חלק יבש מבני נתונים רטוב 1

# תיאור מבנה הנתונים

מבני הנתונים מורכב מ:

1. עץ AVL המכיל את כל הסרטים וממוין לפי מפתח \_id. הנתונים שכל סרט מכיל הם:

* \_id ( int ) – המזהה של הסרט לפיו העץ ממוין.
* totalRating ( double ) – הדירוג הכולל של הסרט.
* ratingsCount( int )\_ – מספר הדירוגים שהסרט קיבל עד עכשיו.
* \_views( int ) – מספר הצפיות שהסרט קיבל עד עכשיו.
* \_vip( bool ) – האם הסרט VIP
* \_genre (Genre) – הז'אנר של הסרט.

1. עץ AVL המכיל את כל המשתמשים וממוין לפי מפתח \_id. הנתונים שכל משתמש מכיל הם:

* Id (int)\_ – מזהה משתמש לפיו ניתן למיין את העץ
* \_vip( bool ) – האם המשתמש VIP
* \_group - מצביע לקבוצה בה המשתמש נמצא
* \_group\_views\_before\_join – מערך של צפיות בקבוצה לפני שהמשתמש הצטרף לפי ז'אנר
* \_views - מערך של צפיות לפי זיאנר
* next\_group\_member – מצביע למשתמש הבא ברשימה המקושרת של משתמשים ששייכים לקבוצה
* prev\_group\_member – מצביע למשתמש הקודם ברשימה המקושרת של משתמשים ששייכים לקבוצה

1. עץ AVL המכיל את כל הקבוצות וממוין לפי מפתח \_id. הנתונים שכל קבוצה מכילה:

* \_id (int) – מזהה קבוצה לפיו ניתן למיין את העץ.
* ( int ) \_vip\_users– מספר החברים בקבוצה שהם VIP.
* users\_num\_ - מספר החברים בקבוצה
* \_first\_user\_in\_group\_list – מצביע למשתמש שהוא האיבר הראשון ברשימה המקושרת של הקבוצה
* \_group\_views – מערך של מספר הפעמים שהקבוצה צפתה לפי ז'אנר.
* \_users\_views – מערך של מספר הצפיות הכולל של חברי הקבוצה שלא באופן קבוצתי לפי ז'אנר.
* \_total\_views - מערך של מספר הצפיות הכולל של חברי הקבוצה הנוכחיים במסגרת הקבוצה.

1. מערך של עצי AVL לכול ז'אנר המורכב ממפתח המכיל פוינטר לסרט. המיון נעשה לפי השדות:
   * \_id של הסרט
   * Rating המחושב מחלוקת totalRating ב-\_ratingsCount
   * ֹ\_views

ראה תיאור הפונקציה get\_group\_recommendation לפרטים נוספים.

## הרחבות לעץ AVL

* **איבר מקסימלי:** לאלגוריתם של עץ ה-AVL הוספנו שמירה של האיבר המקסימלי בעץ. נעדכן את את האיבר המקסימלי בעץ בעלות של O(h) כש-h הוא גובה העץ. האלגוריתם הוא ביצוע פניות ימינה משורש העץ עד להגעה לעלה שהוא בעל הערץ המקסימלי. מפני שראינו בהרצאה שבעץ AVL, h הוא O(log(size)) כש-size גודל העץ, נקבל כי ניתן לעדכן את האיבר המקסימלי בהוצאה והכנסה לעץ AVL מבלי לפגוע בסיבוכיות O(log(size)).
* **Inorder הפוך**: בנוסף, על מנת לקבל מיון בסדר יורד, הוספנו סידור inorder הפוך לעץ הAVL. כלומר סידור inorder בו עוברים קודם על תת העץ הימני, לאחר מכן על השורש ולבסוף על תת העץ השמאלי. ניתן לעשות סידור זה ב O(log(size)) כש-size גודל העץ, לפי אלגוריתם ה- inorder הרגיל שהוצג בכיתה. יש רק להחליף בין תת העץ הימני לשמאלי, מטעמי סימטרית שמאל ימין, לאלגוריתם זה סיבוכיות של אלגוריתם הinorder- הרגיל. באותו אופן בו אלגוריתם הinorder הרגיל מהתרגול מציג את איברי העץ כך שהם ממוינים לפי המפתחות בסדר עולה, כך סידור הinorder ההפוך יציג את איברי העץ כך שהם ממוינים לפי המפתחות בסדר יורד.

