**תרגיל רטוב 1 – החלק היבש**

שרה גריפית – ת"ז: 341312304

ליאור בר יוסף – ת"ז: 207022443

עבור האפליקציה של פיפ"א, נבנה מבנה נתונים המורכב מארבע עצי AVL מרכזיים:

* **m\_teamsByID** – כל הקבוצות שמשתתפות בגביע העולם, מסודרים לפי מזהה הקבוצה.
* **m\_qualifiedTeams** – כל הקובצות המשתתפות בגביע העולם אשר יכולים לשחק (בעלי לפחות 11 שחקנים שלפחות אחד יכול לשחק כשוער).
* **m\_playersByID** – כל השחקנים שמשתתפים בגביע העולם, מסודרים לפי מזהה השחקן.
* **m\_playersByScore** – כל השחקנים שמשתתפים בגביע העולם, מסודרים לפי כמות השערים שהבקיעו + הכרטיסים שקיבלו + מזהה השחקן (במקרה של שוויון).

נוסיף שתי מחלקות: Player ו-Team אשר מכילים את הפרטים של השחקנים ושל הקבוצות כדורגל המשתתפים בגביע העולם.

* **מחלקת Player** תכיל את כל הפרטים הבסיסיים על אותו השחקן: מזהה שחקן, מספר המשחקים שהשחקן שיחק (בנפרד למשחקים ששיחק עם קבוצתו הנוכחית), מספר השערים שהבקיע, מספר הכרטיסים שקיבל והאם יכול לשחק כשוער.
* **מחלקת Team** תכיל את הפרטים הבסיסיים של אותה קבוצה: מזהה קבוצה, מספר הנקודות של הקבוצה, מספר השחקנים בקבוצה, מספר השוערים בקבוצה, סה"כ השערים שהקבוצה הבקיעה, סה"כ כרטיסים שהקבוצה קיבלה ומספר המשחקים שהקבוצה שיחקה.
  + בנוסף לפרטים הבסיסיים, בקבוצה נחזיק שני עצי AVL – אחד של כל השחקנים באותה קבוצה מסודרים לפי מזהה שחקן, והשני של כל השחקנים באותה קבוצה מסודרים לפי כמות השערים שהבקיעו + הכרטיסים שקיבלו + מזהה השחקן (במקרה של שוויון).

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

**world\_cup\_t –**

נאתחל את ארבעת עצי הAVL, וגם אתחול מצביע נוסף לPlayer אשר מצביע על השחקן עם הכמות השערים הכי גדול. כמו כן, נאתחל משתנה שסופר את כל השחקנים המשתתפים בגביע העולם ל-0.

סיבוכיות זמן:

נאתחל את כל המשתנים ל-null או לערכים דיפולטים, ולכן סיבוכיות הזמן הוא O(1).

סיבוכיות מקום:

העצים בשלב זה ריקים, והמצביע לשחקן הוא nullptr. לכן סך סיבוכיות המקום הוא O(1).

**world\_cup\_t~ –**

נשחרר את הזיכרון עבור ארבעת עצי הAVL המרכזיים, העצים שנמצאים בתוך האיברים בעצים המרכזיים, המצביע הנוסף לשחקן שהבקיע הכי הרבה שערים, והמשתנה מטיפוס int שמחזיק את מספר השחקנים בכלל גביע העולם.

ראשית נשחרר את ה-data של כל השחקנים והקבוצות במערכת. נעשה זאת ע"י שחרור ה-data בעצים m\_teamsByID ו-m\_playersByID.

לאחר מכן, ה-destructor-ים הדיפולטים של העצים השונים יקראו, וישחררו את זיכרון ה-nodes של העצים. בנוסף, יקראו ה-destructor-ים הדיפולטים של שאר המשתנים של world\_cup\_t.

סיבוכיות זמן:

כדי לשחרר את ה-data, נעבר כל כל הקבוצות והשחקנים שקיימים במערכת אחד-אחד, ונשחרר את הזיכרון. זהו נעשה בסיבוכיות O(n+k) כאשר n הוא כמות השחקנים במערכת, ו-k הוא כמות הקבוצות במערכת.

