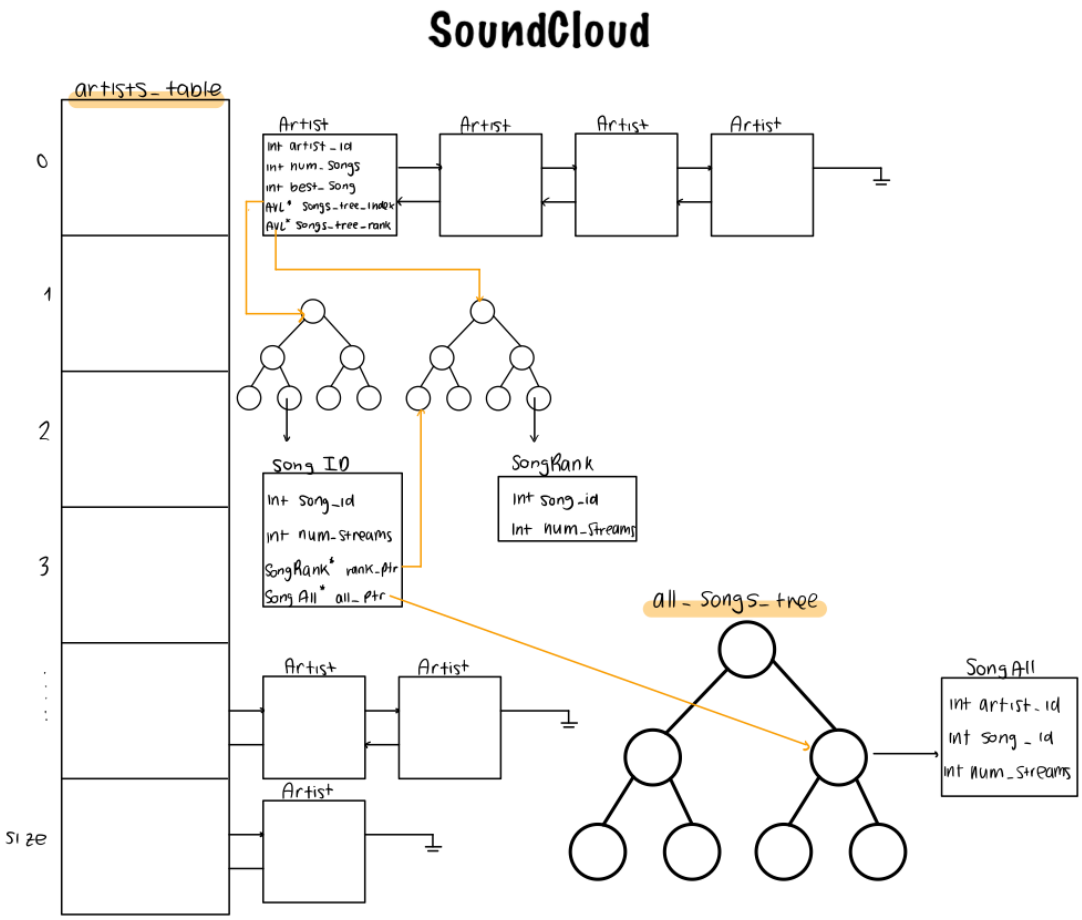
DS 234218 wet 2

Carmel David 207174327
Noam Raveh 313599573

שרטוט מבנה הנתונים



תיאור מבנה הנתונים

מבנה הנתונים הראשי SoundCloud מורכב משני מבנים עיקריים:

- טבלת גיבוב דינאמית (Hash table) בשם artists_table בה כל איבר מצביע לרשימה מקושרת דו כיוונית המכילה חוליות המחזיקות מידע על כל אמן.
הטבלה מתחילה בגודל של 7 חוליות ומכפילה את עצמה כאשר מתמלאה, וכן מחלקת את עצמה כאשר מכילה מספר איברים השווה לרבע מגודלה הנוכחי.
האיברים בטבלה מפוזרים בין הרשימות המקושרות השונות על ידי פונקציית ערבול אשר מחזירה לכל מפתח את שארית החלוקה שלו בגודל הטבלה, ומכניסה את המפתח ואת הערך המתאים לראש הרשימה הנמצאת בתא שחושב.

