שמות הסטודנטים:

* זאהר עאמר 323910828
* אתיר אבראהים 213281470

**תיאור המבנה:**

המבנה שלנו מכיל את מבני הנתונים הבאים:

1. teamsIdTree: זהו עץ דרגות AVL שממוין לפי המזהה הייחודי של הקבוצות, כל צומת בעץ היא ממחלקת TeamById שמכילה:

* playersNum: מספר השחקנים שנמצאים בקבוצה.
* teamId: המזהה הייחודי של הקבוצה.
* allPlayersAbility: סכום היכולות של כל השחקנים הנמצאים בקבוצה.
* hasGk: משתנה בוליאני המעיד על כך אם יש בקבוצה שוער או לא.
* Captain: מסוג Player, הוא הקפטן של הקבוצה כלומר השחקן הכי ותיק.

1. teamsStatsTree: זהו עץ דרגות AVL שממוין לפי סכום היכולות של כל השחקנים בקבוצה, אם הם שווים אז לפי המזהה הייחודי של הקבוצה, כל צומת בעץ היא ממחלקת TeamByStats שמכילה:

* allPlayersAbility: סכום היכולות של כל השחקנים הנמצאים בקבוצה.
* TeamId: המזהה הייחודי של הקבוצה.

1. playersTable: זאת טבלת ערבול דינמית עם רשימה מקושרת

(chained Hash Table) שכל תא בה הוא מצביע ל UFNode שהוא:

* UFNode הוא וכמו שהשם מעיד, צומת בעץ הפוך שמשומש למבנה הנתונים Union Find, בכל צומת יש:

o- Key: המפתח שמשומש למציאת הצומת.

o- data: המידע של הצומת שהוא מסוג Player.

o- gamesRemainder: מספר המשמש למציאת מספר המשחקים ששיחק

שחקן מסויים.

o- spiritRemainder: פרמוטציה המשמשת למציאת partial\_spirit.

o- teamSpirit: פרמוטציית הרוח של הקבוצה שלה שייך השחקן.

o- partialSpirit: פרמוטציה המשמשת למציאת partial\_spirit.

**סיבוכיות המקום:**

* שמרנו עץ אשר מכיל מידע על הקבוצות  **teamsIdTree**שמכיל לכל היותר k צמתים (כאשר k הוא מספר הקבוצות הכולל במערכת) כאשר בכל צומת אננו שומרים O(1) מקום, וסה"כ עבור העץ נקבל O(k) מקום.
* שמרנו עץ אשר מכיל מידע על הקבוצות  **teamsStatsTree**שמכיל לכל היותר k צמתים (כאשר k הוא מספר הקבוצות הכולל במערכת) כאשר בכל צומת אננו שומרים O(1) מקום, וסה"כ עבור העץ נקבל O(k) מקום.
* שמרנו טבלת ערבול לכל השחקנים במערכת. המימוש הינו מערך דינמי שיעלה O(n) סביכיות מקום העובד לפי שיטת ה- chain hashing למניעת התנגשויות דרך רשימה מקושרת שעולה O(1) בממוצע סביכיות מקום.

לכן בסה"כ יעלה O(n) סביכיות מקום בלבד.