לאחר מכן, יקראו ה-destructor-ים הדיפולטים של העצים השונים, אשר עוברים על כל ה-nodes במערכת:

* + יש לנו שני עצים של שחקנים, ולכן עבור n שחקנים, יש לכל עץ מקסימום n nodes. לכן סיבוכיות המקום של מעבר על העצים הללו הוא O(2n) = O(n).
  + יש לנו שני עצים של קבוצות, ולכן עבור k קבוצות, יש לכל עץ מקסימום k nodes. לכן סיבוכיות הזמן של מעבר על העצים הללו הוא O(k) O(2k) =.
    - בתוך כל קבוצה יש שני עצים של השחקנים המשחקים באותה קבוצה. סך כל השחקנים בעצים הללו בכל הקבוצות הוא 2n (n בכל סוג עץ) – זהו משום שכל שחקן משחק רק בקבוצה אחת.

לכן במעבר על כל הקבוצות ביחד, נעבר על עוד 2n nodes עבור כל עץ של קבוצות – סך הכל עוד 4n nodes.

לכן סיבוכיות הזמן סך הכל הוא O(2n+2k+4n)+O(n+k) = O(7n + 3k) = O(n+k).

סיבוכיות מקום:

בפעולה זו לא מקצים זיכרון נוסף ולכן אין הוספה לזיכרון של מבנה הנתונים, והוא נשאר בסיבוכיות של O(n+k).

**add\_team –**

ראשית נוודא את תקינות הקלט, ואם הוא אינו תקין אז נחזיר INVALID\_INPUT.

כמו כן, נוודא כי לא קיים כבר קבוצה עם המזהה הזה, אחרת נחזיר FAILURE.

נוסיף את הקבוצה לעץ AVL שמרכז את כל הקבוצות המסודרות לפי מזהה קבוצה - m\_teamsByID. נשים לב כי בהוספת קבוצה חדשה, מספר השחקנים בה הוא 0 ולכן הקבוצה לא מתווספת לעץ AVL של m\_qualifiedTeams.

בסוף נחזיר SUCCESS.

סיבוכיות זמן:

נעשה חיפוש לפי מזהה קבוצה בעץ AVL m\_teamsByID כדי להכניס את הקבוצה החדשה, בסיבוכיות של O(log(k)).

לאחר ההכנסה, נבצע גלגולים בעץ כדי לאזן אותו בחזרה, בסיבוכיות של O(1) כפי שנלמד בהרצאה.

לכן סך סיבוכיות הזמן של הפעולה היא O(log(k)).

סיבוכיות מקום:

בפעולה הזו אנו מקצים זכרון לעוד קבוצה אחת אשר מתווספת לעץ AVL קיים – ולכן סיבוכיות מקום הנוסף לקיים הוא O(1). לכן סיבוכיות המקום נשאר O(n+k).

**remove\_team –**

ראשית נוודא את תקינות הקלט, ואם הוא אינו תקין אז נחזיר INVALID\_INPUT.

נחפש בעץ AVL של הקבוצות המסודרות לפי מזהה קבוצה - m\_teamsByID, ונבדוק אם בקבוצה קיימים שחקנים. במקרה שיש שחקנים בקבוצה, או שהקבוצה בכלל לא קיימת, אז נחזיר FAILURE.

אחרת, נסיר את הקבוצה על ידי הסרת הnode בעץ AVL m\_teamsByID (נשים לב כי אם לקבוצה אין שחקנים, אז הוא לא ימצא בעץ m\_qualifiedTeams ולכן אין צורך להסיר אותו משם).

בסוף נחזיר SUCCESS.

סיבוכיות זמן:

בפעולה זו אנו מחפשים את הקבוצה המבוקשת פעמיים נפרדות: פעם אחת לבדוק אם קיימים שחקנים בקבוצה – חיפוש בעץ m\_teamsByID, ופעם שנייה כדי להסיר את הקבוצה מ-m\_teamsByID. השני המקרים, סיבוכיות הזמן של החיפוש הוא O(log(k)) כי יש מקסימום k קבוצות בעץ הזה.

לכן סך סיבוכיות הזמן הוא O(log(k)\*2) = O(log(k)).