# תיאור אלגוריתמים :

נסמן: – הגודל של מספר הסרטים בז'אנר Genre, k – מספר הסרטים במערכת, m – מספר הקבוצות במערכת, n – מספר המשתמשים במערכת.

## אתחול המבנה

המבנה מורכב מעצים וממערך של מספר קבוע עצים. עצים מאותחלים ב.O(1) זאת, כפי שראינו בתרגול ובהרצאה. מערך שתא שלו מאותחל בO(1) מאותחל בO(n) כפי שראינו בתרגול בהרצאה, ולכן מערך קבוע מאותחל ב O(1). לכן, בסה"כ, הסיבוכיות הכוללת היא O(1).

## פונקציות Add ו-Remove

את הפונקציות add\_user ו - remove\_user ניתן לעשות ב- O(log(n)) לפי אלגוריתם של הוספה והסרה מעץ AVL הנלמד בכיתה. הוכח בכיתה כי ניתן להכניס ולהוציא מעץ AVL בסיבוכיות log(size) כש- size הוא גודל העץ. אם המשתמש שייך לקבוצה, ניתן להסיר אותו ממנה על ידי אלגוריתם של הוצאה של איבר מרשימה מקושרת דו כיוונית שהוכח כי הוא עובד ב .O(1)-בעת ההוצאה של המשתמש, נעדכן את השדות הרלוונטיים בקבוצה.

באותו אופן, את הפונקציות add\_movie ו remove\_movie ניתן לעשות בO(log(k)).

את הפונקציה add\_group ניתן לבצע בO(log(k)), מפני שהפונקציה מכניסה את הקבוצה לעץ של הקבוצות בסיבוכיות O(log(k)) ויוצרת רשימה מקושרת של משתמשים בסיבוכיות O(1).

סיבוכיות הפונקציה remove\_group היא O(log(k) + ), כש הוא מספר המשתמשים בקבוצה. זאת מפני שיש להוציא את הקבוצה מהעץ בסיבוכיות .log(k) בעת ההוצאה, יש לעבור על הרשימה המקושרת של המשתמשים ולהוציא אותם מרשימת הקבוצה על ידי אלגוריתם של מעבר על רשימה מקושרת . הוכח בהרצאה שהסיבוכיות של אלגוריתם זה היא log(size) כש- sizeהוא גודל הרשימה. בעת המעבר על הרשימה, נעדכן את מספר הצפיות של כול משתמש בכול זאנר במספר הצפיות שהוא צפה עם הקבוצה. מספר הצפיות של המשתמש עם הקבוצה בז'אנר i הוא i]]ן\_group\_views[i] - \_group\_views\_before\_join .

בתור הערה, אגיד שהפכתי את המשתמש עצמו לחוליה ברשימה מקושרת דו כיוונית על ידי שמירת פוינטרים למשתמש הבא והמשתמש הקודם. לכן, לא מעורבת ברשימה המקושרת הקצאת זיכרון שצריך לשחרר.

