Dry

מבנה הנתונים שלנו מכיל 3 עצי AVL ראשיים המכילים את המידע הבא :

1. allCountries : עץ AVL המכיל צמתים (nodes) מסוג Country (מדינות) שממוינים בתוכי לפי country Id .
2. allContestants : עץ AVL המכיל צמתים (nodes) מסוג Contestant (מתחרים)הממוינים לפי Contestant Id.
3. allTeams : עץ AVL המכיל צמתים (nodes) מסוג Team (נבחרות) הממוינים לפי team id.

הסבר על כל class שמימשנו והשתמשנו בו לאורך התרגיל :

subClasses :

intObject : שיהיה משתנה עזר לשימוש בעץ AVL הגנרי, מכיל רק שדה מסוג int ומתנהג כמו int.

ModifyKey : אובייקט שיכיל שני שדות : 1.contestant id מזהה מתחרה.

2.strength : כוח של מתחרה בעל id הנ"ל.

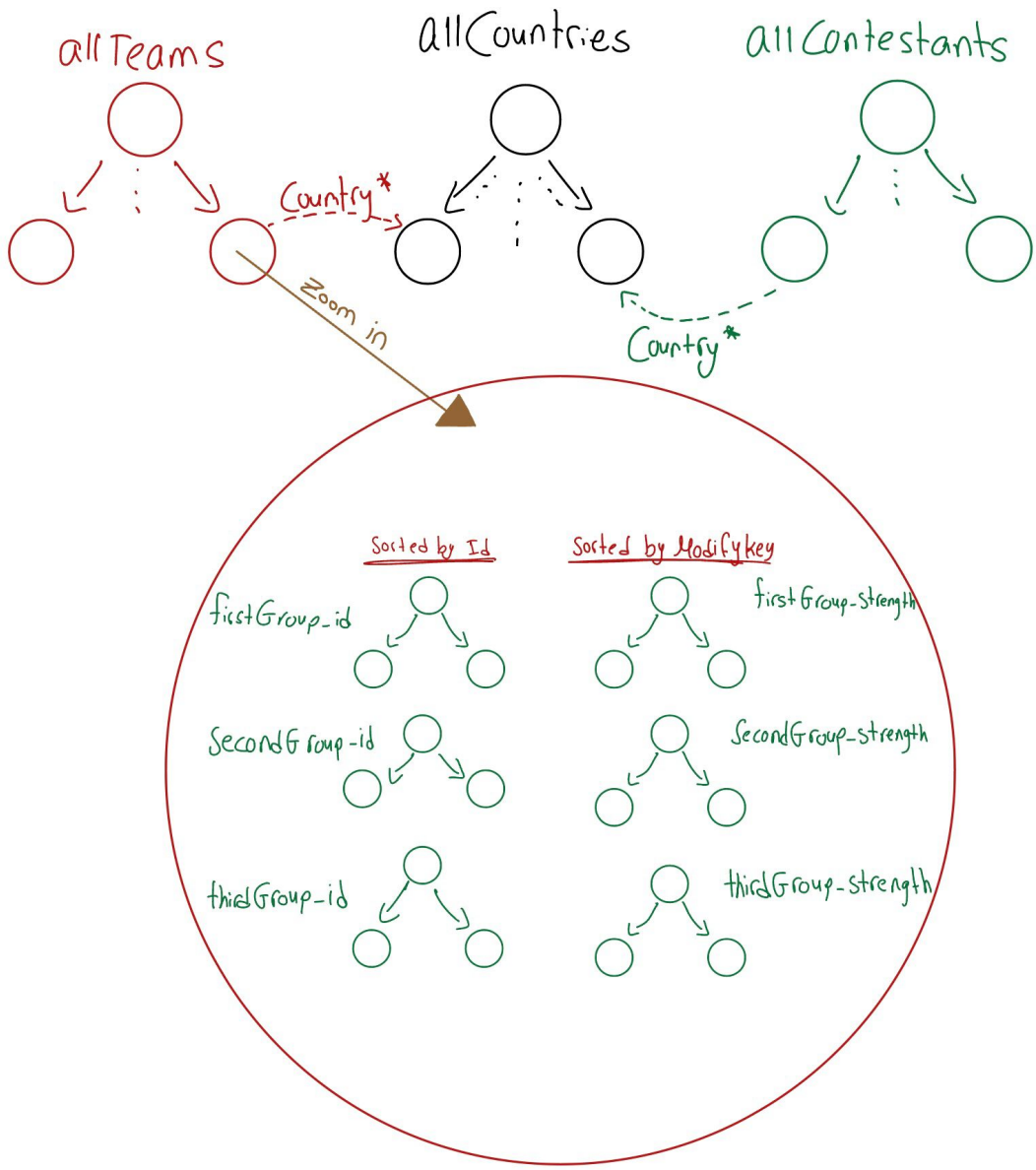
המטרה ממנו היא למיין את המתחרים בעץ לפי strength כעדיפות ראשונה ובמקרה של שוויון כוחות אז ממיינים לפי id.

