מיקומו בעץ) (סה"כ $O(log(n))$ זמן בגרוע כי לפי התירגול פעולות מחיקה והוספה לעץ דרגות AVL לוקחות $O(log(n))$ זמן בגרוע). אחרי זה נוריד את השיר מ-$artistSongsByStreams$, ואז נוסיף מחדש ל-$artistSongsByStreams$ (ולכן נעדכן את מיקומו בעץ) (סה"כ $O(log(n_1))$ זמן בגרוע כי לפי התירגול פעולות מחיקה והוספה לעץ AVL לוקחות $O(log(n_1))$ זמן בגרוע).

3. אם לשיר שהוספנו לו השמעות כעת יש את הכי הרבה השמעות מכל השירים של האומן אז נצביע את $mostStreamed$ אליו ב-$artistSongsByStreams$ (סה"כ $O(log(n_1))$ זמן בגרוע כי סיור בעץ AVL מהשורש לצומת הכי גבוהה לוקחת $O(log(n_1))$ זמן בגרוע).

סה"כ קיבלנו שסיבוכיות הזמן של כל הפונקציה היא $O(log(n))$ זמן בבגרוע, ולכן גם $O(log(n))$ בממוצע על הקלט.

$GetArtistBestSong()$:

1. נבדוק אם קיים האומן עם המספר המזהה הנתון ב-$artistsTable$. אם לא קיים, נחזיר $FAILURE$. (סה"כ $O(1)$ זמן בממוצע על הקלט כי לפי התירגול פעולת חיפוש על טבלת ערבול שרשראות לוקחת $O(1)$ זמן בממוצע על הקלט). אם האומן קיים אבל אין לו שירים, נחזיר גם כן $FAILURE$. (סה"כ $O(1)$ זמן בגרוע כי זה בדיקה האם ה-$mostStreamed$ של האומן היא $NULL$).

2. אחרת נציב את המספר המזהה של השיר שאליו מצביע $mostStreamed$ ב-$songID$ (סה"כ $O(1)$ זמן בגרוע).

סה"כ קיבלנו שסיבוכיות הזמן של כל הפונקציה היא $O(1)$ זמן בממוצע על הקלט.

$GetRecommendedSongInPlace()$:

1. נבדוק אם מספר השירים ב-$rankedSongsTree$ הוא קטן מ-$rank$. אם כן, נחזיר $FAILURE$. (סה"כ $O(1)$ זמן בגרוע כי נדרש רק בדיקה האם $rank$ גדול משדה ה-w של השורש של $rankedSongsTree$).

2. אחרת נחפש את השיר אם דרגת $rank$ ב-$rankedSongsTree$. אחרי שמצאנו אותו נציב ב-$artistID$ את מספר המזהה של האומן של השיר וב-$songID$ את מספר המזהה של השיר עצמו (סה"כ $O(log(n))$ זמן בגרוע כי לפי התירגול פעולת חיפוש של צומת לפי דרגה בעץ דרגות AVL לוקחת $O(log(n))$ זמן בגרוע).

סה"כ קיבלנו שסיבוכיות הזמן של כל הפונקציה היא $O(log(n))$ זמן בגרוע.

$Quit()$:

1. נמחוק את כל המבני נתונים. קיים בדיוק m אובייקטים המייצגים ה-m אומנים במבני הנתונים, וקיים בדיוק $3n$ אובייקטים המייצגים n השירים הכולל במבני הנתונים. לכן מחיקה של כל המבני נתונים ייקח $O(m+n)$ זמן.