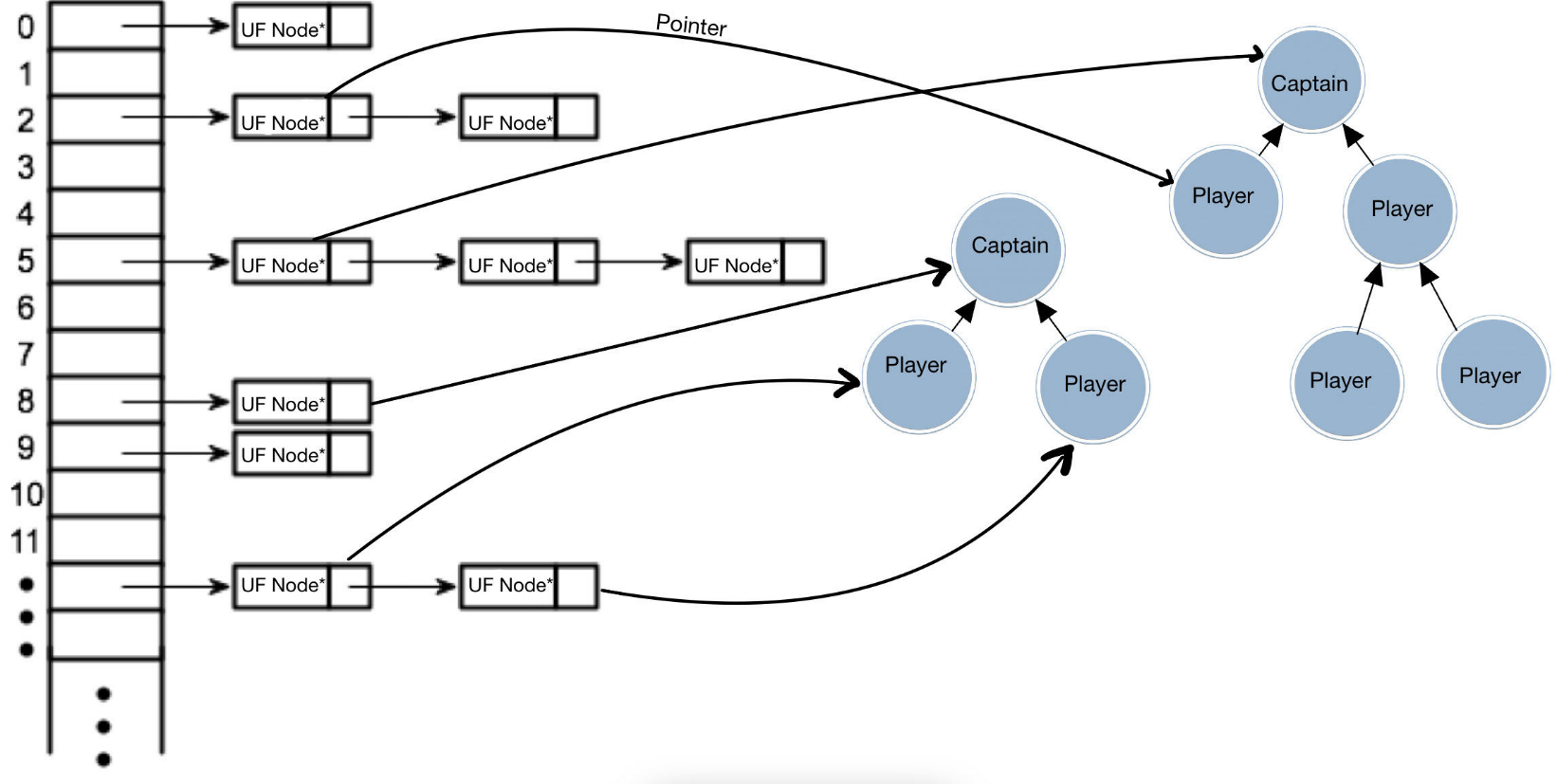
* שמרנו עצים הפוכים למימוש מבנה הנתונים Union Find, כאשר שמרנו לכל היותר n צמתים (כאשר n הוא מספר השחקנים הכולל במערכת), כלומר סה"כ O(n).

לכן סה"כ מקבלים שסיבוכיות המקום היא: כנדרש.

נצרף איור למערכת שלנו ולמבני הנתונים:

Hash Table

Union Find



Eliminated?