פונקציה זו עומדת ב"הנחת הפיזור האחיד" וכן חישובה נעשה בסיבוכיות זמן O(1).

- עץ דרגות AVL המכיל מידע על כל שיר השמור במערכת ברגע נתון.
  החוליות בעץ ממוינות באופן הבא
  - מיון ראשוני לפי מספר השמעות בסדר יורד
  - מיון שני לפי מזהה אמן בסדר עולה
  - מיון שלישי לפי מזהה שיר בסדר עולה

  כפי שנלמד עבור עץ דרגות, כל חוליה בעץ מחזיקה בנוסף למידע השמור בה ולמצביעים לבנים ולאב גם משתנה השומר את מספר החוליות בתת העץ שהיא השורש שלו.

Artist Class

מחלקה לייצוג המידע השמור לכל אמן.

- Int id - מזהה יחודי של כל אמן.
- Int num_songs – מספר השירים השמורים במערכת לאמן.
- Int best_song – מזהה השיר המדורג גבוה ביותר מבין שירי האמן (מספר השמעות הכי גבוה, ובמקרה של שוויון, מזהה שיר הכי נמוך).
- AVLTree<SongID>* songs_tree_index – מצביע לעץ השומר את שירי האמן, מדורגים לפי מזהה השיר.
- AVLTree<SongRank>* songs_tree_rank – מצביע לעץ השומר את שירי האמן, מדורגים לפי מספר השמעות בסדר יורד ומזהה שיר בסדר עולה. משמש לשמירת השיר המושמע ביותר של האמן.

SongID Class

מחלקה לייצוג השירים השמורים בעץ של כל אמן הממויין לפי מזהה השיר.

- Int song_id - מזהה יחודי של כל שיר.
- Int num_streams – מספר ההשמעות של השיר במערכת.
- SongRank* song_rank_tree_ptr – מצביע לחוליה בעץ השירים הממויין לפי דירוגם.
- SongAll* all_songs_tree_ptr – מצביע לחוליה בעץ השירים הראשי של המערכת השומר את כל השירים ממויינים לפי דירוגם.

SongRank Class

מחלקה לייצוג השירים השמורים בעץ של כל אמן הממויין לפי דירוג השירים.

- Int song_id - מזהה יחודי של כל שיר.
- Int num_streams – מספר ההשמעות של השיר במערכת.

SongAll Class

מחלקה לייצוג השירים השמורים בעץ של כלל המערכת הממויין לפי דירוג השירים.

- Int artist_id – מזהה יחודי של כל אמן.
- Int song_id - מזהה יחודי של כל שיר.
- Int num_streams – מספר ההשמעות של השיר במערכת.

פירוט מימוש הפונקציות

*עבור כל הפונקציות המתוארות, במידה והפעולה הצליחה בשלמותה, הפונקציה גתחזיר SUCCESS. בנוסף, במידה והתבצעו הקצאות זיכרון שנכשלו, הפונקציה תחזיר ALLOCATION_ERROR והמבנה יוחזר לקדמותו.

Void* Init

- מימוש:
  - קריאה לבנאי של המחלקה SoundCloud אשר קורא לבנאים של טבלת הגיבוב ושל העץ, והחזרת מצביע לאובייקט שהוא יצר או nullptr במידה ולא הצליח ליצור כזה.
- דרישות סיבוכיות זמן:
  - מדובר במספר קבוע של פעולות שאינו תלוי בגודל הקלט, ולכן סיבוכיות הזמן היא O(1).

