<u>תיאור כללי של מבנה הנתונים:</u>

מבנה הנתונים שלנו מורכב מ- dynamic hash table של הגנרים, שהערכים בו הם הגנרים Genre.

ויש מבנה union-find של השירים ששומר את ה-nodes של העצים בתוך hash table דינמי, כאשר מבני העצים נקבעים ע"י מצביעים ששמורים בתוך ה-nodes האלו, בנפרד מהמצביעים המשמשים את ה-hash table. מבנה ה-union find משתמש בשיטת איחוד לפי גודל וכיווץ מסלולים, ומכיל שדה נוסף לכל node, שמשמש למציאת מספר שינויי הג'נר לכל שיר.

השתמשנו בשיטת הערבול שבה עושים mod כמו שראינו בתרגול, עם הוספת רשימות מקושרות במקרה של התנגשויות.

שני המערכים הם דינמיים כי מספר הג'רים והשירים אינו ידוע מראש, ולכן אנו לא נדע את גודל המערך מראש.

המערכים הללו אחראים לניהול הזיכרון של הנתונים במערכת, ומשחררים את כל הג'נרים והשירים השמורים בה בעת הפעלת ה-destructor של dspotify.

נגדיר את המחלקות הבאות:

המחלקה Song:

כאשר שדות המחלקה הם:

- **song_id**← מציין את הid של השיר.
- **genre_id**← מציין את הג'נר של השיר, הוא מעודכן(רלוונטי) רק עבור שיר שהוא שורש בעץ במבנה ב-UF.

המחלקה Genre:

כאשר שדות המחלקה הם:

- **genre_id** ← מציין את הid של ה-genre.
- **num_songs** ← מספר השירים השייכים ל-genre.
- **root_song** ← מכיל השיר שהוא שורש של עץ השירים של ה-Genre.

המחלקה הגנרית HashNode <K,V>:

מחלקה זו תהווה את התאים השמורים ב-hash table, ו שדות המחלקה הם:

- **key**← מכיל מזהה מסוג K של העצם מסוג V אותו ה-Node מחזיק.
- ***val** ← מכיל עצם מסוג V אותו ה-Node מחזיק, מוחזק by pointer.
- ***next** ← מצביע ל-Node העוקב ברשימה המקושרת. רלוונטי רק עבור הרשימות המקושרות.
- ***prev** ← מצביע ל-Node הקודם ברשימה המקושרת. רלוונטי רק עבור הרשימות המקושרות.

המחלקה הגנרית UFNode <V>:

מחלקה זו תהווה את צמתי העצים ההפוכים שנבנה כדי לממש את מבנה הנתונים union-find.

ושדות המחלקה הם:

- ***parent** ← מצביע ל-Node של האב של ה-Node הנוכחי. צומת שהוא שורש במבנה ב-union-find, מצביע לעצמו בשדה הזה.
- **r** ← שדה נוסף לכל צומת במבנה ה-UF, ישמש אותנו בחישוב מספר שינויי הג'נר לשיר.
- ***val** ← מכיל עצם מסוג V אותו ה-Node מחזיק, מוחזק by pointer.

הערות:

1. במהלך תיאור הפונקציות לא נציין את בדיקות הקלט שנעשות, אבל הן אכן נעשות בקוד עצמו, בהתאם להוראות של תרגיל הבית.
2. במהלך תיאור הפונקציות נניח פיזור אחיד לפונקציות הערבול.

פונקצית עזר:

update_path_from(K key, int r)

זוהי מתודה של מבנה ה-UF שתעזור לנו בתחזוק השדה הנוסף r, שהגדרנו לכל צומת במבנה ה-UF, שיעזור לנו לחשב את מספר שינויי הג'נר של השיר, בדומה לשאלה מהתרגול.
בה אנחנו מחפשים את החוליה בעלת המפתח המתאים, ומוסיפים לשדה r שלה את הערך r.
ולכן סה"כ אנו עושים מספר קבוע של עדכוני ערכים + חיפוש במבנה UF ,בעל טבלה דינמית כמערך, ולכן סיבוכיות הזמן היא O(log*n) בממוצע משוערך.

path_to()

מתודה שסוכמת את כל ערכי השדות r מהצומת עד השורש, כפי שראינו בתרגול על תחזור השדה r, מספר זה יחזיר את המספר הרצוי(מספר שינויי הג'נר של השיר). כלומר למעשה אנו נבצע find במבנה UF ,בעל טבלה דינמית כמערך, (עם כיווץ מסלולים) וסכימה של ערכי השדות בו זמנית. ולכן סיבוכיות הזמן היא O(log*n) בממוצע משוערך.

פונקציות:

DSpotify() -

נקרא לבנאי הדיפולטי, שקורא בעצמו לבנאי הדיפולטי של מבנה-UF של השירים, בנוסף ל-hash table של הג'נרים. שניהם מאתחלים מערכים בגדלים קבועים. ולכן נקבל סיבוכיות זמן O(1).

virtual ~DSpotify () -

נקרא להורסים של המערכים, שעוברים על איברי המערכים והעצים ומשחררים אותם בסיבוכיות לינארית, כלומר O(n+m).