סיבוכיות מקום:

בפעולה זו לא מקצים זיכרון נוסף ולכן אין הוספה לזיכרון של מבנה הנתונים, והוא נשאר בסיבוכיות של O(n+k).

**add\_player –**

ראשית נוודא את תקינות הקלט, ואם הוא אינו תקין אז נחזיר INVALID\_INPUT.

כמו כן, נוודא כי לא קיים כבר שחקן עם המזהה הזה וכי אכן קיימת קבוצה עם המזהה הניתן, אחרת נחזיר FAILURE.

ניצור את השחקן עם הנתונים מהקלט חוץ מכמות המשחקים שהשחקן שיחק – כמות המשחקים ששחקן שיחק מחושב לפי כמות המשחקים שהקבוצה שיחקה + כמות המשחקים שהשחקן שיחק ללא הקבוצה. כדי לא להוסיף אליו משחקים לא רלוונטיים (שהקבוצה שיחקה לפני שהשחקן הצטרף), אז נחסיר מכמות המשחקים של השחקן את כמות המשחקים שהקבוצה שיחקה לפני הצטרפותו.

נכניס אותו לשני עצי AVL הכלליים של השחקנים: אחד המסודר לפי מזהה קבוצה, והשני מסודר לפי ועץ שני המסודר לפי מספר השערים שהשחקן הבקיע, מספר הכרטיסים שקיבל ומזהה הקבוצה במקרה של שוויון. נעדכן את המצביע של השחקן עם הכי הרבה שערים בהתאם.

נוסיף את השחקן גם לקבוצה מתאימה:

בכל קבוצה יש שני עצי AVL של השחקנים של אותה הקבוצה: עץ אחד המסודר לפי מזהה הקבוצה, ועץ שני המסודר לפי מספר השערים שהשחקן הבקיע, מספר הכרטיסים שקיבל ומזהה הקבוצה במקרה של שוויון. לכן נוסיף את השחקן לשני העצים הללו.

בנוסף לכך, נעדכן פרטים של הקבוצה בהתאם לנתונים של השחקן החדש: נוסיף את כמות השערים של השחקן לכמות סה"כ שהבקיעה הקבוצה, ונוסיף את כמות הכרטיסים של השחקן לכמות סה"כ כרטיסים שקיבלה הקבוצה. כמו כן, נוסיף 1 לכמות השוערים של הקבוצה אם השחקן החדש יכול לשחק כשוער.

לאחר מכן, נוסיף את הקבוצה לm\_qualifiedTeams במקרה שכעת הקבוצה בעלת לפחות 11 שחקנים ויש לפחות שוער אחד.

בסוף, נוסיף 1 לכמות השחקנים שיש סה"כ בגביע העולם, ונחזיר SUCCESS.

סיבוכיות זמן:

בפעולה זו, מחפשים אם השחקן כבר קיים ומכניסים אותו לשני עצי AVL של שחקני המשחק בסיבוכיות של O(log(n)) כאשר n היא כמות השחקנים סה"כ בגביע העולם ובכן כמות השחקנים המקסימלית בכל אחד מהעצים הללו. כמו כן, מחפשים ומעדכנים את שני עצי AVL של הקבוצות במשחק בסיבוכיות של O(log(k)) כאשר k היא כמות הקבוצות סה"כ בגביע העולם, ובכן הכמות המקסימלית של קבוצות בשני העצים.

כמו כן, בקבוצה הספציפית של השחקן, מכניסים את השחקן לשני העצים של שחקני הקבוצה. בכל אחד מהעצים יש מקסימום n שחקנים, ולכן סיבוכיות החיפוש וההכנסה של כל אחד מהעצים הוא O(log(n)). מאזנים את כל אחד מהעצים הללו בעזרת גלגולים בשיטה שנלמדה בהרצאה, בסיבוכיות O(1).

גם מתעדכן המצביע לשחקן שהבקיע הכי הרבה שערים – גם של המשחק כולו וגם של הקבוצה. פעולה זו דורשת חיפוש בעצים של השחקנים עם מקסימום n שחקנים ובכן בסיבוכיות של O(log(n)).

