# תרגיל רטוב 2 חלק יבש – מבני נתונים 1

## מבני הנתונים

אלו הם מבני הנתונים בהם נשתמש למימוש הפיתרון:

**HashTable:**

מבנה זה הוא טבלת ערבול לא אוניברסלית, דינמית המשתמשת ברשימות מקושרות הממומשת כפי שראינו בתרגול.

הטבלה מכילה את המאפיינים הבאים:

- capacity: גודל הטבלה.
- size: מספר האיברים בטבלה.
- table: מערך דינמי של רשימות מקושרות המכיל את איברי הטבלה.

כפי שראינו בתרגול, סיבוכיות הזמן של הפעולות insert ו-delete של הטבלה היא $O(\alpha)$ בממוצע על הקלט באופן משוערך וסיבוכיות הזמן של הפעולה find היא $O(\alpha)$ בממוצע על הקלט.

כמו כן, מבנה זה משתמש במערך דינמי, הגדל פי 2 כאשר המערך 75% מלא. באופן זה בכל עת מתקיים $capacity = O(size)$, לכן מתקיים $\alpha = O(1)$, כלומר סיבוכיות הזמן של הפעולות insert ו-delete של הטבלה היא $O(1)$ בממוצע על הקלט באופן משוערך וסיבוכיות הזמן של הפעולה find היא $O(1)$ בממוצע על הקלט.

בנוסף לפעולות אלו, למבנה הפעולות הבאות:

- **HashTable():** פעולה זו מאתחלת את הטבלה למערך של רשימות מקושרות בגודל 6007 (בחרנו 6007 כיוון שזהו מספר ראשוני). אתחול הטבלה נעשה לגודל קבוע, לכן סיבוכיות הזמן של פעולה זו היא $O(1)$.
- **~HashTable():** פעולה זו עוברת על כל תאי הטבלה ובכל תא משחררת את הזיכרון של צמתי הרשימות המקושרות שהתא מכיל. הטבלה בגודל m והיא מכילה n צמתים, לכן סיבוכיות הזמן של פעולה זו היא $O(n+m)$.

**SongNode**

מחלקה זו מייצגת צומת בעץ הפוך, כחלק ממבנה ה-SongUnionFind עליו יפורט בהמשך.

כל צומת מכילה את המאפיינים הבאים:

- songId: מזהה השיר.
- parent: מצביע לצומת ההורה של השיר.
- genreChanges: ערך מאפיין זה נקבע באופן הבא:
  - אם הצומת היא שורש העץ שלה, ערך זה יהיה מספר האיחודים שהקבוצה שצומת זו שייכת אליה עברה.
  - אחרת, ערך זה יהיה מספר האיחודים שהז'אנר אליו השיר שייך עבר לפני שהשיר נוסף לז'אנר. ערך זה מחושב באופן הבא

$$node.parent.genreChanges - node.genreChanges$$

כאשר $node.genreChanges$ הוא ערך ה-$genreChanges$ שהיה לצומת כאשר היא צורפה לז'אנר.

- childrenCount: גודל העץ שצומת זו היא השורש שלו. עבור צמתים שאינם שורשים ערך זה הוא 0.
- genreId: מזהה הז'אנר אליו השירים בעץ שייכים.

**Genre**

מחלקה זו מייצגת ז'אנרה.

למחלקה זו המאפיינים הבאים

- genreId: מזהה הז'אנרה.
- songCount: כמות השירים השייכים לז'אנרה.
- leadingSongUFIdx: אינדקס השיר המייצג את הז'אנר (השיר השייך לז'אנר שהצומת שלו היא שורש העץ ההפוך ב-SongUnionFind ) במערך השירים של SongUnionFind.

**SongUnionFind**

מבנה זה הוא מבנה Union Find אשר צמתיו הם השירים במערכת, וכל קבוצת שירים מייצגת ז'אנר. עבור כל קבוצה נתייחס לשיר השייך לצומת השורש של העץ ההפוך של הקבוצה בתור השיר המייצג את הז'אנר.

מבנה זה ממומש באמצעות מערך דינמי, עצים הפוכים וכיווץ מסלולים כפי שנלמד בהרצאות ובתרגולים.

