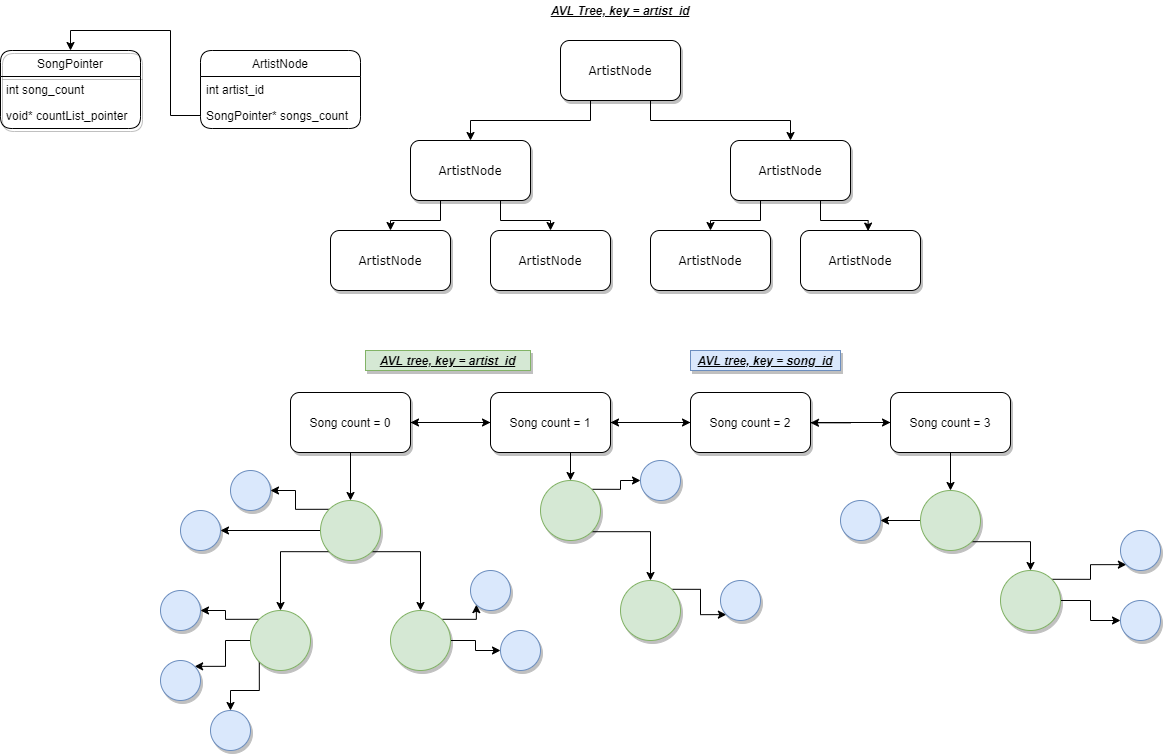
מבנה הנתונים שאשתמש בו יורכב משני חלקים :   
1. עץ AVL שכל צומת בו מכיל את הדברים הבאים

1. מזהה אמן
2. מערך בגודל m מאותחל ל-0. התא ה- i יכיל את מספר ההשמעות של השיר ה- i ובנוסף יכיל מצביע למופע של השיר במבנה השני.

2. רשימת מקושרת דו כיוונית אשר כל תא בה יכריז על מספר ההשמעות של כל השירים לדוגמה התא הראשון ברשימה יכריז לוגית על “Song count = 0 “ ויכיל בתוכו עץ AVL עם אותם שירים שאין להם השמעות כלל.  
כל איבר ברשימה הזאת מצביע לעץ AVL (העץ הירוק בדיאגרמה) אשר כל צומת בו מכילה מזהה אומן ומצביע לעץ AVL (העץ הכחול בדיאגרמה) של השירים של אותו אומן עם מספר ההשמעות המתאים.

למען הנוחות, מעתה אתייחס לעץ AVL הראשי כ- **t1**, לרשימה המקושרת כ- list, לעץ הירוק כ- main\_t ולעץ הכחול כ- sub\_t.