לכן סך הכל סיבוכיות הזמן הוא: O(log(n)\*6 + log(k)\*2) = O(log(n)+log(k)).

סיבוכיות מקום:

בפעולה הזו אנו מקצים זכרון לעוד שחקן אחד אשר מתווסף ל-4 עצי AVL קיימים של שחקנים (שניים כללים, שניים של הקבוצה של השחקן), וגם במקרה הגרוע מוסיפים את הקבוצה של השחקן לעוד עץ AVL של m\_qualifiedTeams– ולכן סיבוכיות מקום הנוסף לקיים הוא O(5) = O(1). לכן סיבוכיות המקום נשאר O(n+k).

**remove\_player-**

**update\_player\_stats –**

**play\_match–**

**get\_num\_played\_games –**

בפעולה זו, נחזיר את כמות המשחקים שהשחקן שיחק סה"כ.

ראשית נבדוק את הקלט, ונחזיר INVALID\_INPUT אם התקבל מזהה שחקן שלילי. לאחר מכן, נחפש את השחקן בעץ כל השחקנים לפי מזהה שחקן (m\_playersByID). אם לא קיים שחקן כזה, נחזיר FAILURE. אחרת, ניגש דרך השחקן אל הקבוצה שבה הוא משחק (יש באובייקט השחקן מצביע ישירות לקבוצה). נשלוף מהקבוצה את כמות המשחקים שהקבוצה שיחקה סך הכל, ונוסיף את ערך זה לכמות המשחקים שהשחקן שיחק. נחזיר את הסכום שהתקבל.

סיבוכיות הזמן:

בפעולה זו, אנו מחפשים בעץ כל השחקנים בשיטה שנלמדה בהרצאה בסיבוכיות O(log(n)) כאשר n הוא כמות השחקנים בכלל המערכת. בנוסף לכך, אנו ניגשים ישירות לקבוצה בה השחקן משחק דרך המצביע בשחקן – בסיבוכיות O(1). לכן סך הכל, סיבוכיות הזמן הוא O(log(n)) + O(1) = O(log(n)).

סיבוכיות המקום:

בפעולה זו, אנו משתמשים במשתנה מקומי נוסף מטיפוס int, ולכן מוסיף זיכרון של O(1). בנוסף, שומרים בצד מצביע לאותו שחקן הנדרש, אשר מוסיף סיבוכיות מקום של O(1). לכן סיבוכיות המקום נשאר O(n+k).

**get\_team\_points–**

בפעולה זו, נחזיר את כמות הנקודות השקבוצה צברה סה"כ.

ראשית נבדוק את הקלט, ונחזיר INVALID\_INPUT אם התקבל מזהה קבוצה שלילי. לאחר מכן, נחפש את הקבוצה בעץ כל הקבוצות לפי מזהה קבוצה (m\_teamsByID). אם לא קיימת קבוצה כזו, נחזיר FAILURE. אחרת, נחזיר את כמות הנקודות של אותה קבוצה.

סיבוכיות הזמן:

בפעולה זו, אנו מחפשים בעץ כל הקבוצות בשיטה שנלמדה בהרצאה בסיבוכיות O(log(k)) כאשר k הוא כמות הקבוצות בכלל המערכת. בקבוצה, אנו ניגשים ישירות למשתנה מטיפוס int אשר הוא member של מחלקת Team, ולכן בסיבוכיות הזמן של O(1). לכן סך הכל, סיבוכיות הזמן הוא O(log(k)) + O(1) = O(log(k)).

סיבוכיות המקום:

בפעולה זו, שומרים בצד מצביע לאותה קבוצה הנדרשת, אשר מוסיף סיבוכיות מקום של O(1). לכן סיבוכיות המקום נשאר O(n+k).

**unite\_teams –**

**get\_top\_scorer –**

בפעולה זו, נחזיר את מזהה השחקן עם ה-score הכי גבוה.

ה-score של השחקן נקבע לפי הפרמטרים הבאים: השחקן עם כמות השערים הגבוה ביותר ששחקן הבקיע. במקרה של שוויון, אז השחקן עם כמות הקלפים הנמוך ביותר. במקרה של שוויון, אז השחקן עם מזהה הקבוצה הכי גבוה.