StatusType AddArtist(void* DS, int artistID)

- מימוש
  - בדיקת הקלט והחזרת שגיאה INVALID_INPUT במידה ואינו תקין. כמו כן, בדיקה אם האמן כבר קיים במערכת והחזרת שגיאה FAILURE במידה ונמצא שכן.
  - יצירת אובייקט מסוג Artist והכנסתו אל טבלת הגיבוב בהתאם לפונקציית הערבול. במידת הצורך, כאשר התא המתאים ריק, תיווצר רשימה מקושרת חדשה ואליה יוכנס המידע.
- דרישות סיבוכיות זמן:
  - בדיקת תקינות הקלט דורשת מספר קבוע של פעולות ולכן נעשית בסיבוכיות זמן O(1).
  - כפי שנלמד בהרצאה, חיפוש והכנסה לטבלת ערבול עם פונקציית ערבול מתאימה נעשים ב(O(1 בממוצע משוערך על הקלט.

לכן בסך הכל סיבוכיות הזמן של הפונקציה היא O(1) משוערך על הקלט בממוצע.

StatusType RemoveArtist(void* DS, int artistID)

- מימוש
  - בדיקת הקלט והחזרת שגיאה INVALID_INPUT במידה ואינו תקין. כמו כן, בדיקה אם האמן כבר קיים במערכת והחזרת שגיאה FAILURE במידה ונמצא שלא.
    בנוסף, במידה וקיימים שירים לאמן זה, נחזיר שגיאה FAILURE.
  - קריאה לפונקציית המחיקה של טבלת הערבול עם מזהה האמן.
- דרישות סיבוכיות זמן:
  - בדיקת תקינות הקלט דורשת מספר קבוע של פעולות ולכן נעשית בסיבוכיות זמן O(1).
  - כפי שנלמד בהרצאה, חיפוש ומחיקה מטבלת ערבול נעשים ב(O(1 משוערך על הקלט בממוצע.
  - לכל אמן אנו שומרים את מספר השירים, ולכן ניתן לדעת ב(O(1 האם קיימים עבורו שירים.

בסך הכל סיבוכיות הזמן של הפונקציה היא O(1) משוערך על הקלט בממוצע.

StatusType AddSong(void* DS,int artistID, int songID

- מימוש
  - בדיקת הקלט והחזרת שגיאה INVALID_INPUT במידה ואינו תקין. כמו כן, בדיקה אם האמן כבר קיים במערכת והחזרת שגיאה FAILURE במידה ונמצא שלא.
    בנוסף בדיקה האם השיר כבר קיים עבור אמן זה, והחזרת FAILURE במידה ונמצא שכן.
  - חיפוש האמן בטבלת הערבול וגישה למידע שלו.
  - יצירת אובייקטים מסוג SongRank (עם 0 השמעות) וSongID והכנסתם לעצים של שירי האמן ממויינים לפי דירוג ומזהה בהתאמה.
  - יצירת אובייקט מסוג SongAll עם 0 השמעות והכנסתו לעץ של כל השירים במערכת.
  - עדכון המצביעים תחת אובייקט SongID החדש שיצרנו למידע המתאים מסוג SongRank וSongAll לצורך גישה אליהם ישירות מאובייקט השיר.
  - הגדלת מונה מספר השירים של האמן באחד.
  - עדכון best_song על ידי גישה לאיבר המינימלי בעץ הממויין לפי הדירוג של אותו אמן, ושמירת מזהה השיר תחתיו.
- דרישות סיבוכיות זמן
  - בדיקת תקינות הקלט דורשת מספר קבוע של פעולות ולכן נעשית בסיבוכיות זמן O(1).
  - כפי שנלמד בהרצאה, חיפוש בטבלת ערבול לצורך מציאת האמן נעשה בO(1) משוערך על הקלט בממוצע.
  - הוספה של אובייקטי השירים לעצים המתאימים תעשה בO(log(n)) במקרה הגרוע , כפי שנלמד בהרצאה על עצי AVL, כאשר n הוא מספר השירים הכולל במערכת.
  - עדכון המצביעים וכן הגדלת המונה יעשו במספר קבוע של פעולות, כלומר בזמן O(1).
  - גישה לאיבר המינימלי בעץ תעשה ב O(1) שכן אנו מחזיקים בעץ מצביע לצומת המינימלי.
    בסך הכל נקבל שסיבוכיות הזמן היא $O(1)_{average}+3*O(log(n))_{wc}+O(1)$ *ולכן בסך הכל* $O(log(n))_{average}$.