main classes :

1. Country: מכילה מידע על מדינה כלשהיא כאשר השדות שלה יהיו :
   * countryId
   * medals
   * numOfContestants : מספר מתחרים ששייכים למדינה זו.
   * numOfTeams : מספר נבחרות ששייכות למדינה זו.

2. Contestant: מכיל מידע על מתחרה כלשהוא במערכת כאשר נשמור לו על המידע הבא:
   *contestId
   (country) : מצביע למדינה בעץ allCountries שאליה המתחרה שייך.
   *sport
   *strength : כוח המתחרה
   (teams): מערך בגודל 3 שישמור בתוכו מספרים מזהים(teamId) לנבחרות שאליהם המתחרה שייך.
   *modifyKey : מסוג ModifyKey שישמור בתוכו את contestId ו strength של המתחרה.
   *objectId : מסוג intObject שישמור contestId.

3. Team : מכיל מידע על כל נבחרת שקיימת במערכת, כאשר לכל נבחרת נשמור את המידע הבא :
   * teamId
   * sport
   * teamStrength .
   * maxTeamStrength : שומר מהי כוח הנבחרת המקסימלית שיכולה להיות אחרי העפה של 3 מתחרים כלשהם מהנבחרת.
   (country) : מצביע למדינה בעץ allCountries שאליה הנבחרת שייכת.
   * firstGroup_id: עץ AVL ששומר בתוכו את השליש הראשון של קבוצת המתחרים השייכים לנבחרת הממוינים לפי contestId.
   *secondGroup_id : עץ AVL ששומר בתוכו את השליש השני של המתחרים הממוינים לפי contestId.
   * thirdGroup_id : עץ AVL ששומר בתוכו את השליש האחרון של המתחרים הממוינים לפי contestId.
   *firstGroup_strength: עץ AVL ששומר בתוכו את השליש הראשון של קבוצת המתחרים השייכים לנבחרת (אותם מתחרים ב firstGroup_id) אבל ממוינים לפי ModifyKey.
   *secondGroup_strength : עץ AVL ששומר בתוכו אותו את המתחרים ב secondGroup_id אך ממוינים לפי ModifyKey .
   * thirdGroup_strength : עץ AVL ששומר בתוכו את המתחרים שב thirdGroup_id אך ממוינים לפי ModifyKey.



***כלומר בהינתן שיש לנו n מתחרים ב Team נשים ב firstGroup_id ה $\frac{n}{3}$ מתחרים הכי קטנים, וב firstGroup_strength נמיין אותם לפי ModifyKey כך שנוכל לדעת מי המתחרה בעל הכוח המקסימלי מתוך ה $\frac{n}{3}$ מתחרים הכי קטנים. באותו אופן עבור שאר העצים.

