**תרגיל רטוב 2 – החלק היבש**

שרה גריפית – ת"ז: 341312304

ליאור בר יוסף – ת"ז: 207022443

**תיאור מבנה הנתונים:**

עבור אפליקצית "מאחורי הקלעים" של פיפ"א, נבנה מבנה נתונים המורכב משני עצי AVL מרכזיים של קבוצות:

* **m\_teamsByID** – כל הקבוצות שמשתתפות בגביע העולם, מסודרים לפי מזהה הקבוצה.
* **m\_teamsByAbility** – כל הקבוצות המשתתפות בגביע העולם, מסודרים לפי היכולת של כל קבוצה (השווה לסכום היכולות של כל השחקנים). חשוב לציין כי זהו עץ AVL שהוא גם עץ דרגות כפי שנלמד בהרצאה.

בגביע העולם יש גם משתנה נוסף הסוכם את כמות השחקנים שמשתתפים בכל גביע העולם (**m**\_**numTotalPlayers**), ועוד משתנה הסוכם את כמות הקבוצות המשתתפות בגביע העולם (**m\_numTeams**).

כמו כן, בגביע העולם יש טבלת ערבול (Hash Table), כך שכל תא בטבלה מכיל מצביע לעץ AVL של שחקני גביע העולם (**m\_playersHashTable**). כלומר, במקום רשימות מקושרות כפתרון להתנגשויות בטבלת הערבול, יש עצי AVL. בהתאם לכך, יש משתנה נוסף במערכת השומר את גודל טבלת הערבול הנוכחי (**m\_currentHashSize).**

כדי לממש את מבנה הנתונים הנ"ל, הוספנו שתי מחלקות: Player ו-Team אשר מכילים את הפרטים של השחקנים ושל הקבוצות כדורגל בגביע העולם.

* **מחלקת Team** תכיל את הפרטים הבסיסיים של אותה קבוצה: מזהה קבוצה, מספר הנקודות של הקבוצה, מספר השחקנים בקבוצה, מספר השוערים בקבוצה, מספר המשחקים שהקבוצה שיחקה, וסך כל היכולות של שחקני הקבוצה. בנוסף, יש לכל קבוצה משתנה מטיפוס permutation\_t השומר את "רוח" הקבוצה (כפל הרוחות של שחקני הקבוצה לפי סדר הוספתם לקבוצה).

כמו כן, יש לכל קבוצה מצביע לשחקן שמשחק בקבוצה. חשוב לציין:

* + כל שחקן מכיל מצביע לשחקן אחר שמשחק איתו באותה קבוצה.
  + השחקנים של אותה קבוצה מסודרים ב"עץ הפוך", בו כל שחקן מצביע על שחקן אחר עד הגעה לשורש העץ ההפוך. "השאיפה" (לפי אלגוריתם Union-Find הנלמד בכיתה) היא שיהיה שחקן שהוא שורש ה"עץ ההפוך", וכל יתר השחקנים באותה קבוצה יצביעו עליו.
  + השחקן שהוא שורש ה"עץ ההפוך" מצביע על הקבוצה בה משחק, והקבוצה בה משחק מצביע על השחקן שהוא השורש.
  + חשוב לציין כי ה"עץ ההפוך" ממומש בעזרת מצביעים בין השחקנים עצמם.
* **מחלקת Player** תכיל את כל הפרטים הבסיסיים של אותו השחקן: מזהה שחקן, מספר המשחקים שהשחקן שיחק (עם התאמה לפי מיקומו בעץ ההפוך – הסבר בפעולה " num\_played\_games\_for\_player()"), היכולת של השחקן, מספר הכרטיסים שקיבל והאם יכול לשחק כשוער. יש לשחקן גם שני משתנים מטיפוס permutation\_t – אחד המכיל את ה"רוח" של אותו שחקן, ואחד המכיל את "הרוח" החלקי של אותו השחקן. ה"רוח" החלקי יבוא לשימוש לחישוב ה"רוח" החלקי של הקבוצה בה השחקן משחק (כפל ה"רוחות" של שחקני הקבוצה לפי סדר הוספתם לקבוצה, עד הוספת השחקן הנוכחי).

בנוסף יכיל מצביע לקבוצה בה משחק (אשר יהיה בשימוש (ולא יהיה nullptr) רק כאשר אותו השחקן הוא שורש ה"עץ ההפוך" של שחקני הקבוצה כפי שמתואר למעלה).

כמו כן, השחקן מכיל מצביע לשחקן אחר המשחק איתו באותה קבוצה.

נשים לב כי נוצר עותק אחד בלבד של כל שחקן וקבוצה המשתתפים בגביע העולם, והעצים/המצביעים השונים מצביעים לאותם העותקים.

**סיבוכיות מקום:** בכל פעולה, אין הוספה של סיבוכיות מקום מעבר לנדרש לכלל התרגיל. לכן סיבוכיות המקום נשאר O(n+k), כאשר n הוא כמות השחקנים ו-k הוא כמות הקבוצות בגביע העולם.

**להלן רשימת הפעולות הנדרשות לממשק והסבר למימושם והסיבוכיות שלהם:**

**world\_cup\_t –**

נאתחל את שני עצי הAVL בעזרת קריאה לconstructor-ים שלהם. נאתחל את ספירת כמות השחקנים והקבוצות ל-0, ואת גודל טבלת הערבול הנוכחי ל-7.