StatusType RemoveSong(void* DS,int artistID, int songID

- מימוש
  - בדיקת הקלט והחזרת שגיאה INVALID_INPUT במידה ואינו תקין. כמו כן, בדיקה אם האמן כבר קיים במערכת והחזרת שגיאה FAILURE במידה ונמצא שלא.
    בנוסף בדיקה האם השיר כבר קיים עבור אמן זה, והחזרת FAILURE במידה ונמצא שלא.
  - חיפוש האמן בטבלת הערבול וגישה למידע שלו.
  - חיפוש השיר תחת עץ השירים של האמן (הממויין לפי מזהי השירים).
  - גישה למצביעים השמורים תחת השיר שמצאנו אל העץ songs_tree_rank וall_songs_tree והסרת הצמתים המתאימים מהעצים.
  - הסרת השיר מעץ השירים songs_tree_id של האמן.
  - הקטנת מונה מספר השירים של האמן באחד.
  - עדכון best_song על ידי גישה לאיבר המינימלי בעץ הממויין לפי הדירוג של אותו אמן, ושמירת מזהה השיר תחתיו.

- דרישות סיבוכיות זמן
  - בדיקת תקינות הקלט דורשת מספר קבוע של פעולות ולכן נעשית בסיבוכיות זמן O(1).
  - כפי שנלמד בהרצאה, חיפוש בטבלת ערבול לצורך מציאת האמן נעשה בO(1) משוערך על הקלט בממוצע.
  - הסרה של אובייקטי השירים מהעצים המתאימים תעשה בO(log(n)) במקרה הגרוע , כפי שנלמד בהרצאה על עצי AVL, כאשר n הוא מספר השירים הכולל במערכת.
  - הקטנת המונה תעשה בO(1).
  - גישה לאיבר המינימלי בעץ תעשה ב O(1) שכן אנו מחזיקים בעץ מצביע לצומת המינימלי.

בסך הכל נקבל שסיבוכיות הזמן היא $O(1)_{average}+3*O(log(n))_{wc}+O(1)$ ולכן בסך הכל $O(log(n))_{average}$.

StatusType AddToSongCount(void* DS,int artistID, int songID,int count)

- מימוש
  - בדיקת הקלט והחזרת שגיאה INVALID_INPUT במידה ואינו תקין. כמו כן, בדיקה אם האמן כבר קיים במערכת והחזרת שגיאה FAILURE במידה ונמצא שלא.
    בנוסף בדיקה האם השיר כבר קיים עבור אמן זה, והחזרת FAILURE במידה ונמצא שלא.
  - חיפוש האמן בטבלת הערבול וגישה למידע שלו.
  - חיפוש השיר בעץ songs_tree_index והחזרת מספר ההשמעות הנוכחי שלו.
  - גישה דרך השיר שנמצא למצביע אל השיר בעצים all_songs_tree ו songs_tree_rank והסרת הצמתים המתאים מהעץ.
  - יצירת אובייקטים מסוג SongAll ו SongRank חדשים, עם מספר ההשמעות המעודכן, והכנסתם לעצים all_songs_tree ו songs_tree_rank בהתאמה.
  - עדכון המצביעים תחת השיר הנמצא אל האובייקטים החדשים שיצרנו.
  - עדכון best_song על ידי גישה לאיבר המינימלי בעץ הממויין לפי הדירוג של אותו אמן, ושמירת מזהה השיר תחתיו.
- דרישות סיבוכיות זמן
  - בדיקת תקינות הקלט דורשת מספר קבוע של פעולות ולכן נעשית בסיבוכיות זמן O(1).
  - כפי שנלמד בהרצאה, חיפוש בטבלת ערבול לצורך מציאת האמן נעשה בO(1) משוערך על הקלט בממוצע.
  - כפי שנלמד, חיפוש, הסרה והוספה של צמתים מעצים יעשו בסיבוכיות זמן O(log(n)) במקרה הגרוע, כאשר n הוא גודל העץ. נבצע חיפוש אחד, וכן שתי הסרות ושתי הוספות
  - עדכון וגישה למצביעים נעשה בסיבוכיות זמן O(1).
  - גישה לאיבר המינימלי בעץ ועדכונו יעשו ב O(1) שכן אנו מחזיקים בעץ מצביע לצומת המינימלי.