למבנה זה המאפיינים הבאים

- capacity: כמות השירים שהמבנה יכול להכיל.
- size: כמות השירים במבנה.
- songs: מערך דינמי של SongNode. מערך זה ממומש באופן שבו ראינו בתרגול, כך שסיבוכיות הזמן המשוערכת של פעולת שינוי הגודל שלו היא $O(1)$.

למבנה זה הפעולות הבאות:

- **SongUnionFind():** פעולה זו מאתחלת את המבנה ע"י הקצאה של מערך של אובייקטים מסוג SongNode בגודל 10. כיוון שגודל המערך ההתחלתי קבוע, סיבוכיות הזמן של פעולה זו היא $O(1)$.
- **~SongUnionFind():** פעולה זו משחררת את המערך songs. גודל המערך songs הוא n, לכן סיבוכיות הזמן של פעולה זו היא $O(n)$.
- **addSong(int songId, int genreId):** פעולה זו בודקת האם המערך songs מלא (ע"י השוואת size ל-capacity), ובמידה וכן מקצה מערך חדש בגודל כפול ומעתיקה אליו את הצמתים מהמערך הקיים. לאחר מכן הפעולה יוצרת אובייקט SongNode חדש ומוסיפה אותו ל-songs. ערך ה-gnereId של האובייקט החדש מאותחל לערך ה-genreId שהועבר כקלט וערך ה-parent שלו מאותחל לערך songId. לבסוף הפעולה מגדילה את הערך של size באחד.
<u>סיבוכיות זמן:</u> אנו מגדילים את songs פי 2 רק כאשר הוא מתמלא, לכן סיבוכיות הזמן של יצירת מערך חדש והעתקת האיברים אליו נעשית ב-$O(1)$ באופן משוערך. שאר הפעולות נעשות מתבצעות בסיבוכיות זמן קבועה. לכן סיבוכיות הזמן של פעולה זו היא $O(1)$ באופן משוערך.
- **findLeader(ufIndex):** פעולה זו מקבלת אינדקס של שיר במערך songs ומחזירה את האינדקס של השיר המייצג של הז'אנר אליו השיר שייך, וכן את סכום ערכי ה-genreChanges של צמתי ה-songNode במסלול בין הצומת לשורש (לא כולל הערך של שדה זה עבור השורש). כמו כן הפעולה מבצעת כיווץ מסלולים עבור הצמתים במסלול החיפוש, כאשר לכל צומת נעדכן את

genreChanges להיות סכום ערכי ה-genreChanges של צמתי ה-songNode במסלול בין הצומת לשורש (לא כולל הערך של שדה זה עבור השורש).
<u>סיבוכיות זמן:</u> פעולה זו מבצעת את הפעולה Find של המבנה UnionFind, לכן סיבוכיות הזמן של פעולה זו היא $O(\log^* n)$ באופן משוערך עם unionSongs (כפי שראינו בהרצאות).
<u>סיבוכיות מקום:</u> כפי שראינו בהרצאות, אורך המסלול הארוך ביותר בין צומת לשורש במבנה UF כאשר מבצעים איחוד לפי גודל הוא $\log n$, לכן במקרה הגרוע פעולה זו עוברת על מסלול באורך $\log n$. פעולה זו ממומשת באופן רקורסיבי, לכן סיבוכיות הזמן במקרה הגרוע היא $O(\log n)$

- **unionSongs(uf_idx1, uf_idx2):** פעולה זה מבצעת איחוד בין הקבוצות לפי גודלן.
<u>סיבוכיות זמן:</u> כפי שראינו בהרצאות, סיבוכיות הזמן של פעולת Union עבור מבנה UnionFind הממומש באמצעות איחוד לפי גודל וכיווץ מסלולים היא $O(\log^* n)$ באופן משוערך עם הפעולה Find (findLeader עבור המבנה שלנו).

**DSpotify**

זהו המבנה הראשי של המערכת.

למבנה המאפיינים הבאים:

- songsUF: SongUnionFind המכיל את קבוצות השירים במערכת בקבוצות, כאשר שירים באותה קבוצה שייכים לאותו הז'אנר.
- songsTable: HashTable בה המפתחות הם מזהי שירים והערכים הם האינדקסים המתאימים לשיר במערך songs של songsUF.
- genresTable: HashTable בה המפתחות הם מזהי ז'אנר והערכים הם אובייקט הז'אנר המתאים.
- leaderToGenre: HashTable בה המפתחות הם מזהי שירים של שירים המייצגים זקאנר ב-songsUF והערכים הם מזהה הז'אנר של הז'אנר אליו השיר שייך.