מימוש הפונקציות תחת חסמי הסיבוכיות :

1. **Init()**נאתחל את **t1**,main\_t,sub\_t כעצים ריקים בסיבוכיות O(1) .  
   נאתחל את list כרשימה בעלת צומת אחד בלבד המכריז על “song count =0” בסיבוכיות O(1).  
   סך הכל, הפעולה תתבצע בסיבוכיות O(1) כנדרש.
2. **StatusType AddArtist(void\* DS, int artistID, int numOfSongs)  
   נחפש** על **t1** את artistID אם לא מצאנו, נכניס אותו במקום המתאים בסיבוכיות O(logn).  
   כדי ליצור צומת ב**t1** עלינו לבנות מערך אשר יאותחל לערך 0 מה שניתן לבצע בסיבוכיות O(1) כפי שנלמד בהרצאה.  
   כעת, נוסיף את האמן אל main\_t.  
   מאחר וכל השירים של האמן הנכנס מאותחלים ל-0 השמעות, נלך לרשימה המקושרת לאיבר הראשון ונמצא מקום להכניס בה את artistID בסיבוכיות O(logn).  
   לאחר מכן, נבנה עץ AVL בעל numOfSongs צמתים כך שהשורש של עץ זה יהיה הצומת שהכנסנו בשלב הקודם בסיבוכיות O(numOfSongs =(m)) .  
   סדר הפעולות בפועל עשוי להיות הפוך מאחר שעלינו ליצור מצביעים מכל תא במערך בתוך צומת של **t1** אל הצומת המתאים ב-sub\_t של אותו אמן בסיבוכיות O(m).  
   סך הכל, הפעולה תתבצע בסיבוכיות O(logn +m).
3. **StatusType RemoveArtist(void\* DS, int artistID)  
   נחפש** על **t1** את artistID בסיבוכיות O(logn).  
   אם מצאנו, נעבור על המערך בסיבוכיות O(m).  
   בעבור כל תא במערך נקפוץ בעזרת הפוינטר אל הצומת המתאים ב-sub\_t ונמחק אותו בסיבוכיות O(1).  
   אם במהלך תהליך המחיקה הגענו למצב של עץ ריק, נמחק את השורש בסיבוכיות O(1).  
   בסיום פעולות אלו, נמחק את הצומת המתאים מ- **t1** בסיבוכיות O(m).  
   יש לציין כי בעת מחיקת צמתים מהעצים יתבצעו גלגולים בסיבוכיות O(logn).  
   סך הכל, הפעולה תתבצע בסיבוכיות O(mlogn).
4. **StatusType AddToSongCount(void\* DS, int artistID, int songID)  
   נחפש** על **t1** את artistID בסיבוכיות O(logn) .  
   אם מצאנו, נוסיף לתא songID במערך +1 בסיבוכיות O(1).  
   כעת בעזרת המצביע שמאוחסן בתא זה נקפוץ לצומת של שיר זה ב- sub\_t.  
   בהנחה שלשיר היו n השמעות לפני הקריאה לפונקציה, נבדוק האם ברשימה list קיים צומת המכריז על “Song count = n+1”. אם קיים, נכניס את artistID ל- main\_t בסיבוכיות O(logn) (אם לא קיים כבר) ונכניס את songID ל-sub\_t בסיבוכיות O(logm) .  
   אם צומת כזה לא קיים, ניצור אותו ונחבר אותו במקומו המתאים לרשימה בסיבוכיות O(1) ונבצע את פעולות ההכנסה שתיארתי בשורה מעלה.  
   סך הכל, הפעולה תתבצע בסיבוכיות O(logn + logm) .
5. **StatusType NumberOfStreams(void\* DS, int artistID, int songID, int\* streams)  
   נחפש** על **t1** את artistID בסיבוכיות O(logn).  
   אם מצאנו, נעדכן את הערך של streams להיות ערכו של התא ה- songID במערך המאוחסן בצומת בסיבוכיות O(1).  
   סך הכל, הפעולה תתבצע בסיבוכיות O(logn) .
6. **StatusType GetRecommendedSongs(void\* DS, int numOfSongs, int\* artists, int\* songs)**ניגש לסוף הרשימה list ונצבע חיפוש inorder על main\_t ונכניס למערך המתאים את כל הצמתים שלו בסדר inorder.   
   אם בשלב כלשהו הכנסו כבר numOfSongs שירים למערך נעצור ונחזיר SUCCESS .  
   אם עדיין לא הכנסנו numOfSongs שירים, נעבור לאיבר הקודם ברשימה ונחזור על פעולות ההכנסה בסדר inorder .  
   סך הכל, הפעולה מבצעת חיפושי inorder עם פעולת השמה למערך (שלוקחת בפני עצמה O(1) )  
   כאמור, החיפוש נעצר בעת שהכנסו למערך numOfSongs שירים ולכן הפעולה תתבצע ב O(numOfSongs).
7. **void Quit(void\*\* DS)**