בסך הכל נקבל שסיבוכיות הזמן היא $O(1)_{average}+5*O(log(n))_{wc}+O(1)$ ולכן בסך הכל $O(log(n))_{average}$.

StatusType GetArtistBestSong(void* DS,int artistID,int* songID)

- מימוש
  - בדיקת הקלט והחזרת שגיאה INVALID_INPUT במידה ואינו תקין. כמו כן, בדיקה אם האמן כבר קיים במערכת והחזרת שגיאה FAILURE במידה ונמצא שלא.
  - בנוסף, נבדוק האם קיימים שירים תחת האמן ונחזיר FAILURE במידה ונמצא שלא.
  - חיפוש האמן בטבלת הערבול וגישה למידע שלו.
  - החזרת הערך השמור במידע האמן תחת best_song אל המצביע שקיבלנו לsongID.
- דרישות סיבוכיות זמן
  - בדיקת תקינות הקלט דורשת מספר קבוע של פעולות ולכן נעשית בסיבוכיות זמן O(1).
  - כפי שנלמד בהרצאה, חיפוש בטבלת ערבול לצורך מציאת האמן נעשה בO(1) משוערך על הקלט בממוצע.
  - עדכון המצביע יעשה בO(1).

  בסך הכל נקבל שסיבוכיות הזמן היא O(1) משוערך על הקלט בממוצע.

StatusType GetRecommendedSongInPlace(void* DS,int rank,int* artistID, int* songID)

- מימוש
  - בדיקת הקלט והחזרת שגיאה INVALID_INPUT במידה ואינו תקין. כמו כן, בדיקה אם מספר השירים הכולל במערכת גדול מrank והחזרת FAILURE במידה ונמצא שלא.
  - חיפוש השיר שדרגתו היא rank בעץ all_songs כפי שנלמד בתרגול על ידי פונקציית select המוגדרת בעץ הדרגות.
  - עדכון המצביעים artistID וsongID בהתאם לפרטי השיר שהוחזר מהפונקציה select.
- דרישות סיבוכיות זמן
  - בדיקת תקינות הקלט דורשת מספר קבוע של פעולות ולכן נעשית בסיבוכיות זמן O(1).
  - כפי שנלמד בהרצאה, אלגוריתם החיפוש של צומת מדרגה מבוקשת בעץ דרגות מסוג AVL מתבצע בסיבוכיות זמן O(log(n)) במקרה הגרוע.
  - עדכון המצביעים יעשה בO(1).

  לכן בסך הכל נקבל שסיבוכיות הזמן של הפונקציה היא O(log(n)) במקרה הגרוע.

Void Quit(void **DS)

- מימוש
  - תתבצע קריאה להורס של SoundCloud, אשר יקרא להורסים של טבלת הערבול ושל עץ הדרגות מסוג AVL. באופן רקורסיבי יקראו ההורסים של כל שדה פנימי במערכת.
- דרישות סיבוכיות זמן
  - לפי הגדרת התרגיל, ישנם m אמנים, ולכן כפי שנלמד מחיקת טבלת ערבול המכילה m איברים המפוזרים ברשימות מקושרות שונות תעשה בסיבוכיות זמן O(m).
  - לפי הגדרת התרגיל, ישנם n שירים במערכת, ולכן כפי שנלמד מחיקת העץ הכולל all_songs תעשה בסיבוכיות זמן של מספר האיברים בעץ – O(n).