נאתחל את טבלת הערבול לפי הגודל המוגדר לעיל (7), ונעבור על כל תא של המערך ונאתחל אותו לעץ מטיפוס שחקנים בעזרת קריאה ל-constructor של העץ AVL.

סיבוכיות זמן:

נאתחל את כל המשתנים לint-ים או לערכים דיפולטים. עבור העצים – עץ ריק מכיל רק node אחד, וסיבוכיות הזמן ליצירתו הוא O(1). ולכן סיבוכיות הזמן הוא **O(1)**.

נשים לב כי המערך בהתחלה הוא בגודל 7 ולכן בעל סיבוכיות מקום של O(7) = O(1).

**world\_cup\_t~ –**

ראשית נשחרר את ה-data של כל השחקנים והקבוצות בגביע העולם. נעשה זאת ע"י שחרור הזכרון של ה-data בעץ m\_teamsByID ובטבלת הערבול -m\_playersHashTable שמכילים את כל הקבוצות וכל השחקנים. כמו כן, נמחוק את כל העצים בטבלת הערבול.

לאחר מכן, נמחוק את טבלת הערבול. בסוף, יקראו ה-destructor-ים הדיפולטים של עצי השחקנים יקראו, וישחררו את הזיכרון של ה-nodes של העצים, ובכן יקראו ה-destructor-ים הדיפולטים של המשתנים שנותרו.

סיבוכיות זמן:

כדי לשחרר את ה-data, נעבור על כל הקבוצות והשחקנים שקיימים בגביע העולם, ונשחרר את הזיכרון. שחרור הזיכרון של הקבוצות נעשה במעבר על כלל הקבוצות ובסיבוכיות של O(k). שחרור הזיכרון של השחקנים, נעבור על כל תא בטבלת הערבול. בכל תא של טבלת הערבול, נעבור על כלל השחקנים בעץ של אותו התא ונמחוק אותם. לאחר מכן, נעבור שוב על אותו העץ של שחקנים ונשחרר את זכרון ה-nodes של העצים. בתהליך זה נעבור על n תאים בטבלת הערבול, ונעבור סך הכל על n שחקנים\nodes בכל העצים בטבלת הערבול, ולכן סך סיבוכיות הזמן הוא O(3n) = O(n).

לאחר מכן, יקראו ה-destructor-ים הדיפולטים של עצי הקבוצות, אשר עוברים על כל ה-nodes בגביע העולם:

* + יש לנו שני עצים של קבוצות, ולכן עבור k קבוצות, יש לכל עץ k nodes. לכן סיבוכיות הזמן של מעבר על העצים הללו הוא O(k) O(2k) =.

לכן סיבוכיות הזמן סך הכל הוא **O(n+k).**

**add\_team –**

נוסיף קבוצה חדשה לגביע העולם.

ראשית, אם התקבל מזהה קבוצה קטן או שווה ל-0 נחזיר INVALID\_INPUT. אם לא, אז ניצור קבוצה חדשה ונכניס אותה לעץ הקבוצות המסודר לפי מזהה קבוצה ועץ הקבוצות המסודר לפי יכולת הקבוצה. במקרה שכבר קיימת קבוצה עם מזהה זה נחזיר FAILURE, ובמקרה של בעיה בהקצאה זיכרון נחזיר ALLOCATION\_ERROR. נוסיף אחד למשתנה שסופר את כמות הקבוצות בגביע העולם ובסוף נחזיר SUCCESS.

סיבוכיות זמן:

נעשה חיפוש לפי מזהה קבוצה בעץ AVL m\_teamsByID כדי להכניס את הקבוצה החדשה, בסיבוכיות של O(log(k)) כאשר k היא כמות הקבוצות בגביע העולם. לאחר ההכנסה, נבצע גלגולים בעץ כדי לאזן אותו בחזרה, בסיבוכיות של O(1) לגלגול כפי שנלמד בהרצאה. נעבור על כל ה-nodes במעלה העץ (מקסימום log(k) nodes) עד השורש בעדכון גובה העץ וה-balance factor, ולכן איזון העץ הוא בסיבוכיות O(log(k)). את אותו התהליך נעשה גם בהכנסת הקבוצה לעץ m\_teamsByAbility, ולכן סיבוכיות ההכנסה עבורו גם O(log(k)). לכן סך סיבוכיות הזמן של הפעולה היא **O(log(k)).**

**remove\_team –**

נסיר את הקבוצה מגביע העולם.

ראשית נוודא את תקינות הקלט, ונחזיר INVALID\_INPUT אם התקבל מזהה קבוצה קטן או שווה ל-0.