<u>צריכת מקום</u>

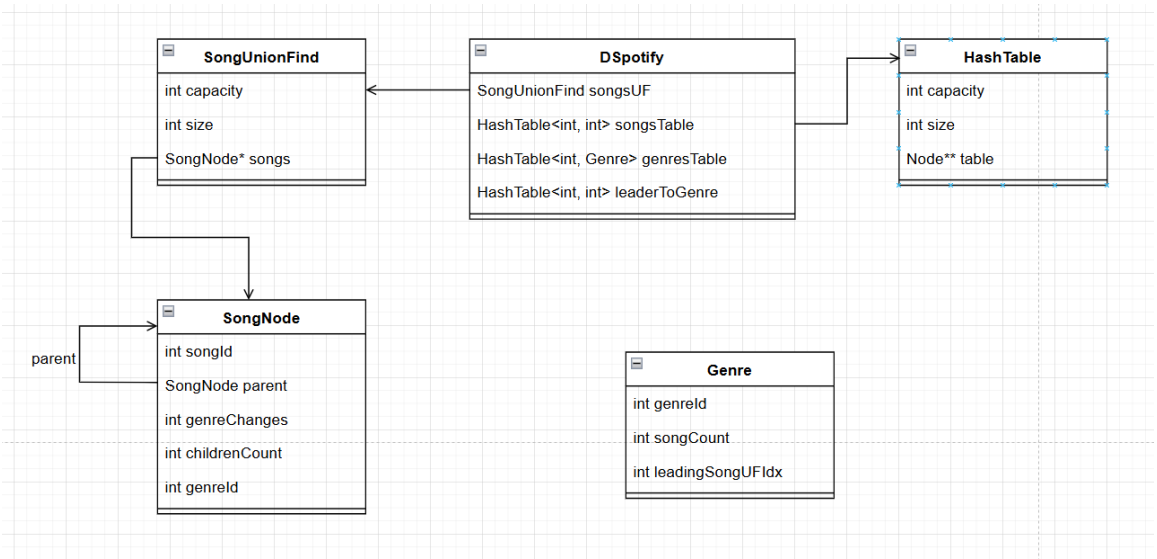
בכל רגע בזמן הריצה המבנים בהם DSpotify משתמש צורכים מקום באופן הבא:

- songsUF מורכב ממערך בגודל 2n לכל היותר.
- songsTable מורכב מטבלה בגודל 2n לכל היותר.
- genresTable מורכב מטבלה בגודל 2m לכל היותר.
- leaderToGenre מורכב מטבלה בגודל 2n לכל היותר.

לכן סך הכל סיבוכיות המקום של מבנה הנתונים שתיארנו היא

$$O(2n + 2n + 2n + 2m) = O(6n + 2m) = O(n + m)$$

**שרטוט מבני הנתונים**




## פעולות עזר

נשתמש בפעולות העזר הבאות:

**validateLeader(SongUnionFind &uf)**

פעולה זו בודקת האם leadingSongUFIdx הוא אכן המזהה של השיר המייצג של הז'אנרה ב-uf ומחזירה את genreId אם כן, אחרת הפעולה מחזירה 1-.
הבדיקה נעשית ע"י השוואת leadingSongUFIdx לערך ההחזרה של uf.getLeader(genreId).
<u>סיבוכיות זמן:</u> SongUnionFind מממומשת באמצעות איחוד לפי גדלים וכיווץ מסלולים, לכן סיבוכיות הזמן של פעולה זו היא

$$O(\log^* n)$$

באופן משוערך.
<u>סיבוכיות מקום:</u> במקרה הגרוע uf.getLeader עוברת על מסלול באורך $\log n$ מצומת לשורש העץ אליה היא שייכת, לכן סיבוכיות הזמן של הפעולה במקרה הגרוע היא

$$O(\log n)$$

## מימוש הפעולות

**DSpotify_t()**

<u>אופן מימוש:</u> נאתחל את songUF כפי שמאתחלים מבנה Union Find ריק ואת songsTable, genresTable ו-leaderToGenre כפי שמאתחלים טבלאות ערבול ריקות.