בנוסף, עבור כל אמן מוחזקים שני עצים המכילים את כל השירים שלו. היות ובסך הכל, מספר השירים של כל האמנים הוא n, מחיקת כל אותם עצים תעשה בסיבוכיות זמן 2*O(n) = O(n)

*בסך הכל נקבל שסיבוכיות הזמן של הפונקציה היא 3*O(n) + O(m) = O(n+m) במקרה הגרוע.*

דרישות סיבוכיות מקום

מבנה הנתונים שלנו מכיל שני תתי מבנים עיקריים. נראה שכל אחד מהם, ושניהם ביחד, עומדים בסיבוכיות המקום.

- עץ השירים all_songs_tree
  - העץ מכיל את כל השירים הנמצאים במערכת ברגע נתון ולכן גודלו הוא לכל היותר n, כאשר n מוגדר בתרגיל להיות מספר השירים.

בכל הוספה של שיר למערכת נוסיף את השיר לעץ זה וכן בכל הסרה של שיר נוציא את חוליית השיר מהעץ ונמחק אותה, ולכן נקבל שבכל רגע נתון מספר החוליות בעץ הוא כמספר השירים העדכני באותו רגע.

תחת כל צומת בעץ הזה, נשמרים מספר קבוע של שדות, ולכן נקבל בסך הכל שסיבוכיות המקום שלו היא O(n).

- טבלת ערבול artists_table
  - טבלת הערבול שהגדרנו הינה טבלת ערבול דינמית. כלומר, בכל רגע נתון לאחר ביצוע הכנסה או הוצאה, גודל הטבלה הוא לכל היותר פעמיים מספר האיברים שהכנסנו לטבלה (מספר האמנים באותו רגע, m)
    - כאשר נקבל לאחר הכנסה שגודל הטבלה הנוכחי שווה למספר האיברים שהוכנסו, נגדיל את הטבלה פי 2 ובכך נקבל שלאחר הגדלה גודלה הוא פעמים מספר האיברים שהוכנסו.
    - כאשר נקבל לאחר ביצוע הוצאה שגודל הטבלה הנוכחי שווה לרבע ממספר האיברים שהוכנסו, נקטין את הטבלה למחצית גודלה ובכך נקבל שלאחר ההקטנה גודלה הוא פעמים מספר האיברים שהוכנסו.

לכן, נקבל שבכל רגע נתון גודל הטבלה הוא O(m), כאשר m מספר האמנים באותו רגע.

  - תחת כל האמן המוחזק בטבלת הערבול, ישנם מספר קבוע של שדות, וכן שני עצים המחזיקים את השירים של כל אמן.

בסך הכל, כל האמנים ביחד מחזיקים עצים בגודל 2n, כאשר n הוא מספר השירים הכולל במערכת באותו רגע.

גם כאן, היות ואנו מסירים מהעצים של האומן המתאים את השירים אותם התבקשנו להסיר מהמערכת, גודל כל העצים ביחד לא יעלה על 2n בכל רגע נתון.

תחת כל חוליה המייצגת את השירים בעצים, ישנו מספר קבוע של שדות.

לכן נקבל שבסך הכל עבור כל האמנים ביחד סיבוכיות המקום עבור עצי השירים שלהם היא 2*O(n) = O(n)

בסך הכל עבור הטבלה וכל המידע המוחזק בה, נקבל סיבוכיות מקום של $O(n)+O(m) = O(n+m)$ במקרה הגרוע כאשר m מספר האמנים ו n מספר השירים הכולל במערכת בכל שלב.

בסך הכל, עבור הטבלה והעץ הראשי, נקבל סיבוכיות מקום כוללת של $O(n+m)+O(n) = O(n+m)$, כנדרש.