## הפונקציה get\_group\_recommendation

עבור הפונקציה get\_group\_recommendation:

* נחפש את את הקבוצה על ידי אלגוריתם חיפוש בעץ AVL בlog(k).
* לאחר מכן, נקבל את ה-Genre האהוב עליה ב-O(1). מכיון שמספר הג'אנרים הוא קבוע, ניתן ב-O(1) לקבל את מספר הצפיות בכול ז'אנר ומספר הז'אנרים סופי. מספר הצפיות בז'אנר i הוא \_total\_views[i] + \_ users\_views[i] .
* נקבל את הסרט בעל הדירוג המקסימלי בז'אנר האהוב על חברי הקבוצה באמצעות האיבר המקסימלי בעץ. ניקח את האיבר המקסימלי בעץ AVL של הז'אנר המתאים בO(1).
* סה"כ קיבלנו סיבוכיות של O(1).

## הפוקציה get\_all\_movies

עבור הפונקציה get\_all\_movies לז'אנר מסוים, נעתיק המעץ המתאים למערך שיהיה ממוין נכון בשל המעבר על העץ בשיטת inorder הפוך. ניתן לעשות זאת זה בסיבוכיות O(), הגודל של מספר הסרטים בז'אנר, כפי שהראיתי בתיאור התוספות inroder הפוך שהוספנו לאלגוריתם הAVL.

אם Genre הוא NONE, נעתיק בדומה, לתוך מערך ייעודי עבור כל ז'אנר. נעשה זאת בסיבוכיות של O(k), כשK הוא מספר הסרטים הכולל. לאחר כך, נמזג את חמשת המערכים הממוינים שנוצרו למערך ממוין אחד. נבצע זאת ב O(k), מפני שמספר המערכים סופי (לפי מספר הז'אנרים) ולכן ניתן למזגם בסיבוכיות של סכום גדלי המערכים. האלגוריתם הוא כדלקמן:

* ניצור מערך חדש בגודל k (סכום גדלי המערכים) .
* נחזיק אינדקס לאיבר הנוכחי בכול אחד מהמערכים.
* בכול פעם נעתיק למערך הגדול את האיבר המקסימלי מבין האיברים עליהם מצביעים האינדקסים הנוכחיים שלא חרגו מגבולות המערך שלהם (אם יש כאלה).
* נקדם את האינדקס המצביע על איבר זה באחד.
* נסיים כשאין אינדקסים שלא חורגים מגבולות המערך.
* בסיום התהליך עברנו פעם אחד על כול אחד מהמערכים, ולכן הסיבוכיות היא של סכום גדלי המערכים .

הסיבוכיות הכוללת במקרה זה היא O(k).

## הפונקציה get\_num\_views

עבור הפונקציה get\_num\_views, נמצא את המשתמש ב O(log(K)) כש-K מספר המשתמשים במערכת. לאחר מכן, נחבר את מספר הצפיות שלו לבד למספר הצפיות שלו עם הקבוצה. מכיוון שכול משתמש מחזיק פוינטר לקבוצה שלו, ניתן לקבל את מספר הפעמים שהקבוצה צפתה בז'אנר (לא מספר הצפיות הכולל) דרך הפוינטר בO(1). סה"כ הסיבוכיות היא O(k).

## הפונקציה add\_user\_to\_group

את הפונקציה add\_user\_to\_group ניתן לבצע ב O(log(n) + log(m)):

* ראשית, נמצא את הקבוצה ואת המשתמש לפי אלגוריתם לחיפוש של קבוצה בעץ AVL בO(log(n) + log(m)).
* לאחר מכן, נוסיף את המשתמש לרשימת הקבוצה לפי אלגוריתם להוספה של איבר בראש רשימה מקושרת בO(1)-. כמו כן, נעדכן את השדות \_users\_views בקבוצה ו \_group\_views\_before\_join במשתמש. נעדכן את מספר המשתמשים ומשתמשי הvip בקבוצה.

## הפונקציה get\_all\_movies\_count

את הפונקציה get\_all\_movies\_count ניתן לבצע בO(1) על ידי שמירה בכל עץ את הגודל שלו. הגודל מתעדכן בהוספה ומחיקה של איברים מהעץ בO(1).