נחפש בעץ AVL של הקבוצות המסודרות לפי מזהה קבוצה - m\_teamsByID, ונחזיר FAILURE במקרה שהקבוצה לא קיימת. אחרת, נסיר את הקבוצה על ידי הסרת הnode מהעצי AVL m\_teamsByID ו-m\_teamsByAbility. כאשר מסירים את הקבוצה מהעץ דרגות m\_teamsByAbility, מעדכנים את הדרגה של כל node בעליה במסלול החיפוש. אם לקבוצה קיימים שחקנים, אז נוסיף את כמות המשחקנים שהקבוצה שיחקה לשחקן אשר הוא השורש ב"עץ ההפוך" של שחקני הקבוצה. כמו כן, "ננתק" את השחקנים מהקבוצה על ידי כך שהשחקן שהוא שורש ה"עץ ההפוך" של שחקני הקבוצה כעת יצביע ל- nullptr (במקום לקבוצה שאותה מסירים. נשחרר את הזכרון של הקבוצה הזו, נחסיר 1 מספירת כלל הקבוצות בגביע העולם, ובסוף נחזיר SUCCESS.

סיבוכיות זמן:

בפעולה זו אנו מחפשים את הקבוצה המבוקשת כמה פעמים: פעם אחת לבדוק אם קיימים שחקנים בקבוצה – חיפוש בעץ m\_teamsByID ועוד פעמיים כדי להסיר את הקבוצה מ-m\_teamsByID ו-m\_teamsByAbility. בשני המקרים, סיבוכיות הזמן של החיפוש הוא O(log(k)) כי יש מקסימום k קבוצות בעצים הללו. ההסרה מהעץ כוללת גלגולים בשיטה שנלמדה בהרצאה – בסיבוכיות O(1) לכל גלגול. בפוטנציאל יש log(k) גלגולים כי עוברים על כל הnodes במעלה העצים. נשים לב כי עדכון הדרגה של ה-nodes בעץ הדרגות m\_teamsByAbility לאחר הסרת הקבוצה, מתבצע לאורך מסלול החיפוש חזרה לשורש, ולכן מתבצע בסיבוכיות של O(log(k)). לכן סך סיבוכיות ההסרה של הקבוצה משני העצים הוא O(log(k)). עדכון השחקן שהוא שורש ה"עץ ההפוך" של אותה קבוצה נעשה ישירות דרך המשתנים הרלוונטיים ולכן בסיבוכיות זמן של O(1). כמו כן גם מחיקת הקבוצה ועדכון כמות הקבוצות בגביע העולם. לכן סך סיבוכיות הזמן הוא **O(log(k))**.

**add\_player –**

נוסיף שחקן לגביע העולם.

ראשית נוודא את תקינות הקלט, ונחזיר INVALID\_INPUT אם התקבלו ערכים לא תקינים. בדומה לכך, נחזיר FAILURE במקרה ששחקן כבר קיים עם אותו מזהה שחקן, או אם לא קיימת קבוצה עם מזהה הקבוצה הנתונה. נחשב את כמות המשחקים שישחק השחקן ואת ה"רוח" החלקי שלו:

* + לערך של כמות המשחקים שהשחקן שיחק, נחסיר את כמות המשחקים שהקבוצה שלו שיחקה לפני שהצטרף, ובכן את כמות המשחקים ששיחק השחקן שהוא שורש ה"עץ ההפוך" של שחקני הקבוצה.
  + נעשה תהליך דומה עם ה"רוח" החלקי של אותו שחקן: נחשב אותו לפי ה"רוח" הקבוצתי אשר הוא כפל "רוחות" השחקנים של הקבוצה עד הצטרפותו של השחקן החדש, ונכפיל אותו ב"רוח" של השחקן החדש. נכפיל את הערך המתקבל בהפיך של הרוח החלקי של השחקן שהוא שורש ה"עץ ההפוך" של שחקני הקבוצה.
  + החישובים הנ"ל מאפשרים עדכון והחזרת כמות המשחקים של השחקן וה"רוח" החלקי של השחקן בסיבוכיות זמן מינימלית.

ניצור שחקן חדש עם הפרטים הנתונים והחדשים שחישבנו לעיל.

נכניס את השחקן לטבלת הערבול בדרך הבאה:

* + אם פקטור העומס של טבלת הערבול שווה ל-1 (כולל השחקן החדש שכעת הצטרף), נגדיל את טבלת הערבול כפי שראינו בהרצאה.
    - במקרה הזה, נכניס מחדש את כלל השחקנים בגביע העולם לטבלת הערבול הגדול יותר בעזרת פונקצית האש המעודכנת.
    - נשחרר את הזיכרון של הטבלה הישנה.
    - נעדכן את פונקצית ההאש, כפי שראינו בהרצאה.
  + נכניס את השחקן החדש לטבלת הערבול לפי פונקציית ההאש שהוא f(x) = xmodk כאשר k הוא גודל טבלת הערבול.

אם השחקן הוא הראשון בקבוצה, אז השחקן הזה הופך לשורש ה"עץ ההפוך" של שחקני הקבוצה. נעדכן בהתאם את המצביע לשורש ה"עץ ההפוך" של הקבוצה ואת המצביע לקבוצה של השחקן.

את הקבוצה בה השחקן משחק מהעץ דרגות m\_teamsByAbility כדי שנוכל להכניסו מחדש לאחר עדכון הקבוצה. נעדכן את הקבוצה בפרטים של השחקן: נעדכן את "רוח" הקבוצה, יכולת הקבוצה, כמות השחקנים בקבוצה וכמות השוערים בקבוצה. נכניס מחדש את הקבוצה המעודכנת לעץ m\_teamsByAbility, ונוסיף אחד לספירת כמות השחקנים בגביע העולם. במקרה של כישלון בהקצאת זיכרון בכל שלב בפעולה, נחזיר ALLOCATION\_ERROR. אחרת נחזיר SUCCESS בסוף הפעולה.