<u>סיכום קצר על פונקציות חשובות בעץ AVL שלנו :</u>

<u>getMax,getSize</u> מחזירות את המקסימום ואת הגודל בעץ שהום שדות בו, ולכם זה O(1).

<u>insert,remove,find,getMin</u> מתרחשות ב O(log n) כאשר n מספר צמתים בעץ.(כפי שנלמד בהרצאה).

<u>treeClear</u> מוחקת את כל הצמתים ולכן זה O(n).

TreeKey_to_array, TreeData_to_array,array_to_tree : לפי מה שנלמד בהרצאה ותרגול O(n).

<u>פונקציות עזר במחלקה Team :</u>

במקרה הגרוע :

k- מספר מדינות במערכת.

m- מספר נבחרות.

n- מספר מתחרים.

כדי שנוכל לעמוד בסיבוכיות השתמשנו במספר פונקציות עזר שבעזרתם נעשה כל מני חישובים ופעולות תוך כדי הכנסת והוצאת מתחרים מהנבחרות, וגם כדי לשמור על שלושה עצים שתמיד יהיו מחולקים כמו שצריך כלומר בכל הוצאה או הכנסה של מתחרה לנבחרת מסדרים את העצים כך שתמיד יהיו באופן הבא :

firstGroup_id יכיל את $\frac{n}{3}$ מתחרים בעלי מזהים קטנים ביותר, secondGroup_id יכיל את ה $\frac{n}{3}$ האמצעיים ואת האחרוניים הגדולים ביותר יהיו ב thirdGroup_id.

1. **updateSize**: בכל הוצאה או הכנסה של מתחרים לעצים בתוך הנבחרת אנו מעדכנים את ה numOfContestants שלה דרך חיבור הsize של כל עצי ה id, כיוון ש getSize של AVL שלנו לוקחת O(1) אז סיבוכיות הזמן בפעולה זו היא O(1).

2. **modifyStrength** : מטרתה היא לעדכן את ה teamStrength בכל הוצאה או הכנסה שח מתחרה לנבחרת,בכל פעם היא מחברת את ה Max של כל עץ משלושת העצים הממוינים לפי ModifyKey ומעדכנת את הכוח להנבחרת. getMax מחזירה ה max של ה AVL ל3 העצים לכן זה O(1).

3. **modifyMax Strength:** מטרתה היא לעדכן את maxTeamStrength בכל הוצאה או הכנסה של מתחרה לנבחרת,כדי לעשות את זה חילקנו לכל 9 המקרים של הוצאת 3 מתחרים כל שהם מהנבחרת,בנוסף למקרה שבו ה maxTeamStrength לא משתנה(הוצאת ה min מכל עץ מהשלושה),השתמשנו בפונקציות הללו :

3.1) **removeThreeFromFirstTree**
3.2) **removeThreeFromSecondTree**
3.3) **removeThreeFromThirdTree**
3.4) **removeTwoFromFirstTreeAndOneFromSecondTree**
3.5) **removeTwoFromFirstTreeAndOneFromThirdTree**
3.6) **removeTwoFromSecondTreeAndOneFromThirdTree**
3.7) **removeTwoFromSecondTreeAndOneFromFirstTree**
3.8) **removeTwoFromThirdTreeAndOneFromSecondTree**
3.9) **removeTwoFromThirdTreeAndOneFromFirstTree**

כל אחת מהפונקציות הנ"ל מוציאה 3 מתחרים בהתאם למקרה,מסדרת את העץ דרך insert,remove מספר קבוע של פעמים וזה O(log n) ומחשבת את teamStrength במקרה זה דרך ה max של כל עץ וזה כמו שציינתי O(1), אחר כך מחזירה את העץ למצב המקורי באותו אופן ומחזירה את התוצאה ⇐ נקבל כי כל אחת מהפונקציות הנ"ל מתעסקת בהכנסה והוצאה ל AVL של מתחרים מספר קבוע של פעמים לכן זה O(log n) .
בסוף ניקח את המקסימום מבין התוצאות ונעדכן את maxTeamStrength ולכן זה : $9*Olog(n) = O(log\ n)$.

