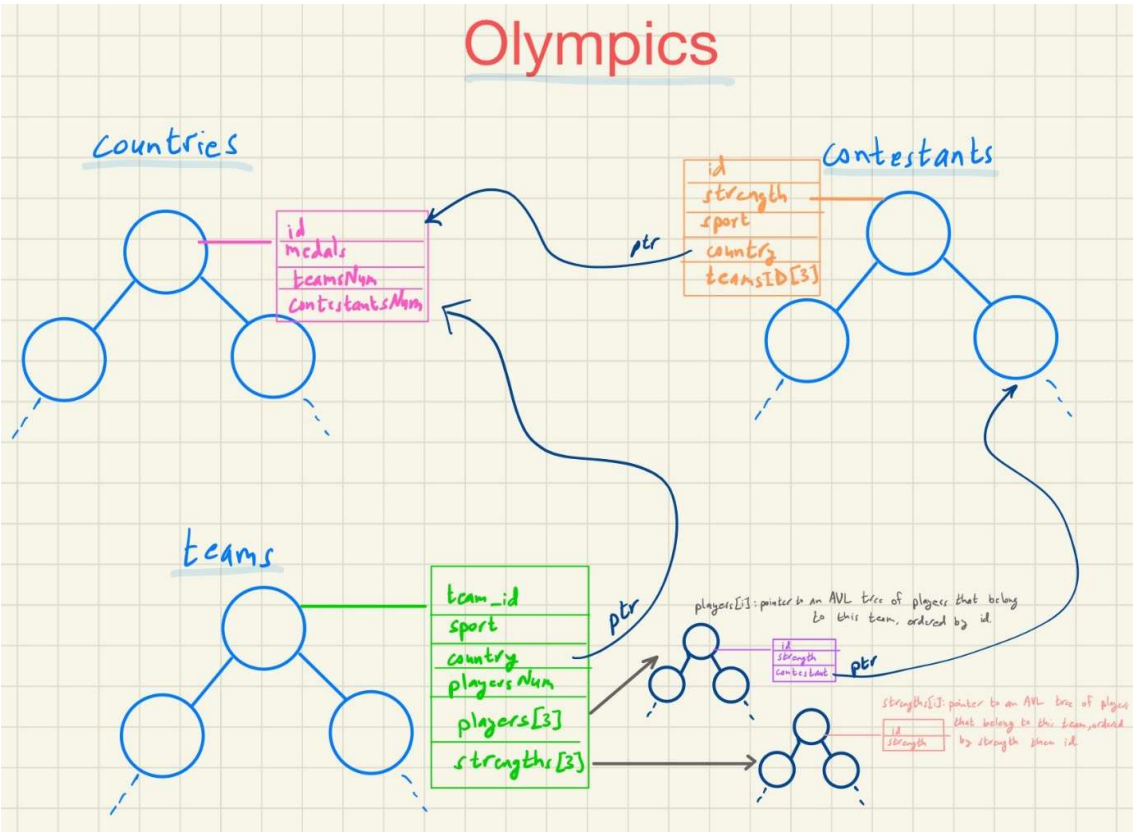
Data Structures 1 – wet 1 – dry section

תיאור של מבני הנתונים

מבנה הנתונים הראשי Olympics מכיל 3 עצים מטיפוסים שונים. נסביר קודם על המחלקות השונות שבהן השתמשנו לאורך התרגיל ואח״כ נסביר על המחלקה הראשית.