ראשית נבדוק את הקלט, ונחזיר INVALID\_INPUT אם התקבל מזהה קבוצה השווה ל-0. כמו כן, אם אין שחקנים כלל המשתתפים בגביע העולם, נחזיר FAILURE.

* אם התקבל מזהה קבוצה הגדולה מ-0, אז נחפש את הקבוצה בעץ כל הקבוצות לפי מזהה קבוצה (m\_teamsByID). אם לא קיימת קבוצה כזו או אם בקבוצה זו אין שחקנים אז נחזיר FAILURE. אחרת, נחזיר את מזהה השחקן שמשחק בקבוצה זו עם ה-score הכי גבוה מתוך כלל השחקנים במשחקים בקבוצה זו.
* אחרת, התקבל מזהה קבוצה הקטנה מ-0, ולכן נחזיר את מזהה השחקן עם ה-score הכי גבוה מתוך כלל השחקנים בגביע העולם.

סיבוכיות הזמן:

בפעולה זו, במקרה שמזהה הקבוצה גדולה מ-0, אז מחפשים בעץ כל הקבוצות בשיטה שנלמדה בהרצאה בסיבוכיות O(log(k)) כאשר k הוא כמות הקבוצות בכלל המערכת. בקבוצה, ניגשים ישירות למצביע של השחקן עם ה-score הגבוה ביותר בין השחקנים באותה קבוצה, אשר הוא member של מחלקת Team, ולכן בסיבוכיות הזמן של O(1). לכן סך הכל, סיבוכיות הזמן הוא O(log(k)) + O(1) = O(log(k)).

במקרה שמזהה הקבוצה קטנה מ-0, אז ניגש ישירות למצביע לשחקן עם ה-score הגבוה ביותר בכלל גביע העולם, אשר הוא member של מחלקת world\_cup\_t. לכן סיבוכיות הזמן סך הכל הוא O(1).

סיבוכיות המקום:

בפעולה זו, שומרים בצד מצביע לאותה קבוצה הנדרשת, אשר מוסיף סיבוכיות מקום של O(1). לכן סיבוכיות המקום נשאר O(n+k).

**get\_all\_players\_count -**

בפעולה זו, נחזיר את מספר השחקנים בקבוצה או בכלל גביע העולם.

ראשית נבדוק את הקלט, ונחזיר INVALID\_INPUT אם התקבל מזהה קבוצה השווה ל-0.

* אם התקבל מזהה קבוצה הגדולה מ-0, אז נחפש את הקבוצה בעץ כל הקבוצות לפי מזהה קבוצה (m\_teamsByID). אם לא קיימת קבוצה כזו אז FAILURE. אחרת, נחזיר כמות השחקנים במשחקים באותה קבוצה.
* אחרת, התקבל מזהה קבוצה הקטנה מ-0, ולכן נחזיר את כמות השחקנים בכלל גביע העולם.

סיבוכיות הזמן:

בפעולה זו, במקרה שמזהה הקבוצה גדולה מ-0, אז מחפשים בעץ כל הקבוצות בשיטה שנלמדה בהרצאה בסיבוכיות O(log(k)) כאשר k הוא כמות הקבוצות בכלל המערכת. בקבוצה, ניגשים ישר למשתנה מטיפוס int שמחזיק את כמות השחקנים באותה קבוצה, אשר הוא member של מחלקת Team, ולכן בסיבוכיות הזמן של O(1). לכן סך הכל, סיבוכיות הזמן הוא O(log(k)) + O(1) = O(log(k)).

במקרה שמזהה הקבוצה קטנה מ-0, אז ניגש ישירות למשתנה מטיפוס int שמחזיק את כמות השחקנים בכלל גביע העולם, אשר הוא member של מחלקת world\_cup\_t. לכן סיבוכיות הזמן סך הכל הוא O(1).

סיבוכיות המקום:

בפעולה זו, שומרים בצד מצביע לאותה קבוצה הנדרשת, אשר מוסיף סיבוכיות מקום של O(1). לכן סיבוכיות המקום נשאר O(n+k).

**get\_all\_players -**

**get\_closest\_player -**

**knockout\_winner -**