***פונקציות שמטרתן הכנסת מתחרה חדש לנבחרת:

4. **addContestant:** יש לנו מתחרה שאותו רוצים להכניס לנבחרת,בפעולה זו חילקנו למקרים :

2.1) אם יש לנו בנבחרת כרגע פחות מ3 מתחרים אז בודקים חלוקת המתחרים בהם ומסדרים אותם ידני מחדש בהתאם ל מזהה של המתחרה החדש,נעשה פעולות של הוצאה והכנסה לעצים(שמכילים Contestants) כדי לסדר אותם מספר קבוע של פעמים לכן במקרה זה סיבוכיות הזמן תהיה O(log n) במקרה הגרוע כאשר כל הכנסה או הוצאה למתחרה מתרחשת ב O(logn).

2.2)אם יש לנו כבר 3 מתחרים ומעלה אז נשתמש בפונקציית העזר **addContestantAux(Contestant&)** ש תכף נראה שגם היא O(log n) , סה"כ פעולה זו מתעסקת בהכנסה והוצאה של מתחרים מעצי AVL מספר קבוע של פעמים, ועושה modifyStrength,modifyTeamStrength שגם הם O(log n) ולכן היא O(log n) .

5. **addContestantAux:** פעולה זו תכניס את המתחרה לעץ המתאים במבחינת גודל המזהה contestId, (משתמשים ב getMax בAVL שלנו שלוקחת O(1))ואז תקרא לפעולה balanceTreesAfterAdding(),סה"כ הכנסת המתחרה לעץ היא O(log n) במקרה הגרוע ו balanceTreesAfterAdding() שתכף נראה שהיא O(log n) נקבל ששימוש בפעולה זו מתרחש ב O(log n) זמן.

6. **balanceTreesAfterAdding** : אחרי הכנסת המתחרה החדש,פונקצייה זו מטרתה היא לסדר את המתחרים בעצים לפי הסדר שהגדרנו,היא מתעסקת בהכנסה והוצאה של מתחרים ב 6 עצי AVL הנמצאים בנבחרת,דרך חילוק למקרים וקריאה ל פונקצייה המתאימה למקרה מבין הפונקציות:

6.1) **numModulo3is1AfterAdding**
6.2) **numModulo3is2AfterAdding**
6.3) **numModulo3is0AfterAdding**
כל אחת מהפונקציות הללו עושה פעולות הכנסה והוצאה של מתחרים ב 6 עצי ה AVL מספר קבוע של פעמים שכל אחת מהם מתרחשת ב O(log n), ולכן סה"כ סיבוכיות הזמן של הפונקצייה הזאת היא O(log n).

***פונקציות שמטרתן הוצאת מתחרה מנבחרת:

7. **removeContestant(Contestant&)** : בהינתן מתחרה,רוצים להוציא אותו מהנבחרת:
קודם כל נחפש את מספרה של הנבחרת במערך ה teams שלו ונשנה אותו ל0 , זה O(1) כי אורך המערך קבוע.
אחר כך נחסיר 1 מ numOf Contestats של הנבחרת וזה O(1), ואז נקרא ל
7.1) **removeContestantAux** שבהתאם למספר המתחרים המצופה להיות בנבחרת אחרי מחיקת המתחרה היא קוראת לאחת מהפונקיות הללו בהתאם למקרה:
7.2)**numModulo3is0AfterDeleting**
7.2)**numModulo3is1AfterDeleting**
7.3)**numModulo3is2AfterDeleting**
כל אחת משלושת הפונקציות הנ"ל מחפשת את המתחרה ב O(log n) ומוחקת אותו ומסדרת את העצים לפי הסדר שהגדרנו ע"י insert,remove שמשתמשת בהם מספר קבוע של פעמים לכן סיבוכיות הזמן של כל אחת מהם היא O(log n).
נקבל ש *removeContestantAux* מתרחשת ב O(log n) .
⇐ סיבוכיות הזמן הכוללת של *removeContestant* גם היא ב סה"כ O(log n).