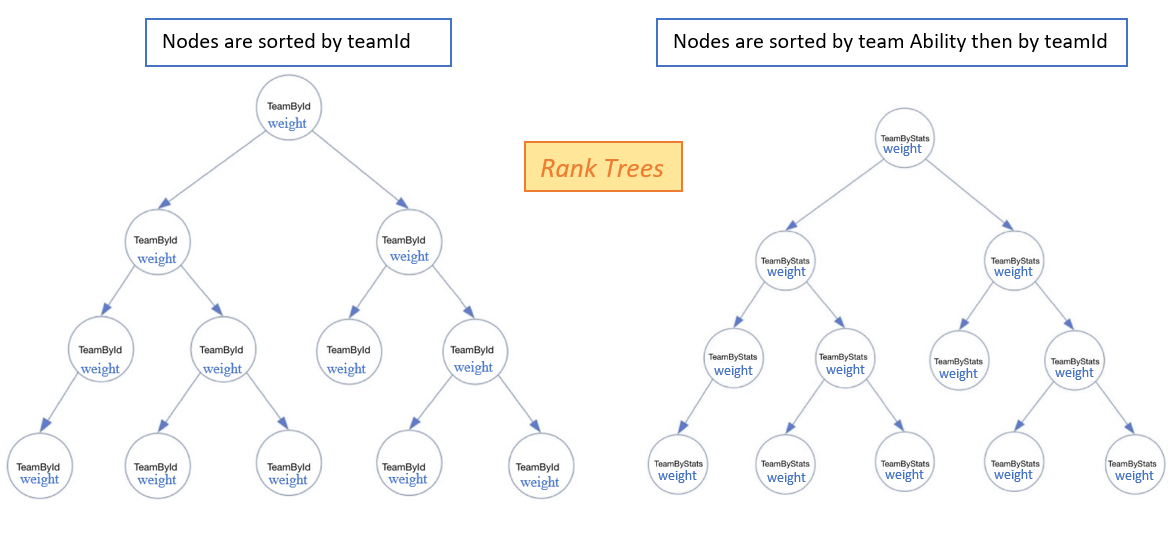
teamGames

teamId

Eliminated?

teamGames

teamId



**תיאור הפונקציות והוכחת סיבוכיות:**

(נציין שלאורך הוכחת הסיבוכיות עבור הפונקציות- k הוא מספר הקבוצות הכולל במערכת ו- n הוא מספר השחקנים הכולל במערכת).

StatusType add\_team(int teamId):

* בהתחלה בודקים אם הקלט תקים, O(1) זמן.
* מחפשים את הקבוצה בעץ הקבוצות לפי ID בסיבוכיות זמן , אם מוצאים מחזירים FAILURE.
* מכניסים את הקבוצה לשני עצי ה AVL, עץ ה teamsById ןעץ ה teamByStats, זה לוקח O(2logk).
* מחזירים SUCCESS.

לכן סה"כ לוקח זמן.

:StatusType remove\_team(int teamId)

* בהתחלה בודקים אם הקלט תקין, O(1) זמן.
* מחפשים את הקבוצה בעץ, אם לא מוצאים מחזירים FAILURE, זמן.
* בודקים אם יש שחקנים בקבוצה, אם כן מכבים את הקפטן של הקבוצה בעזרת מציאתו בטבלת הערבול הדינמית על ידי סימון המשתנה Eliminated = true, זה לוקח O(1) זמן בממוצע.
* מוצאים את הקבוצה בעץ הקבוצות לפי הסטטיסטיקות, .
* מוחקים את הקבוצה משני העצים, זה לוקח O(2logk).

לכן סה"כ לוקח O(logk) זמן.

StatusType add\_player(int playerId, int teamId, const permutation\_t &spirit, int gamesPlayed,  
 int ability, int cards, bool goalKeeper)

* בהתחלה בודקים אם הקלט תקין, O(1) זמן.
* מוצאים את הקבוצה של השחקן בעץ הקבוצות, אז בודקים אם השחקן נמצא בטבלת הערבול הדינמית, זה לוקח זמן.
* אם הכל תקין, מכניסים את השחקן לטבלת הערבול הדינמית (גם ל UF), זה לוקח O(1) זמן.
* מעדכנים את ה team Ability של הקבוצה הרלוונטית, מוחקים אותה מעץ הסטטסטיקות ומכניסים מחדש, (כדי שמקומה בעץ יהיה נכון), זה לוקח זמן.
* בודקים אם השחקן הוא הקפטן של הקבוצה, אם לא עושים לו Union עם הקפטן של הקבוצה שמוצאים בטבלת הערבול הדינמית, סך כל O(1) זמן.

לכן סה"כ סיבוכיות הזמן, .

output\_t<int> play\_match(int teamId1, int teamId2):

* בהתחלה בודקים אם הקלט תקין, O(1) זמן.
* מוצאים את שתי הקבוצות בעץ הקבוצות, אם לא מוצאים אחת מהן מחזירים FAILURE, O(2logk).
* מוצאים את הקפטנים של הקבוצות בטבלת הערבול, מוסיפים 1 למספר המשחקים, O(1).
* בודקים מי הקבוצה המנצחת לפי הדרישות ומחזרים ערך בהתאם, O(1).