סיבוכיות זמן:

כדי לבדוק אם השחקן קיים, נחשב את מיקומו בטבלת הערבול בעזרת פונקציית ההאש בסיבוכיות O(1), ואז ניגש ישירות לעץ השחקנים בתא זה בטבלת הערבול – גם בסיבוכיות O(1). נחפש את השחקן בעץ הזה. לפי הנחת הפיזור, כמות השחקנים בכל תא בטבלת הערבול הוא בממוצע 1 (כלומר, גודל העץ הממוצע הוא 1), ולכן החיפוש הוא בסיבוכיות משוערכת בממוצע על הקלט הוא O(1).

חיפוש הקבוצה בעץ AVL של הקבוצות המסודר לפי מזהה קבוצה הוא בסיבוכיות O(log(k)) כאשר k הוא כמות הקבוצות בגביע העולם (כפי שנלמד בכיתה). החישובים של כמות המשחקים שהשחקן שיחק וה"רוח" החלקי שלו נעשה בגישה ישירה למשתנים הרלוונטיים ולכן בסיבוכיות של O(1) (גם החישוב של הpermutation\_t ההפיך וכפל permutation\_t הוא בעזרת לולאה באורך קבוע (5) ולכן בסיבוכיות O(1)).

בהגדלת טבלת הערבול, גודל הטבלה החדשה הוא כעת 2n כאשר n הוא כמות השחקנים בגביע העולם. ניצור עצים חדשים בכל תא בטבלה החדשה – בסיבוכיות O(2n) = O(n). ניצור מערך בגודל n ונאתחל את איבריו ל-nullptr. זהו גם בסיבוכיות O(2n) = O(n). נעבור על כל התאים בטבלת הערבול הישן (בגודל n), ונאסוף את כל השחקנים במערך שיצרנו. סה"כ נעבור על n תאים בטבלת הערבול, וסך הכל על n שחקנים ולכן סיבוכיות פעולה זו היא גם O(2n) = O(n). נכניס כל אחד מהשחקנים לטבלת ערבול החדשה. לפי הנחת הפיזור, כמות השחקנים בכל תא בטבלת הערבול הוא בממוצע 1 (כלומר, גודל העץ הממוצע הוא 1), ולכן גם ההכנסה הוא בסיבוכיות משוערכת בממוצע על הקלט של O(1). בסוף התהליך נעבור על כל תא בטבלת הערבול הישן (בגודל n) ונשחרר את הזיכרון של העצים – סך הכל נעבור על n nodes (כי יש סך הכל n שחקנים בכל הטבלת ערבול) ולכן סיבוכיות פעולה זו היא גם O(2n) = O(n).

נשים לב כי פעולה זו של הגדלת הטבלת ערבול מתרחשת רק כאשר פקטור העומס שווה ל-1 (כמות השחקנים בגביע העולם שווה לגודל טבלת הערבול). לכן הסיבוכיות המשוערכת בממוצע על הקלט עבור הגדלת הטבלה הוא O(1) כפי שנלמד בהרצאה.

נכניס את השחקן החדש לטבלת הערבול. כפי שמתואר למעלה, פעולה זו היא בסיבוכיות משוערכת O(1).

עדכון המצביעים של השחקן הקבוצה נעשה בגישה ישירה של המשתנים ולכן בסיבוכיות של O(1). הסרת הקבוצה מהעץ דרגות m\_teamsByAbility והכנסתו בחזרה כוללת חיפוש הקבוצה הרלוונטית בסיבוכיות O(log(k)) וגלגולים בשיטה שנלמדה בהרצאה – בסיבוכיות O(1) לכל גלגול. בפוטנציאל יש log(k) גלגולים כי עוברים על כל הnodes במעלה העצים. נשים לב כי עדכון הדרגה של ה-nodes בעץ הדרגות m\_teamsByAbility לאחר הסרת והוספת הקבוצה, מתבצע לאורך מסלול החיפוש חזרה לשורש, ולכן מתבצע בסיבוכיות של O(log(k)). לכן ההסרה וההכנסה כל אחת בסיבוכיות O(log(k)). עדכון פרטי הקבוצה נעשה ישירות בעזרת המשתנים של הקבוצה ולכן בסיבוכיות של O(1).

לכן סך הסיבוכיות המשוערכת בממוצע על הקלט הוא **O(log(k)).**

**play\_match –**

שתי קבוצות ישחקו משחק.

ראשית נבדוק את תקינות הקלט, ונחזיר INVALID\_INPUT אם התקבלו מזהי קבוצות קטנים או שווים ל-0 או מזהי קבוצות שוות. נחזיר FAILURE אם לא קיימים קבוצות עם המזהים הנתונים בגביע העולם או אם אחת הקבוצות לא כשרות לשחק (בעלי לפחות שוער אחד). הקבוצה המנצחת היא הקבוצה בעלת היכולת הגבוהה ביותר, אשר היא סכום בין מספר הנקודות + סכום היכולות של כלל שחקני הקבוצה. לקבוצה המנצחת נוסיף 3 נקודות. אם יש שוויון, נחשב את ה"כוח הרוחני" של הקבוצות, והקבוצה בעלת ה"הכוח הרוחני" הגדול ביותר תנצח ב-3 נקודות. במקרה של תיקו אז כל קבוצה תזכה בנקודה אחת. בסוף נוסיף 1 מספר המשחקים ששיחק כל קבוצה ונחזיר את ערכי החזרה כפי שנדרש בתרגיל.