פונקציות : Functions

במקרה הגרוע :
k- מספר מדינות במערכת.
m- מספר נבחרות
n- מספר מתחרים.

1) **olympics()** : נאתחל ארבע עצי AVL ריקים ⇐ סיבוכיות זמן $O(1)$.

2) **add_country()** : בודקים קודם אם הנתונים חוקיים,זה O(1), נבדוק גם אם המדינה כבר קיימת (בעץ allCountries) וזה O(logk) אם כן נחזיר FAILURE, אחרת ניצור מדינה וזה O(1) ונוסיף אותה למערכת, כלומר נכניס צומת חדשה לעץ ה AVL $allCountries$ ⇐ סיבוכיות זמן $O(log\ k)$.

3) **remove_country** : בודקים אם מדינה כזו נמצאת בעץ ה(allCountries) AVL וזה $O(log\ k)$,אם אינה נמצאת מחזירים FAILURE,אחרת נוציא את המדינה מהמערכת אם אין מתחרים או נבחרות השייכים אליה,נבדוק את זה דרך המשתנים שלה numOfContestants,numOfTeams וזה $O(1)$, ואז הוצאת צומת מ AVL של allCountries זה $O(log\ k)$.

4) **add_team** : רוצים להכניס נבחרת להמערכת, בודקים :
4.1) אם הפרמטרים חוקיים ⇐ $O(1)$.
4.2) אם הנבחרת כבר קיימת במערכת כלומר ב allTeams נחזיר FAILURE וכל זה $O(log\ m)$.
4.3) אם המדינה המצורפת אינה נמצאת ב allCountries נחזיר FAILURE ⇐ $O(log\ k)$, אחרת נחזיק את הצומת.
אם הכל היה תקין ניצור נבחרת עם אתחול של 6 עצי AVL ריקים ומצביע למדינה שהחזקנו ((O(1)),נוסיף את הנבחרת (הוספת צומת חדש) ל allTeams וזה $O(log\ m)$.
סה"כ נקבל : $O(log\ m\ + log\ k\ )\ =\ O(2log\ m + log\ k)$.

5) **remove_team** : רוצים להוציא את הנבחרת מהמערכת, בודקים :
5.1) תקינות פרמטרים ⇐ $O(1)$.
5.2) אם נבחרת זו לא נמצאת ב allTeams, וזה $O(log\ m)$ נחזיר FAILURE, אחרת נחזיק אותה.

5.3) אם בנבחרת קיימים מתחרים, בודקים את numOfContestants $\Leftarrow$ $O(1)$. מחזירים FAILURE.
אם הכל תקין נחסיר 1 מ numOfTeams דרך המצביע למדינה שאליה שייכת הנבחרת $\Leftarrow$ $O(1)$.
ונוציא את הנבחרת (הוצאת צומת) מ allTeams וזה O(log m) .
סה"כ סיבוכיות זמן היא : $O(log\ k\ + log\ m)$.

6) **add_contestant** : הוספת מתחרה למערכת עבור מדינה מסוימת, קודם בודקים :
6.1) תקינות פרמטרים : O(1)
6.2) אם המתחרה כבר נמצא ב allContestans נחזיר FAILURE זה $O(log\ n)$.
6.3) אם המדינה לא קיימת ב allCountries נחזיר FAILURE אחרת נחזיק מצביע לה O(log k).
אם הכל תקין ניצור מתחרה חדש ונוסיף אותו ל allContestants וזה O(log n), ונוסיף 1 ל numOfContestants של המדינה שאליה הוא שייך דרך המצביע.
סה"כ סיבוכיות זמן : $O(log\ n + logk\ )$.