StatusType addGenre(int genreId) -

מחפשים אם קיים ג'נר בעל המזהה **genreId** במערכת, מחפשים אותו בטבלת הערבול של הג'נרים ב- O(1) בממוצע. אם כן מחזירים FAILURE. אחרת, יוצרים עצם מסוג ג'נר בעל המזהה **genreId** ומכניסים אותו לטבלת הגנ'רים ב- O(1) בממוצע משוערך, כי הטבלה דינמית (ראינו הוכחה לסיבוכיות משוערכת בכיתה).
סה"כ סיבוכיות זמן O(1) ממוצע משוערך.

StatusType addSong(int songId, int genreId) -

נחפש תחילה את הג'נר בעל המזהה **genreId** בטבלת הערבול של הג'נרים ב- O(1) בממוצע. אם לא קיים נחזיר FAILURE. אחרת, נבדוק אם קיים שיר במערכת/במבנה ה-UF בעל המזהה

songId. אם קיים נחזיר FAILRE. אחרת, יוצרים עצם חדש מסוג שיר בעל המזהה **songId** ומכניסים אותו למבנה ה-UF של השירים(כחוליה בודדת תחילה) O(1) בממוצע משוערך, כי הוא מוחזק בעזרת טבלת ערבול דינמית. כעת, בודקים את הגודל של הג'נר בעל המזהה **genreId**, אם יש שירים אחרים שמשויכים לג'נר זה אז עושים איחוד בין השיר החדש לשורש עץ השירים המקורי. ולבסוף נעדכן את גודל הג'נר ב- 1+.
עשינו מספר קבוע של איחודים וחיפוש בעצי UF, בנוסף לחיפוש בטבלת ערבול ולכן סה"כ נקבל, סיבוכיות O(log*n) בממוצע משוערך עם **getSongGenre, mergeGenres, getNumberOfGenreChanges.**

StatusType mergeGenres(int genreId1, int genreId2, int genreId3) -
נחפש תחילה את הג'נרים בעלי המזהים **genreId1, genreId2** בטבלת הערבול של הג'נרים ב- O(1) בממוצע. אם אחד מהם לא קיים נחזיר FAILURE. ניצור ג'נר חדש בעל מזהה **genreId3** בעזרת הפונקציה **addGenre** ב-O(1) ממוצע משוערך. כעת, נבדוק אם הגדלים של שני הג'נרים אפסים כלומר הם ריקים, אז לא צריך לעשות איחוד, ונשאיר את הג'נר החדש בעל עץ שירים ריק, ונחזיר SUCCESS. אחרת, נעשה בין עצי השירים שלהם איחוד, ונעדכן את עץ השירים של הג'נר השלישי להחזיק את השורש של העץ המאוחד, בנוסף את הגודל שלו. ונקרא לפונקציה update_path_from שתעדכן את ערכי השדה r, בסיבוכיות O(log*n) בממוצע משוערך עם **getSongGenre, mergeGenres, getNumberOfGenreChanges.**
סה"כ עשינו מספר קבוע של פעולות חיפוש בטבלת ערובל + קריאה לפונקציה **addGenre** שמתבצעת ב-O(1) ממוצע משוערך + מספר קבוע של בדיקות + איחוד בין שני עצים במבנה UF + קריאה לפונקציה update_path_from. ולכן סה"כ סיבוכיות הזמן היא בסיבוכיות O(log*n) בממוצע משוערך עם **getSongGenre, addSong, getNumberOfGenreChanges.**

output_t<int> getSongGenre(int songId) -
נחפש את השורש בעץ של השיר בעל המזהה **songId** במבנה ה-UF, ב- O(1) בממוצע כדי לחפש את האביר עצמו, ו-O(log*n) במשוערך כדי לעשות פעולה דומה ל-find בה מחזירים את השורש של העץ, אם לא קיים נחזיר FAILURE. אחרת, נחזיר את השדה **genre_id** השמור בשורש, כי שדה זה מעודכן רק לשורש בכל עץ במנה ה-UF.
ביצענו חיפוש במבנה UF, ולכן סיבוכיות הזמן היא O(log*n) בממוצע משוערך עם **addSong, mergeGenres, getNumberOfGenreChanges.**

output_t<int> getNumberOfSongsByGenre(int genreId) -
נחפש את הג'נר בעל המזהה **genreId** בטבלת הערבול של הג'נרים ב- O(1) בממוצע. אם לא קיים נחזיר FAILURE. אחרת, נחזיר את השדה **num_songs** שלו שמחזיק רת מספר השירים השייכים לג'נר זה.
סה"כ ביצענו חיפוש בטבלת ערבול + מספר קבוע של פעולות ולכן O(1) בממוצע.

output_t<int> getNumberOfGenreChanges(int songId) -

קוראים לפונקצית העזר path_to אשר כפי שהוסבר קודם, היא מחזירה את סכום כל השדות r עד השורש שהוא מהווה את מספר שינויי הג'נרים של השיר, בסיבוכיות זמן O(log*n) בממוצע משוערך + מספר קבוע של בדיקות שמתבצעות ב- O(1).

ולכן סה"כ הסיבוכיות היא O(log*n) בממוצע משוערך עם הפונקציות **addSong, mergeGenres, getSongGenre**.

ניתוח סיבוכיות מקום:

★ טבלת ערבול של הג'נרים. O(m).

★ מבנה -UF, בעל מערך שהוא טבלת ערבול, של השירים. O(n).

סה"כ סיבוכיות המקום היא O(n+m).