1. המחלקה avlTree : מממשת מבנה נתונים מסוג AVL TREE גנרי. מחלקה זו מכילה 4 שדות : root מצביע לשורש העץ, vert_with_min_val, vert_with_max_val מצביע לצומת בעץ בעל הערך המינימלי והמקסימלי בהתאמה, size מספר הצמתים בעץ בזמן נתון. מחלקה זו תומכת בפעולות הבסיסיות של עץ חיפוש : יצירת עץ, חיפוש, הוצאת והכנסת איברים לעץ ובעוד מספר פעולות שנרחיב עליהן בהמשך.
2. המחלקה vertice : היא מחלקה של צומת גנרי (היחידה המרכיבה את עץ ה- AVL). מחלקה זו מכילה 5 שדות : value - ערך (נתונים) טמפלטי השמור בצומת, parent מצביע לצומת האב, left_son, right_son – מצביע לצומת הבן הימני והשמאלי בהתאמה, height – גובה הצומת בעץ. מחלקה זו היא מחלקת עזר למימוש עץ ה-AVL, ולכן הפעולות המוגדרות עליה הן getters and setters.
3. המחלקה Country :מתארת מדינה המשתתפת באולימפיאדה. מחלקה זו מכילה 4 שדות : id-מזהה המדינה, medals- מספר המדליות שהמדינה זכתה בהן, teamsNum-מספר הנבחרות השייכות למדינה זו, contestantsNum- מספר השחקנים השייכים למדינה זו. יחס הסדר המוגדר על אובייקט זה הוא יחס הסדר הרגיל על השדה id.
4. המחלקה Contestant :מתארת מתחרה המשתתף באולימפיאדה. מחלקה זו מכילה 5 שדות : id- מזהה המתחרה, strength- הכוח של המתחרה, sport-סוג הספורט שהמתחרה משחק, country- מצביע למדינה של המתחרה (*Country) ובכך נאפשר גישה ועדכון לשדות המדינה, teamsId- מערך בגודל 3 של מספרים (INT) המייצג את הנבחרות אליהן משתייך המתחרה(הוא יכול להשתתף בעד 3 נבחרות), כל תא מכיל מזהה הנבחרת או 0 המסמן שהתא ריק אם הוא שייך לפחות מ-3 נבחרות. יחס הסדר המוגדר על אובייקט זה הוא יחס הסדר הרגיל על השדה id.
5. המחלקה Player : מתארת שחקן (מתחרה) הפעיל בנבחרת מסוימת. מחלקה זו מכילה 3 שדות : id- מזהה השחקן, strength- הכוח של השחקן, contestant - מצביע לאובייקט מטיפוס Contestant המייצג שחקן זה ובכך נאפשר גישה ועדכון לשדות המתחרה. יחס הסדר המוגדר על אובייקט זה הוא יחס הסדר הרגיל על השדה id.
6. המחלקה Strength : מתארת שחקן הפעיל בנבחרת מסוימת עם התייחסות לכוח שלו. מחלקה זו מכילה 2 שדות : strength- הכוח של השחקן, playerId- מזהה השחקן. לאובייקט זה מוגדר יחס הסדר הבא ; בהינתן 2 שחקנים : השחקן בעל הכוח הגדול יותר הוא הגדול יותר, ואם לשני השחקנים יש אותו כוח אז נשווה את מספר המזהה של השחקן, השחקן בעל המזהה הגדול יותר הינו האובייקט הגדול יותר.
7. המחלקה Team : מתארת נבחרת המשתתפת באולימפיאדה. מחלקה זו מכילה 6 שדות : team_id- מזהה הנבחרת, sport-סוג הספורט של הנבחרת, country- מצביע למדינה של הנבחרת (*Country) ובכך נאפשר גישה ועדכון לשדות המדינה, playersNum- מספר השחקנים המשתתפים בנבחרת, players- מערך בגודל 3, כל תא מכיל מצביע shared_ptr<avlTree<Player>> המצביע לעץ AVL של שחקנים הפעילים בנבחרת זו, 3 העצים יחד מכילים את כל השחקנים של הנבחרת נסמנו N ובכל עץ יש $\left\lfloor \frac{N}{3} \right\rfloor$ או $\left\lceil \frac{N}{3} \right\rceil$ שחקנים, כאשר השחקנים בעלי מזהה מינימלי נמצאים בתא 0 והשחקנים בעלי מזהה מקסימלי נמצאים בתא 2 והנותרים נמצאים בתא 1. נשים לב כי כאשר N מתחלק ב-3 אז בכ״א מהעצים יש בדיוק $\frac{N}{3}$ שחקנים. strengths- מערך בגודל 3, כל תא מכיל מצביע shared_ptr<avlTree<Strength>> : std: המצביע לעץ AVL של שחקנים הפעילים בנבחרת זו והעץ מסודר לפי הכוח, כאשר כל השחקנים בעץ strengths[i] הם אותם שחקנים בעץ players[i] ההבדל הוא שבעץ הראשון הם מסודרים לפי הכוח ובשני לפי המזהה. יחס הסדר המוגדר על אובייקט זה הוא יחס הסדר הרגיל על השדה team_id.
8. מבנה הנתונים הראשי Olympics : זהו מבנה הנתונים הראשי הממשמש את הפעולות הדרושות בתרגיל. המבנה מכיל :
   - **countries** : עץ AVL מטיפוס Country (כל צומת בו הוא מטיפוס Country). העץ מכיל את כל המדינות המשתתפות באולימפיאדה בזמן נתון. עץ זה הינו בגודל k כאשר k הוא מספר המדינות במערכת.
   - **contestants** : עץ AVL מטיפוס Contestant. העץ מכיל את כל המתחרים באולימפיאדה. עץ זה הינו בגודל n כאשר n הוא מספר השחקנים במערכת.
   - **teams** : עץ AVL מטיפוס Team העץ מכיל את כל הנבחרות המשתתפות באולימפיאדה. עץ זה הינו בגודל m כאשר m הוא מספר הנבחרות במערכת. נשים לב כי כל צומת בעץ מכילה אף היא 6 עצים כפי שהזכרנו קודם, אם נסמן את מספר השחקנים בנבחרת ב-n_{TeamID} אז מתקיים כי ב-6 העצים יש $2n_{TeamID}$(לפי התיאור הנ״ל) וגודל כ״א מהעצים ב- players,strengths חסום ע״י n_{TeamID}. נשים לב כי עבור כל נבחרת מתקיים ש- $n_{TeamID} \leq n$ כי כל שחקן פעיל הוא בפרט מתחרה ומתקיים כי גודל כל תתי העצים(המכילים כל השחקנים של הנבחרת פעמיים, פעם כאובייקט מטיפוס Player ופעם מטיפוס Strength) בכל הצמתים בעץ יחדיו שווה לשתיים כפול מספר השחקנים הפעילים בנבחרת כלשהיא שהוא חסום ע״י $2 \cdot 3n = 6n$ כי **מספר השחקנים**