7) **remove_contestant**: הוצאת מתחרה ממבנה הנתונים, נבדוק קודם :
7.1) תקינית קלט O(1) , מחזירים INVALID_INPUT במקרה שהקלט לא חוקי.
7.2) נבדוק אם מתחרה זה קיים במערכת דרך חיפוש של מתחרה זה ב עץ allContestants וזה $O(log\ n)$, במקרה שהיה לא קיים מחזירים FAILURE אחרת נחזיק אותו.
7.3) נבדוק אם מתחרה זה שייך לנבחרת כלשהיא: עוברים על המערך שבאורך קבוע 3 דרך פונקציית העזר howManyTeams , אם מצאנו מספר שהוא
גדול מ 0 זאת אומרת שהוא שייך לנבחרת זו ואז מחזירים FAILURE,אחרת נמשיך. זאת O(1) כי סיור במערך מספר קבוע של פעמים שהוא גם באורך קבוע זה O(1).
אם הכל היה תקין אז ניגש למדינה שאליה המתחרה שייך דרך המצביע שנמצא עם המתחרה ונחסיר 1 מ numOfCOntestants שלה. זה לוקח O(1).
<u>סה"כ</u>: חיפוש צומת ב AVL של allContestants מספר קבוע של פעמים ושינוי משתנים , סיבוכיות הזמן במקרה הגרוע היא $O(log\ n)$.

8) **add_contestant_to_team:** רוצים להכניס מתחרה לנבחרת נתונה, נבדוק קודם :
8.1) תקינות קלט O(1) אחרת מחזירים INVALID_INPUT.
8.2) אם הנבחרת אינה ב allTeams או המתחרה אינו ב allContestants מחזירים FAILURE זה O(log n + log m), אחרת נחזיק אותם.
8.3) כעת נבדוק אם הם לא שייכים לאותה מדינה דרך המצביע למדינה אליה המתחרה והנבחרת שייכים, זה O(1), נבדוק גם אי-שוויון Sport, גם O(1), נבדוק אם יש ב teams של המתחרה מספר הנבחרת הזו O(1),ואם יש מקום ריק ב teams, סה"כ O(1).אם אם התקיים אחד מאלה מחזירים FAILURE.
אם הכל היה תקין נוסיף מספר הנבחרת ל teams במקום פנוי O(1), אוז נקרא ל addContestant של הנבחרת שלוקחת O(log n).
סה""כ סיבוכיות הזמן של הפונקצייה היא O(log n+log m).

9) **remove_contestant_from_team** : רוצים להוציא מתחרה מהנבחרת הנתונה, נבדוק :
9.1) תקינות קלט O(1) אחרת מחזירים INVALID_INPUT.
9.2) נחפש ב allContestants את המתחרה וב allTeams את הנבחרת, אם אחד לא קיים נחזיר FAILURE אחרת נחזיק אותם .וזה O(log n+ log m).
9.3)נבדוק אם המתחרה כן קיים בנבחרת דרך חיפוש מספר הנבחרת ב teams שלו וזה כמו שציינו קודם O(1) ,אם לא קיים נחזיר FAILURE.
אם הכל היה תקין נקרא לפונקצייה removeContestant ב team וניתן לה את המתחרה, שסיבוכיות הזמן שלה לפי מה שהראינו היא O(log n).
סה"כ הסיבוכיות של remove_contestant_from_team מתרחשת ב O(log n +log m).

10) **update_contestant_strength :** עדכון כוח של מתחרה קיים, נבדוק :
10.1) תקינות קלט O(1) אחרת מחזירים INVALID_INPUT
10.2) בודקים אם מתחרה זה נמצא ב allContestants, אם לא קיים או שהשינוי בכוח אינו חוקי מחזירים FAILURE אחרת נחזיק אותו , סיבוכיות זמן O(log n).