<u>סיבוכיות זמן:</u> כפי שתיארנו בהסבר על מבני הנתונים, כל האתחולים הנעשים בפעולה זו נעשים בסיבוכיות זמן $O(1)$, לכן סיבוכיות הזמן של פעולה זו היא

$$O(1+1+1+1) = O(4) = O(1)$$

<u>סיבוכיות מקום:</u> פעולה זו מקצה גודל קבוע של זיכרון עבור מבני הנתונים של DSpotify_t, לכן סיבוכיות המקום של פעולה זו היא

$$O(1)$$

**virtual ~DSpotify_t()**

אופן מימוש: נשחרר את הזיכרון שהוקצה למבנים SongUF, songsTable, genresTable ו-leaderToGenre באמצעות ההורסים של מבנים אלו.

סיבוכיות זמן: כי שפורט בתיאור מבני הנתונים, סיבוכיות הזמן של פעולה זו היא

$$O(n+m+n+m+n+m+n) = O(4n+3m) = O(n+m)$$

סיבוכיות מקום: פעולה זו משחררת זיכרון של מערכים דינמיים באופן איטרטיבי, לכן סיבוכיות המקום של פעולה זו היא

$$O(1)$$

**StatusType addGenre(int genreId)**

אופן מימוש: נחפש את הז'אנר ב-genresTable. אם השיר אינו קיים בטבלה ניצור אובייקט Genre חדש עם ה-genreId שקיבלנו בקלט ונוסיף רשומה מתאימה ל-genresTable .

סיבוכיות זמן: כפי שראינו בפירוט של HashTable, סיבוכיות הזמן של הפעולה insert של הטבלה היא $O(1)$ בממוצע על הקלט באופן משוערך, וסיבוכיות הזמן של הפעולה find היא $O(1)$ בממוצע על הקלט, לכן סיבוכיות הזמן של פעולה זו היא $O(1)$ באופן משוערך בממוצע על הקלט.

סיבוכיות מקום: פעולה זו מקצה כמות קבועה של זיכרון עבור אובייקט Genre, לכן סיבוכיות המקום של פעולה זו היא

$$O(1)$$

**StatusType addSong(int songId, int genreId)**

אופן מימוש: נחפש שיר עם המזהה המתאים ב-songsTable וז'אנר עם המזהה המתאים ב-genresTable. אם לא נמצא שיר מתאים ונמצא ז'אנר מתאים, ניצור אובייקט SongNode מתאים ונוסיף אותו ל-songsUF באמצעות songsUF.addSong ונוסיף רשומה לשיר ב-songsTable. לאחר מכן נודא כי הרשומה הרלוונטית ל-gnereId בטבלה leaderToGenre תקינה באמצעות genre.validatedLeader, ונבצע תיקון לרשומות במידה ולא. לבסוף נבדוק האם ערך ה-leadingSongUFidx של הז'אנר השייך ל-genreId שהתקבל אכן שייך לשיר המייצג של הז'אנרה, אם המזהה אינו תקין המשמעות היא שלז'אנר לא היו שירים לפני הפעולה ולכן נוסיף רשומה ב-leaderToGenre בין songId ל-genreId.

סיבוכיות זמן: כפי שראינו בפירוט של HashTable, חיפוש והוספת רשומות ל-songsTable ו-genresTable מתבצעים בסיבוכיות זמן $O(1)$ בממוצע על הקלט. כמו כן, יצירת אובייקט SongNode נעשית בסיבוכיות זמן $O(1)$, וכפי שראינו בפירוט של SongUnionFind סיבוכיות הזמן של addSong היא $O(1)$ באופן משוערך וסיבוכיות הזמן של findLeader היא $O(\log^* n)$ באופן משוערך עם getSongGenre, mergeGenres ו-getNumberOfGenreChanges בממוצע על הקלט. לכן סיבוכיות הזמן של פעולה זו היא

$$O(\log^* n)$$

באופן משוערך עם getSongGenre, mergeGenres ו-getNumberOfGenreChanges בממוצע על הקלט.