## הפונקציה rate\_movie

* עבור הפונקציה rate\_movie, נחפש את המשתמש בעץ AVL של המשתמשים, ואת הסרט בעץ AVL של הסרטים. ניתן לעשות זאת ב O(log(n) + log(k))במקרה הגרוע, כאשר n הוא מספר המשתמשים ו k הוא מספר הסרטים במערכת .זאת לפי סיבוכיות של אלגוריתם חיפוש בעץ AVL הנלמד בכיתה.
* במידה וה-Id של המשתמש וה-Id של הסרט לא תקינים לא נמשיך בעדכון.
* לאחר מכן, על מנת לשמור על המיון של עץ הז'אנר, נחפש את הסרט בעץ של הז'אנר המתאים. נמחק אותו, ונוסיף אותו מחדש לעץ לאחר שהrating השתנה. ניתן לבצע את החיפוש ההוצאה וההכנסה של הסרט מהעץ של הזיאנר שלו בO(Log(), כלומר O(Log(k)) כש- מספר הסרטים בעץ של הז'אנר של הסרט.
* סה"כ הסיבוכיות היא O(log(n) + log(k)), מפני שביצענו מספר סופי של פעולות שהן O(log(n) + log(k)).

## הפונקציה user\_watch

עבור הפונקציה user\_watch:

* נחפש את המשתמש בעץ AVL של המשתמשים ואת הסרט בעץ AVL של הסרטים. ניתן לעשות זאת ב O(log(n) + log(k))במקרה הגרוע, כאשר n הוא מספר המשתמשים ו k הוא מספר הסרטים במערכת .זאת לפי סיבוכיות של אלגוריתם חיפוש בעץ AVL הנלמד בכיתה.
* במידה וה-Id של המשתמש וה-Id של הסרט לא תקינים, או שאין התאמה בין סטטוס vip של הסרט והמשתמש, לא נמשיך בעדכון.
* לאחר מכן, על מנת לשמור על המיון של עץ הז'אנר, נחפש את הסרט בעץ של הז'אנר המתאים, נמחק אותו, ונוסיף אותו מחדש לעץ לאחר שמספר הצפיות השתנה בסרט ובמשתמש. ניתן לבצע את החיפוש ההוצאה וההכנסה של הסרט מהעץ של הזיאנר שלו בO(Log(), כלומר O(Log(k)) כש- מספר הסרטים בעץ של הז'אנר של הסרט.
* בנוסף, נעדכן את צפיות המשתמשים בקבוצה בז'אנר המתאים בO(1)- בעזרת המצביע לקבוצה שבמשתמש מחזיק.

סה"כ עמדנו בסיבוכיות O(log(n) + log(k)).

## הפונקציה group\_watch

עבור הפונקציה group\_watch:

* נחפש את הקבוצה בעץ AVL של הקבוצות, ואת הסרט בעץ AVLשל הסרטים. ניתן לעשות זאת ב O(log(m) + log(k))במקרה הגרוע .זאת לפי סיבוכיות של אלגוריתם חיפוש בעץ AVL הנלמד בכיתה.
* במידה וה-Id של המשתמש וה-Id של הסרט לא תקינים, או שאין התאמה בין סטטוס vip של הסרט והקבוצה, לא נמשיך בעדכון.
* לאחר מכן, על מנת לשמור על המיון של עץ הז'אנר, נחפש את הסרט בעץ של הז'אנר המתאים, נמחק אותו, ונוסיף אותו מחדש לעץ לאחר שמספר הצפיות השתנה בסרט. ניתן לבצע את החיפוש ההוצאה וההכנסה של הסרט מהעץ של הזיאנר שלו בO(Log(), כלומר O(Log(k)) כש- מספר הסרטים בעץ של הז'אנר של הסרט.
* נגדיל לקבוצה את השדה \_group\_views בכול ז'אנר באחד ואת השדה \_total\_views בכול ז'אנר במספר המשתמשים בקבוצה.

סה"כ עמדנו בסיבוכיות O(log(n) + log(k)).