אם הכל תקין, נבצע את הפעולות הללו:
עוברים על מערך teams של המתחרה((O(1),עבור כל נבחרת שהוא נמצא בה,נוציא אותו ממנה בעזרת remove_contestant_from_team , שהיא בסיבוכיות O(log n+log m) .
כעת מעדכנים את הכוח של המתחרה ב allContestant וזה O(1) ומכניסים אותו שוב(מעודכן) לנבחרת דרך add_contestant_to_team שכבר הראנו שהיא O(log n+ log m) .
סה"כ סיבוכיות הזמן של הפונקצייה תהיה O(logn +log m).

11) **get_strength** : החזרת כוח של המתחרה הנתון :
בודקים תקינות קלט שזה O(1) אחרת מחזירים INVALID_INPUT.
בודקים אם המתחרה אינו נמצא ב allContestants שזה O(log n) מחזירים FAILURE אחרת מחזירים את הכוח שלו שזה O(1).
סיבוכיות הזמן בסוף תהיה O(log n).

12) **get_medals** : החזרת כמות המידליות למדינה הנתונה:
בודקים תקינות קלט שזה O(1) אחרת מחזירים INVALID_INPUT.
מחפשים את המדינה ב allCountries אם לא מצאנו מחזירים FAILURE אחרת נחזיר את medals שלה.
חיפוש של צומת ב AVL זה O(log k), לכן הסיבוכיות סה"כ במקרה הגרוע היא O(log k).

13) **get_team_strength** : החזרת הכוח של נבחרת:
בודקים אם teamId⩽0 מחזירים INVALID_INPUT וזה O(1).
נחפש את הנבחרת בתוך allTeams שזה O(log m) , אם לא מצאנו אותה מחזירים FAILURE אחרת נחזיר את teamStrength שלה עם SUCCESS.
נשים לב שלאורך כל התרגיל היינו מעדכנים את teamStrength תוך כדי הוצאת והכנסת מתחרים לנבחרת.
סה"כ סיבוכיות הזמן במקרה הגרוע היא O(log m).

14) **unite_teams** : רוצים לאחד שתי הנבחרות הנתונות לאחת, בודקים :
14.1) תקינות אופרטורים O(1) אחרת מחזירים INVALID_INPUT.
14.2) מחפשים את שתי הנבחרות ב allTeams שזה O(log m), אם אחת לא נמצאת, או שקיים אי-שוויון של country או sport שלהם (בדיקת זה O(1)), מחזירים Failure, אחרת מחזיקים אותם.

כעת מעבירים את כל המתחרים בשלושת עצי ה id של הנבחרת הראשונה למערך גדול ממוין בסיור inorder דרך TreeKey_to_array ו TreeData_to_array, שזה O(n_{team1}) במקרה הגרוע.
באותו אופן עבור הנבחרת השנייה,זה $O(n_{team2})$.
נמזג את שני ה מערכים של Key למערך אחד גדול ממוין בשיטת merge שזה $O(n_{team1} + n_{team2})$ ואת השניים של Data באותו אופן. $\Leftarrow$ $O(n_{team1} + n_{team2})$.
נחלק את שני המערכים הגדולים כל אחד לשלושה מערכים ממוינים לפי הסדר שהגדרנו ומעבירים כל אחד לעץ המתאים ב team1 דרך array_to_tree שזה $O(n_{team1} + n_{team2})$.
באותו אופן נעביר את שלושת העצים של ModifyKey(הממוינים לפי strength כעדיפות ראשונה) בכל נבחרת כל עץ למערך קטן, נמזג כל שלושה מערכים לכל נבחרת ונמיין אותם במערך אחד, ואז נמזג את שני המערכים הגדולים של כל נבחרת למערך אחד גדול, שעכשיו הוא מכיל את כל המתחרים הממוינים לפי ModifyKey(אחד עם key ואחד Data).
עכשיו נגדיר שלושה מערכים בגדלי עצי id החדשים, נעבור על המערך Key של האיחוד ונעביר כל מתחרה למערך המתאים לעץ id שאליו המתחרה שייך.
בסוף נעביר את המערכים לעצי ה ModifyKey המתאימים בעזרת array_to_tree.
כעת נעשה remove_team לנבחרת השנייה שלוקח O(log m).
נעדכן את teamStrength, maxTeamStrength,numOfCOntestants בנבחרת ראשונה שזה O(1).
לאורך כל הפונקצייה השתמשנו בחיפוש והסרה בעלי (log m) ,בסיור במערכים בגדלים חוקיים ובמספר קבוע של פעמים, השתמתשנו ב :
TreeKey_to_array, TreeData_to_array,array_to_tree שגם היו בסיבוכיות מתאימה,לכן סה"כ הסיבוכיות בפונקצייה זו היא : $O(log\ m\ + n_{team1} + n_{team2})$ במקרה הגרוע כאשר n_{team1} הוא מספר המתחרים בנבחרת ראשונה, ו n_{team2} הוא מספר המתחרים בנבחרת שנייה.