לכן סה"כ סיבוכיות הזמן, O(logk).

output\_t<int> num\_played\_games\_for\_player(int playerId)

* בודקים את תקינות הקלט, O(1).
* מוצאים את השחקן בטבלת הערבול הדינמית, O(1).
* מוצאים את הקפטן של השחקן ב UF, .
* מחזירים את ה gamesPlayed + gamesRemainder + captain.TotalGames, O(1).

לכן סה"כ סיבוכיות הזמן, O(log\*n).

StatusType add\_player\_cards(int playerId, int cards):

* בודקים את תקינות הקלט, O(1).
* מוצאים את השחקן בטבלת הערבול הדינמית, O(1).
* מחפשים את הקפטן של השחקן ב Union Find, אם הוא כבוי (מודח מהתחרות) אז מחזירים FAILURE, .
* אם הכל תקין, מוסיפים את מספר הכרטיסים לשחקן ומחזירים SUCCESS, O(1).

לכן סה"כ סיבוכיות הזמן, O(log\*n).

output\_t<int> get\_player\_cards(int playerId):

* בודקים את תקינות הקלט, O(1).
* מוצאים את השחקן בטבלת הערבול הדינמית ומחזירים את הכרטיסים שלו, O(1).

לכן סה"כ סיבוכיות הזמן, O(1).

output\_t<int> get\_team\_points(int teamId):

* בודקים את תקינות הקלט, O(1) זמן.
* מוצאים את הקבוצה בעץ הקבוצות לפי המזהה ומחזירים את הנקודות שלה, O(logk).

לכן סה"כ סיבוכיות הזמן, O(logk).

:output\_t<int> get\_ith\_pointless\_ability(int i)

* בודקים את תקינות הקלט, O(1).
* עושים select ב Rank tree כפי שנלמד בהרצאה למספר i בעץ הקבוצות שממוין לפי הסטטיסטיקות, O(logk).

לכן סה"כ סיבוכיות הזמן היא, O(logk).

output\_t<permutation\_t> get\_partial\_spirit(int playerId)

* *בודקים את תקינות הקלט, O(1).*
* *בודקים אם הקפטן של השחקן ב UF כבוי, אם כן מחזירים כישלון, O(log\*n).*
* *אם הכל תקין, מוצאים את השחקן בטבלת הערבול הדינמית, O(1).*
* *מחזירים את ה spiritRemainder של הקפטן כפול spiritRemainder של השחקן כפול partialSpirit של השחקן, O(1).*

*לכן סה"כ O(log\*n) זמן.*

StatusType buy\_team(int buyerId, int boughtId):

* בודקים תקינות קלט, O(1).
* מחפשים את שתי הקבוצות בעץ teamsById, אם לא מוצאים מחזירים Failure, O(2logk).
* מחלקים למקרים, אם הקבוצה השנייה לא מכילה שחקנים, סך כל מוחקים אותה, O(logk).
* אם הקבוצה הראשונה לא מכילה שחקנים, מוחקים אותה, מוצאים את הקבוצה השנייה בעץ הסטטסטיקות, מעדכנים את ה teamAbility, משנים את ה teamId של הקבוצה השנייה ל teamId1 ומוחקים אותה משני העצים ומכניסים מחדש, O(5logk).
* אם שתי הקבוצות מכילות שחקנים, נבדוק אם הקבוצה המאוחדת נהיתה חוקית כלומר יש בה שוער, נעדכן את מספר השחקנים בקבוצה הראשונה למספר כל השחקנים בשתי הקבוצות, נבדוק אם לפני האיחוד מספר השחקנים בקבוצה הראשונה היה גדול ממספר השחקנים בקבוצה השנייה, אם כן, נמצא את הקבוצה בעץ הקבוצות לפי הסטטסטיקות, נעדכן את הקבוצה על ידי מחיקתה והוספת second Team Ability והכנסתה מחדש, בסוף נאחד את הקפטן של הקבוצה השנייה עם הקפטן של הקבוצה הראשונה ב UF, כל אלו עולות O(logk + log\*n), למציאת הקבוצות והקפטנים ב UF.
* אם הקבוצה השנייה גדולה מהקבוצה הראשונה, נעשה את אותם הדברים אך נשנה את הקפטן של הקבוצה הראשונה לקפטן השני וב UF נאחד את הקפטן הראשון לשני ולא להפך (איחוד לפי גודל), זה עולה O(log\*n) למצוא את הקפטנים ב UF.

לכן סה"כ סיבוכיות הזמן: .