סיבוכיות מקום: פעולה זו מקצה כמות קבועה של זיכרון עבור אובייקט SongNode. כמו כן סיבוכיות המקום של validatedLeader במקרה הגרוע היא $\log n$. לכן סיבוכיות המקום של פעולה זו היא

$$O(1+\log n) = O(\log n)$$

במקרה הגרוע.

**StatusType mergeGenres(int genreId1, int genreId2, int genreId3)**

<u>אופן מימוש:</u> תחילה נחפש ז'אנרות בעלות המזהים genreId1, genreId2 ו-genreId3 ב-genresTable. אם ישנן ז'אנרות בעלות המזהים genreId1 ו-genreId2 ולא קיימת ז'אנרה בעלת המזהה genreId3, ניצור ז'אנרה בעלת מזהה זה באמצעות addGenre ונעדכן את שדה ה-songCount של הז'אנרה החדשה להיות הסכום של ערכי ה-songCount של הז'אנרה המתאימה ל-genreId1 ו-genreId2. לאחר מכן נודא כי הרשומות הרלוונטיות ל-gnereId1 ו-genreId2 בטבלה leaderToGenre תקינות באמצעות genre.validatedLeader, ונבצע תיקון לרשומות במידה ולא. לאחר מכן נאחד את הז'אנרות באמצעות songsUF.unionSongs. לאחר מכן נגדיל את מאפיין ה-genreChanges של השיר המייצג החדש ב-1. לבסוף נעדכן את ערך המאפיין genreId של השיר המייצג החדש להיות genreId3, נעדכן את המאפיין leadingSongUFIdx של הז'אנרה החדשה להיות האינדקס של השיר המייצג, נסיר את הרשומות המתאימות לשירים המייצגים הקודמים ב-leaderToGenre ונוסיף רשומה חדשה ל-leaderToGenre בין השיר המייצג החדש למזהה gereId3.

<u>סיבוכיות זמן:</u> כפי שראינו בפירוט של HashTable, חיפוש והוספת רשומות ל-leaderToGenre ו-genresTable מתבצעים בסיבוכיות זמן $O(1)$ בממוצע על הקלט. כמו כן, כפי שראינו בפירוט של SongUnionFind סיבוכיות הזמן של unionSongs היא $O(1)$ היא $O(\log^* n)$ באופן משוערך עם getSongGenre, addSong ו-getNumberOfGenreChanges בממוצע על הקלט. שאר הפעולות מתבצעות מספר קבוע של פעמים. לכן סיבוכיות הזמן של פעולה זו היא

$$O(\log^* n)$$

באופן משוערך עם getSongGenre, addSong ו-getNumberOfGenreChanges בממוצע על הקלט.

$$O(\log^* n)$$

<u>סיבוכיות מקום:</u> פעולה זו מקצה כמות קבועה של זיכרון עבור אובייקט Genre. כמו כן סיבוכיות המקום של validatedLeader במקרה הגרוע היא $\log n$. לכן סיבוכיות המקום של פעולה זו במקרה הגרוע היא

$$O(1 + \log n) = O(\log n)$$

**output_t<int> getSongGenre(int songId)**

<u>אופן מימוש:</u> נחפש שיר בעל מזהה מתאים ב-songsTable. אם נמצא שיר כזה, נקבל את השיר המייצג בקבוצה אליה השיר שייך ב-songsUF באמצעות $songsUF.findLeader(songId)$, ונחזיר את ערך ה-genreId של שיר זה.

<u>סיבוכיות זמן:</u> כפי שראינו בתיאור של המבנה HashTable, סיבוכיות הזמן של חיפוש ב-HashTable היא $O(1)$ בממוצע על הקלט. כמו כן ראינו כי סיבוכיות הזמן של findLeader של songsUf היא $O(\log^* n)$ באופן משוערך עם addSong, mergeGenres ו-getNumberOfGenreChanges. לכן סיבוכיות הזמן של הפעולה היא

$$O(1 + \log^* n) = O(\log^* n)$$

באופן משוערך עם addSong, mergeGenres ו-getNumberOfGenreChanges בממוצע על הקלט.