סיבוכיות זמן:

חיפוש הקבוצות בעץ AVL של m\_teamsByID הוא בסיבוכיות O(log(k)) כפי שנלמד בהרצאה, כאשר k הוא כמות הקבוצות בגביע העולם. הבדיקה אם הקבוצות כשרות נעשה באופן ישיר דרך משתנים של הקבוצות ולכן בעל סיבוכיות זמן של O(1), בדומה גם לחישוב היכולת של הקבוצות. חישוב ה"כוח הרוחני" של כל קבוצה נעשה בעזרת לולאה באורך קבוע (5) ולכן גם בעלת סיבוכיות זמן של O(1). עדכון נקודות לקבוצות הזוכות או במקרה של תיקו וכמו כן עדכון כמות המשחקים ששיחקה כל קבוצה נעשה באופן ישיר בעזרת גישה למשתני הקבוצות ולכן גם בסיבוכיות של O(1). לכן סך הכל, סיבוכיות הזמן של פעולה זו היא **O(log(k)).**

**num\_played\_games\_for\_player –**

נחזיר את כמות המשחקים ששחקן שיחק.

ראשית נבדוק את תקינות הקלט, ונחזיר INVALID\_INPUT במקרה שהתקבל מזהה שחקן הקטן או שווה ל-0. נחשב את מיקום השחקן בטבלת הערבול בעזרת הפונקציית האש, ניגש לתא זה ישירות בטבלה ונחפש את השחקן בעץ המתאים, כדי לבדוק אם השחקן נמצא. במקרה שלא נחזיר FAILURE.

במקרה שהשחקן לא מחובר ישירות לשורש ה"עץ ההפוך" של השחקנים באותה קבוצה, נבצע פעולת find() לפי אלגוריתם הUNION-FIND שנלמדה בהרצאה:

* + נעדכן שלושה משתנים: כמות המשחקים שהשחקן שיחק, ה"רוח" החלקי שלו ואת המצביע לשחקן מעליו ב"עץ ההפוך" באותה שיטה של רקורסיה:
    - נעבור על השחקנים מהשחקן הנוכחי עד השורש, ו"נשטח" אותם כך שיצביעו ישירות על השורש.
    - נעדכן את הנתונים הנוספים כך שיהיו תלויים אך ורק בשורש בלא בשחקנים האחרים ביניהם.

כעת השחקן מצביע ישירות על שורש ה"עץ ההפוך" של שחקני הקבוצה, שהוא מצביע על הקבוצה המתאימה. נוסיף לכמות המשחקים של השחקן, את כמות המשחקים של השחקן בשורש ה"עץ ההפוך" (אם הוא קיים) ואת כמות המשחקים של הקבוצה בה משחק (אם לא הודחה). נשים לב שאנו מוסיפים את הערכים הללו כי החסרנו אותם בתהליך הכנסת השחקן (פעולה add\_player). נחזיר את הערך המתקבל.

סיבוכיות זמן:

חישוב מיקום השחקן בטבלת הערבול נעשה בסיבוכיות O(1),והגישה למיקום זה בטבלה הוא בסיבוכיות O(1). נחפש את השחקן בעץ הזה. לפי הנחת הפיזור, כמות השחקנים בכל תא בטבלת הערבול הוא בממוצע 1 (כלומר, גודל העץ הממוצע הוא 1), ולכן החיפוש הוא בסיבוכיות משוערכת בממוצע על הקלט הוא O(1).

כפי שראינו בהרצאה, הסיבוכיות המשוערכת בממוצע על הקלט של כל פעולות הfind() ו-union() בכל הפעולות (חוץ מ-add\_player) הוא log\*(n) כאשר n הוא כמות השחקנים בגביע העולם.

עדכון הערך של כמות המשחקים שהשחקן שיחק נעשה באופן ישיר דרך המשתנים של השחקנים והקבוצה ולכן בסיבוכיות של O(1). לכן סך הסיבוכיות המשוערכת בממוצע על הקלט עבור פעולה זו היא **O(log\*(n))**.

**add\_player\_cards –**

נעדכן את כמות הכרטיסים ששחקן קיבל.