הפעילים חסום ע"י 3n(מהנתון שכל מתחרה יכול להשתתף בעד 3 נבחרות). בחישובי הסיבוכיות נשתמש בסימנים ובחסם זה לפי הצורך.



<u>ריכוז הפעולות והאלגוריתמים שנעשה בהם שימוש חוזר בצירוף הוכחת סיבוכיות הזמן והמקום שלהם:</u>

<u>הערה כללית:</u> בכל המחלקות שמכילות שדה מטיפוס shared_ptr ה-c'tor, cp c'tor, operator =, d'tor, הוגדרו כ-default, כלומר, מבצעים השמה של פוינטרים ונותנים ל-shared_ptr לנהל את הזיכרון (הוא ימחק את האובייקט אליו הוא מצביע כאשר אין עוד מצביע המצביע אליו).

<u>הערה כללית 2:</u> בכל המחלקות המכילות מצביע למדינה או למתחרה, הם בעצם מצביעים לצומת בעץ המכילה את המדינה או המתחרה המתאים, כלומר הם כבר קיימים במערכת ופשוט שומרים מצביע אליהם כדי שנוכל למשל לעדכן את מספר המתחרים/הנבחרות ישירות ב"או" של 1 מבלי לחפש את המדינה בעץ המדינות, וכדי לעדכן את מערך הנבחרות של המתחרה כאשר מוסיפים אותו לנבחרת חדשה/מאחדים נבחרות.

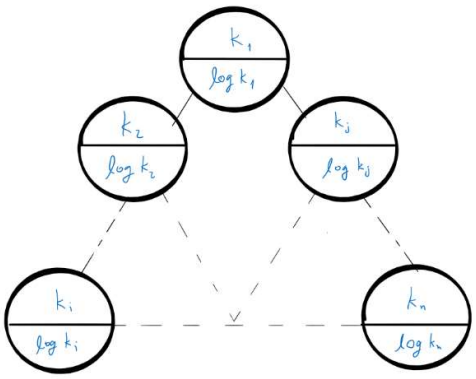
1. במחלקת vertice : פעולות ה-GETTERS מחזירים שדה מסוים כמצביע או רפרנס או מספר שלם ולכן סיבוכיות הזמן והמקום היא $O(1)$. פעולות ה-SETTERS שמעדכנים את השדות : parent, left_son, right_son, height שכ"א מהם הוא מצביע או מספר שלם קורות בסיבוכיות זמן ומקום $O(1)$. בפעולת setValue נקרא אופרטור ההשמה של T (הטיפוס הטמפלטי) ולכן סיבוכיות הזמן והמקום היא כסיבוכיות הזמן והמקום של אופרטור ההשמה של T. כנ"ל לגבי ה- c'tor שבו נקרא ה- copy c'tor של T.
2. במחלקת avlTree : עבור עץ בעל N צמתים, וכל צומת הינה מטיפוס T :

1) c'tor : מאתחל את כל השדות ל NULL או 0 בהתאם ולכן סיבוכיות הזמן והמקום היא O(1).
2) d'tor : מוחק את כל הצמתים של העץ באופן רקורסיבי ע"י קריאה לבנים ואח"כ שימוש ב- delete, שקורא ל-d'tor של T.