<u>סיבוכיות מקום:</u> במקרה הגרוע הפעולה findLeader עוברת על המסלול מצומת השיר לשורש העץ פעמיים, פעם אחת למציאת השורש ופעם שנייה לכיווץ המסלולים. כפי שראינו בהרצאה, האורך

המקסימלי של מסלול כזה הוא $\log n$. הפעולה מבצעת את מעברים אלו באופן רקורסיבי, לכן במקרה הגרוע סיבוכיות המקום של פעולה זו היא

$$O(2\log n) = O(\log n)$$

**output_t<int> getNumberOfSongsByGenre(int genreId)**

<u>אופן מימוש:</u> נחפש ז'אנר עם מזהה מתאים ב-genresTable. אם מצאנו, נחזיר את genre.songsCount.

<u>סיבוכיות זמן:</u> כפי שראינו בתיאור של המבנה HashTable, סיבוכיות הזמן של חיפוש ב-HashTable היא $O(1)$ בממוצע על הקלט. לכן סיבוכיות הזמן של הפעולה היא

$$O(1)$$

בממוצע על הקלט.

<u>סיבוכיות מקום:</u> פעולה זו אינה מקצה זיכרון, לכן סיבוכיות המקום של פעולה זו היא

$$O(1)$$

**output_t<int> getNumberOfGenreChanges(int songId)**

<u>אופן מימוש:</u> תחילה נבדוק האם ישנו שיר בטבלה songsTable בעל המזהה songId. במידה וכן, נאתחל משתנה בשם delta לערך 0 ונעבור על כל צומת מצומת השיר עד לשורש העץ ההפוך אליה הצומת שייך. בכל צומת במסלול שאינה השורש, נוסיף ל-delta את ערך ה-genreChanges של הצומת הנוכחית. לבסוף נחזיר את הערך

$$root.genreChanges - delta + 1$$

<u>נכונות ערך ההחזרה:</u>

נשים לב כי $root.genreChanges$ הוא מספר האיחודים שהקבוצה עברה.

כמו כן, כפי שראינו בפירוט של SongNode, לכל צומת שאינה שורש במסלול בין צומת השיר לצומת שורש בעץ ההפוך אליה הצומת שייכת, הערך genreChanges הוא מספר האיחודים שהז'אנר אליו השיר שייך עבר לפני שהשיר נוסף לז'אנר, כלומר delta הוא מספר האיחודים הכולל שהז'אנר אליו השיר בעל המזהה songId שייך עבר לפני שהשיר בעל המזהה songId נוסף לז'אנר.

לכן הערך $root.genreChanges - delta$ הוא מספר האיחודים שהז'אנר עבר אחרי שהשיר נוסף לז'אנר, כלומר מספר הפעמים שהתחלף לשיר ז'אנר מאז שהוא נוסף לקבוצה לראשונה. כמו כן יש לספור את הוספת השיר לז'אנר לראשונה, שכן זהו השינוי הראשון של הז'אנר של השיר. לכן מספר הז'אנרות השונות שהיו לשיר הוא $root.genreChanges - delta + 1$.

<u>סיבוכיות זמן:</u> כפי שראינו בפירוט על HashTable, חיפוש ב-songsTable נעשה בסיבוכיות זמן $O(1)$ באופן משוערך בממוצע על הקלט. כמו כן, הפעולה מבצעת findLeader לצומת של השיר בעל המזהה songId. כפי שראינו בפירוט של SongUnionFind סיבוכיות הזמן של פעולה זו היא $O(\log^* n)$ באופן משוערך עם פעולת ה-union של SongUnionFind, לכן סיבוכיות הזמן של הפעולה היא

$$O(\log^* n)$$

באופן משוערך עם getSongGenre, mergeGenres ו-addSong בממוצע על הקלט.

<u>סיבוכיות מקום:</u> המעבר מהצומת של השיר בעל המזהה songId נעשית באופן רקורסיבי. במקרה הגרוע אורך מסלול זה הוא $\log n$ לכן סיבוכיות המקום של פעולה זו במקרה הגרוע היא

$$O(\log n)$$

**<u>סיבוכיות המקום של המבנה</u> DSpotify_t**

כפי שראינו, סיבוכיות המקום של המבנה DSpotify_t היא $O(n+m)$. כמו כן סיבוכיות המקום במקרה הגרוע של כל פעולה של המבנה היא $O(1)$, לכן סיבוכיות המקום של המבנה DSpotify_t במקרה הגרוע היא

$$O(n+m+\log n) = O(n+m)$$