ראשית נבדוק את תקינות הקלט, ונחזיר INVALID\_INPUT במקרה שהתקבל מזהה שחקן הקטן או שווה ל-0 או אם כמות הכטיסים שלילי. נבדוק אם השחקן הודח מגביע העולם על ידי הפעולות הבאות:

* + נחשב את מיקום השחקן בטבלת הערבול בעזרת הפונקציית האש, ניגש לתא זה ישירות בטבלה ונחפש את השחקן בעץ המתאים, כדי לבדוק אם השחקן נמצא. במקרה שלא נחזיר FAILURE.
  + במקרה שהשחקן לא מחובר ישירות לשורש ה"עץ ההפוך" של השחקנים באותה קבוצה, נבצע פעולת find() לפי אלגוריתם הUNION-FIND שנלמדה בהרצאה:
    - נעדכן שלושה משתנים: כמות המשחקים שהשחקן שיחק, ה"רוח" החלקי שלו ואת המצביע לשחקן מעליו ב"עץ ההפוך" באותה שיטה של רקורסיה:
      * נעבור על השחקנים מהשחקן הנוכחי עד השורש, ו"נשטח" אותם כך שיצביעו ישירות על השורש.
      * נעדכן את הנתונים הנוספים כך שיהיו תלויים אך ורק בשורש בלא בשחקנים האחרים ביניהם.
  + כעת השחקן מצביע ישירות על שורש ה"עץ ההפוך" של שחקני הקבוצה, שהוא מצביע על הקבוצה המתאימה. נבדוק אם שורש ה"עץ ההפוך" של שחקני הקבוצה מצביע ל-nullptr. במקרה שכן סימן שהקבוצה הודחה מהטורניר, ובהתאם לכך גם השחקן הזה, ונחזיר FAILURE בהתאם.

אחרת, נחשב את מיקום השחקן בטבלת הערבול בעזרת הפונקציית האש, ניגש לתא זה ישירות בטבלה ונחפש את השחקן בעץ המתאים. נעדכן את כמות הכרטיסים שלו ונחזיר SUCCESS.

סיבוכיות זמן:

חישוב מיקום השחקן בטבלת הערבול נעשה בסיבוכיות O(1),והגישה למיקום זה בטבלה הוא בסיבוכיות O(1). נחפש את השחקן בעץ הזה. לפי הנחת הפיזור, כמות השחקנים בכל תא בטבלת הערבול הוא בממוצע 1 (כלומר, גודל העץ הממוצע הוא 1), ולכן החיפוש הוא בסיבוכיות משוערכת בממוצע על הקלט הוא O(1).

כפי שראינו בהרצאה, הסיבוכיות המשוערכת בממוצע על הקלט של כל פעולות הfind() ו-union() בכל הפעולות (חוץ מ-add\_player) הוא log\*(n) כאשר n הוא כמות השחקנים בגביע העולם. הבדיקה אם הקבוצה הודחה נעשית באופן ישיר דרך מצביעים של השחקנים הרלוונטיים, לכן סך הסיבוכיות המשוערכת בממוצע על הקלט עבור פעולה זו היא **O(log\*(n)).**

**get\_player\_cards –**

נחזיר את כמות הכרטיסים ששחקן קיבל.

ראשית נבדוק את תקינות הקלט, ונחזיר INVALID\_INPUT במקרה שהתקבל מזהה שחקן הקטן או שווה ל-0. נחשב את מיקום השחקן בטבלת הערבול בעזרת הפונקציית האש, ניגש לתא זה ישירות בטבלה ונחפש את השחקן בעץ המתאים. אם השחקן לא נמצא, נחזיר FAILURE.

נחזיר את כמות הכרטיסים של השחקן.

סיבוכיות זמן:

חישוב מיקום השחקן בטבלת הערבול נעשה בסיבוכיות O(1),והגישה למיקום זה בטבלה הוא בסיבוכיות O(1). נחפש את השחקן בעץ הזה. לפי הנחת הפיזור, כמות השחקנים בכל תא בטבלת הערבול הוא בממוצע 1 (כלומר, גודל העץ הממוצע הוא 1), ולכן החיפוש הוא בסיבוכיות משוערכת בממוצע על הקלט הוא O(1).

משיכת כמות הכרטיסים של השחקן נעשית באופן ישיר דרך השחקן, לכן סך סיבוכיות הזמן עבור פעולה זו בממוצע על הקלט היא **O(1).**

**get\_team\_points–**

נחזיר את כמות הנקודות של הקבוצה.

ראשית נבדוק את תקינות הקלט, ונחזיר INVALID\_INPUT במקרה שהתקבל מזהה קבוצה הקטן או שווה ל-0. נחפש את הקבוצה בעץ m\_teamsByID, ואם השחקן לא נמצא, נחזיר FAILURE.

נחזיר את כמות הנקודות של הקבוצה.

סיבוכיות זמן:

חיפוש הקבוצה בעץ הAVL נעשה בסיבוכיות O(log(k)) כפי שנלמד בהרצאה, כאשר k הוא כמות הקבוצות בגביע העולם. משיכת כמות הנקודות של הקבוצה נעשית באופן ישיר דרך הקבוצה, לכן סך סיבוכיות הזמן עבור זה הוא O(1). סך סיבוכיות הזמן עבור פעולה זו היא **O(log(k)).**

**get\_ith\_pointless\_ability–**

נחזיר את היכולת של השחקן ה – i.

ראשית נבדוק את תקינות הקלט, ונחזיר FAILURE אם התקבל אינדקס i שלילי, אם אין קבוצות בגביע העולם כלל, או אם האינדקס i שהתקבל גדול מכמות הקבוצות בגביע העולם.

נתחיל בחיפוש אחר האינדקס הזה בעץ m\_teamsByAbility, שהוא עץ דרגות בו הקבוצות מסודרות לפי היכולת של הקבוצה (אשר היא סכום היכולות של כלל שחקניה). נרד בעץ דרגות עד הגעה ל-node עם האינדקס המתאים, ונחזיר את ה-node שנמצא.