15) **play_match** : שתי נבחרות הנתונות משחקות אחת מול השנייה , נבדוק :

15.1) אם הפרמטרים חוקיים וזה O(1) ,אחרת נחזיר INVALID_INPUT.

15.2)נחפש את שתי הנבחרות ב allTeams, זה O(log m) לכל אחת לכן O(log m), במקרה ש אחת לא נמצאת, או במקרה של אי-שוויון sport (בדיקת זה O(1)) , מחזירים FAILURE, אחרת מחזיקים את שתי הנבחרות.

כעת מחשבים תוצאות : ניגשים למדינה של כל נבחרת דרך המצביע ולוקחים medals, כיוון שעשינו את זה דרך מצביע ללא חיפוש אז זה O(1),ואז מחברים לכל אחת גם את teamStrength של הנבחרת דרך getTeamStrength שהיא O(log m), במקרה של אי-שוויון תוצאות ניגשים למדינה של הנבחרת המנצחת דרך allContries שזה O(log k) ומוסיפים 1 ל medals שזה O(1) .

הסיבוכיות סה"כ במקרה הגרוע תהיה O(log k+log m).

16) **austerity_measures** : מחזירים את כוח הנבחרת המקסימלית שיכולה להיות אחרי העפה של 3 מתחרים כלשהם מהנבחרת.

בודקים אם teamId ⩽0, זה O(1), אם כן מחזירים INVALID_INPUT.

מחפשים את הנבחרת ב allTeams, זה O(log m),אם לא נמצאת או שיש בה פחות מ 3 מתחרים(בודקים numOfContestants ב O(1)) מחזירים FAILURE, אחרת מחפשים אותה ומחזירים את maxTeamStrength.

נשים לב שלאורך כל התרגיל היינו מעדכנים את maxTeamStrength תוך כדי הוצאת והכנסת מתחרים לנבחרת.

סה"כ סיבוכיות הזמן במקרה הגרוע היא O(log m).

17) **~olympics_t** : ההורס קורא להורס של AVL לכל עץ משלושת העצים הראשיים שבתורו קורא ל treeClear.

בסופו של דבר אנו עוברים על כל הצמתים שנמצאים במערכת , כך שלגבי Contestants יש לנו לכל מתחרה לכל היותר 4 צמתים שהוא מספר קבוע (1 ב allContestants ו 3 ב allTeams לכל היותר) , לכן סיבוכיות הזמן תהיה O(n + k+ m) במקרה הגרוע.

<u>סיבוכיות מקום :</u>

סה"כ החזקנו **במקרה הגרוע** :

4 צמתים סה"כ לכל מתחרה מבין n המתחרים(1 ב allContestants ו 3 ב allTeams לכל היותר בכל נבחרת שהוא נמצא בה) לכן זה O(4n)=O(n).

m צמתים עבור allTeams שיש בה במקרה הגרוע m נבחרות וזה O(m)

k צמתים עבור allCountries שיש בה במקרה הגרוע k מדינות וזה O(k).

סה"כ סיבוכיות מקום : O(n+k+m).