בקבלת ה-node, נשלוף את הקבוצה שאליו מצביע ואז את מזהה הקבוצה, אותו נחזיר.

סיבוכיות זמן:

נחפש את ה-node בעץ AVL ודרגות של m\_teamsByAbility בסיבוכיות של O(log(k)) כפי שנלמד בהרצאה, כאשר k היא כמות הקבוצות בגביע העולם. הגישה לקבוצה ולמזהה שלו הוא באופן ישיר לאחר מציאת הקבוצה בעץ, ולכן בסיבוכיות של O(1). לכן סך סיבוכיות הזמן עבור פעולה זו היא **O(log(k)).**

**get\_partial\_spirit –**

נחזיר את ה"רוח" החלקי של השחקן.

ראשית נבדוק את תקינות הקלט, ונחזיר INVALID\_INPUT במקרה שהתקבל מזהה שחקן הקטן או שווה ל-0. נבדוק אם השחקן הודח מגביע העולם על ידי הפעולות הבאות:

* + נחשב את מיקום השחקן בטבלת הערבול בעזרת הפונקציית האש, ניגש לתא זה ישירות בטבלה ונחפש את השחקן בעץ המתאים, כדי לבדוק אם השחקן נמצא. במקרה שלא נחזיר FAILURE.
  + במקרה שהשחקן לא מחובר ישירות לשורש ה"עץ ההפוך" של השחקנים באותה קבוצה, נבצע פעולת find() לפי אלגוריתם הUNION-FIND שנלמדה בהרצאה:
    - נעדכן שלושה משתנים: כמות המשחקים שהשחקן שיחק, ה"רוח" החלקי שלו ואת המצביע לשחקן מעליו ב"עץ ההפוך" באותה שיטה של רקורסיה:
      * נעבור על השחקנים מהשחקן הנוכחי עד השורש, ו"נשטח" אותם כך שיצביעו ישירות על השורש.
      * נעדכן את הנתונים הנוספים כך שיהיו תלויים אך ורק בשורש בלא בשחקנים האחרים ביניהם.
  + כעת השחקן מצביע ישירות על שורש ה"עץ ההפוך" של שחקני הקבוצה, שהוא מצביע על הקבוצה המתאימה. נבדוק אם שורש ה"עץ ההפוך" של שחקני הקבוצה מצביע ל-nullptr. במקרה שכן סימן שהקבוצה הודחה מהטורניר, ובהתאם לכך גם השחקן הזה, ונחזיר FAILURE בהתאם.

אחרת, נחשב את מיקום השחקן בטבלת הערבול בעזרת הפונקציית האש, ניגש לתא זה ישירות בטבלה ונחפש את השחקן בעץ המתאים.

במקרה שהשחקן לא מחובר ישירות לשורש ה"עץ ההפוך" של השחקנים באותה קבוצה, נבצע פעולת find() לפי אלגוריתם הUNION-FIND שנלמדה בהרצאה:

* + נעדכן שלושה משתנים: כמות המשחקים שהשחקן שיחק, ה"רוח" החלקי שלו ואת המצביע לשחקן מעליו ב"עץ ההפוך" באותה שיטה של רקורסיה:
    - נעבור על השחקנים מהשחקן הנוכחי עד השורש, ו"נשטח" אותם כך שיצביעו ישירות על השורש.
    - נעדכן את הנתונים הנוספים כך שיהיו תלויים אך ורק בשורש בלא בשחקנים האחרים ביניהם.

כעת השחקן מצביע ישירות על שורש ה"עץ ההפוך" של שחקני הקבוצה. נכפיל את ה"רוח" החלקי של השחקן הנוכחי, ב"רוח" החלקי של שורש ה"עץ ההפוך" של שחקני הקבוצה (אם הוא קיים). נשים לב נכפיל ערך זה כי הכפלנו את ההפיך שלו בהוספת השחקן (פעולה add\_player). נחזיר את הערך המתקבל.

סיבוכיות זמן:

חישוב מיקום השחקן בטבלת הערבול נעשה בסיבוכיות O(1),והגישה למיקום זה בטבלה הוא בסיבוכיות O(1). נחפש את השחקן בעץ הזה. לפי הנחת הפיזור, כמות השחקנים בכל תא בטבלת הערבול הוא בממוצע 1 (כלומר, גודל העץ הממוצע הוא 1), ולכן החיפוש הוא בסיבוכיות משוערכת בממוצע על הקלט הוא O(1).

כפי שראינו בהרצאה, הסיבוכיות המשוערכת בממוצע על הקלט של כל פעולות הfind() ו-union() בכל הפעולות (חוץ מ-add\_player) הוא log\*(n) כאשר n הוא כמות השחקנים בגביע העולם.

שליפת ה"רוח" החלקי של השחקן נעשה באופן ישיר דרך המשתנים המתאימים, וכפל ה"רוח" החלקי נעשה בעזרת לולאה בגודל קבוע (5). לכן סיבוכיות הזמן עבור פעולות אלו הוא O(1). לכן סך הסיבוכיות המשוערכת בממוצע על הקלט עבור פעולה זו היא **O(log\*(n))**.

**buy\_team –**

קבוצה אחת קונה את השנייה.

ראשית נבדוק את תקינות הקלט, ונחזיר INVALID\_INPUT אם התקבל מזהי קבוצה קטנים או שווים ל-0 או אם מזהי הקבוצה שווים. נחפש את הקבוצות בעץ m\_teamsByID, ואם הן לא נמצאות אז נחזיר FAILURE. אם לשתי הקבוצות יש שחקנים, נעשה את הפעולות הבאות:

* + נעדכן את כמות המשחקים שכל קבוצה שיחקה בשחקן שהוא השורש של "עץ ההפוך" של שחקני הקבוצה.
  + נבצע פעולת union() כחלק מאלגוריתם Union-Find שנלמדה בהרצאה בין ה"עצים ההפוכים" של הקבוצות הנדרשות:
    - נאחד את הקבוצה הקטנה לקבוצה הגדולה.
    - נעדכן את המצביעים בין השחקנים והקבוצות בהתאם.
    - נעדכן בהתאם את ה"רוח" החלקי של שורשי ה"עצים ההפוכים".
    - נעדכן את כמות המשחקים ששיחקו השחקנים שהם שורשי ה"עצים ההפוכים".
  + נעדכן את המצביע של הקבוצה שקנתה ל"עץ ההפוך" המאוחד של השחקנים, ונעדכן את המצביע של שורש ה"עץ ההפוך" לקבוצה שקנתה.

אחרת, אם לאחת מהקבוצות יש שחקנים ולשנייה אין, נעדכן את נתוני כמות המשחקים שהקבוצות שיחקו בשורשי ה"עצים ההפוכים" ואת המצביעים המתאימים בין הקבוצה לשורש ה"עץ ההפוך" החדש וההפך בהתאם.

נעדכן את נתוני הקבוצה החדשה להיות איחוד הנתונים של הקבוצות שהתאחדו – כולל כפל ה"רוח" של הקבוצות שהתאחדו, חיבור מספר הנקודות של הקבוצות שהתאחדו, חיבור מספר השחקנים והשוערים של הקבוצות שהתאחדו, ובכן חיבור היכולת הקבוצתית של הקבוצות שהתאחדו. בנוסף לכך, נאתחל את כמות המשחקים של הקבוצה החדשה ל-0.

נסיר את הקבוצה שנקנתה על ידי פעולת remove\_team המתוארת למעלה, ונסיר ונכניס מחדש את הקבוצה שקנתה לעץ הדרגות m\_teamsByAbility כדי למקם אותו מחדש עם הנתונים העדכניים.

במקרה של בעיה בהקצאת זיכרון בכל שלב בפעולה, נחזיר ALLOCATION\_ERROR. בסוף נחזיר SUCCESS.

סיבוכיות זמן:

חיפוש הקבוצות בעץ AVL של m\_teamsByID הוא בסיבוכיות O(log(k)) כפי שנלמד בהרצאה, כאשר k הוא כמות הקבוצות בגביע העולם. איחוד ה"עצים ההפוכים" נעשה בשיטה שנלמדה בהרצאה, כחלק מאלגוריתם Union-Find. לכן הסיבוכיות המשוערכת בממוצע על הקלט עבור כל פעולות ה-find ו-union בכל הפעולות (חוץ מadd\_player) הוא O(log\*(n)) כפי שנלמד בהרצאה, כאשר n הוא כמות השחקנים בגביע העולם.

עדכון כלל נתוני המשחקים שישחקו הקבוצות והשחקנים הרלוונטיים, ועדכון המצביעים השונים נעשה באופן ישיר באמצעות משתנים של הקבוצות והשחקנים, ולכן נעשה בסיבוכיות O(1). כפל וחישוב הפיך של משתנים מטיפוס permutation\_t נעשה בעזרת לולאות באורך קבוע (5) ולכן גם בסיבוכיות O(1). עדכון פרטי ה"רוח" החלקי של השחקנים נעשה בעזרת גישה ישירה דרך המשתנים הרלוונטיים, ולכן גם בסיבוכיות O(1).

עדכון פרטי הקבוצה נעשה באופן ישיר דרך המשתנים של הקבוצות ולכן גם בסיבוכיות של O(1).

הסרת הקבוצה בעזרת פעולת remove\_team הוא בסיבוכיות O(log(k)) כפי שחושב למעלה. הסרת הקבוצה מהעץ דרגות m\_teamsByAbility והכנסתו בחזרה כוללת חיפוש הקבוצה הרלוונטית בסיבוכיות O(log(k)) וגלגולים בשיטה שנלמדה בהרצאה – בסיבוכיות O(1) לכל גלגול. בפוטנציאל יש log(k) גלגולים כי עוברים על כל הnodes במעלה העצים. נשים לב כי עדכון הדרגה של ה-nodes בעץ הדרגות m\_teamsByAbility לאחר הסרת והוספת הקבוצה, מתבצע לאורך מסלול החיפוש חזרה לשורש, ולכן מתבצע בסיבוכיות של O(log(k)). לכן ההסרה וההכנסה כל אחת בסיבוכיות O(log(k)).

לכן סך הסיבוכיות המשוערכת בממוצע על הקלט עבור פעולה זו היא **O(log(k)+log\*(n)).**