- בכל המחלקות שהגדרנו סיבוכיות הזמן והמקום של ה-d'tor של T היא O(1) , אז סיבוכיות הזמן היא לינארית במספר הצמתים בעץ כי בכל קריאה מתבצע מספר קבוע של פעולות. וסיבוכיות המקום היא כעומק עץ הקריאות הרקורסיביות ששווה לעומק העץ, לפי מה שלמדנו בהרצאה זה שווה ל- log(N) ולכן סיבוכיות הזמן של הדסטרקטור של העץ היא : O(N) וסיבוכיות המקום היא : (O(log(N)).

- עבור Team : סיבוכיות הזמן של ה- d'tor של T היא $O(n_{TeamID})$ וסיבוכיות המקום היא $O(\log(n_{TeamID}))$, כאשר n_{TeamID} הוא מספר השחקנים בנבחרת. אז נקבל כי סיבוכיות הזמן של שחרור צומת כזה היא $O(n_{TeamID})$ וסיבוכיות המקום היא $O(\log(n_{TeamID}))$. אם נסמן ב- T(N) את סיבוכיות הזמן של הדסטרקטור של ה- AVL וב- P(N) את סיבוכיות המקום שלו נקבל כי:

<u>לפי שיטת עץ רקורסיות</u>: עבור הצומת $TeamID$ סיבוכיות הזמן היא n_{TeamID}והמקום היא $\log(n_{TeamID})$ ולכן:

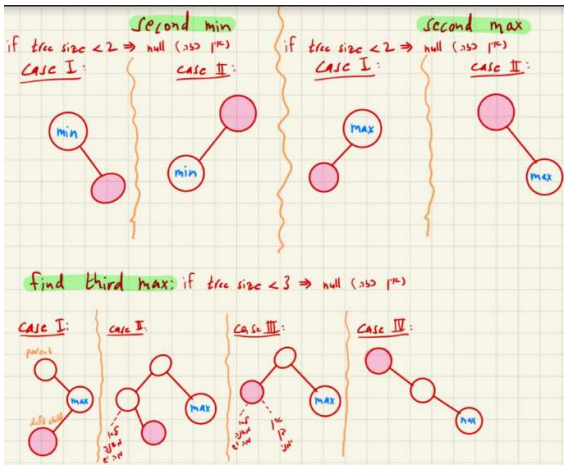


$P(N) = O\left(\left(\sum_{i=1}^{\log N}(\log n_{TeamID})\right)\right) \leq$
$=O(n_{TeamID} \log n_{TeamID}) \leq O(n \log n)$
סכום סיבוכיות המקום של כל צומת בענף של עץ הרקורסיות מהשורש ועד לעלה הכי עמוק.
הסבר: המעבר הראשון לפי טענה מהתרגול והשני לפי החסם שהיצגנו לעיל: $n_{TeamID} \leq n$

$T(N) = O\left(\sum_{i=1}^{N} n_{TeamID} + 1\right) =$
$O\left(\left(\sum_{i=1}^{N} n_{TeamID}\right) + N\right) \leq$
$=O(3n + m) = O(n + m)$ סיבוכיות הזמן
של כל הצמתים יחדיו. הסבר למעבר הראשון: מספר הצמתים בעץ הנבחרות הוא מספר הנבחרות שסימנו ב-m ולפי החסם שמצאנו קודם: " **מספר השחקנים הפעילים חסום ע"י 3n**".

3) findPtr: ממומשת לפי האלגוריתם של find שנלמד בהרצאה. וראינו שסיבוכיות הזמן והמקום שלה היא $O(log(N))$. והיא מחזירה מצביע ל-T או nullptr אם אין T כזה.

4) פעולות GETTERS: מחזירות מצביע או רפרנס ומבצעות מספר קבוע של בדיקות/השוואות/גישה לצומת בן/אב ולכן סיבוכיות הזמן והמקום שלהן היא O(1). נפרט אותן:



- GetMin,GetMax: מחזירות מצביע לאיבר המקסימלי/המינימלי בהתאם.
- GetThirdMax,GetSecondMin,GetSecondMax: הצומת שאנו מחפשים מודגשת בציור וניגשים אליה ב- $O(1)$ מ-vert_with_max_val , vert_with_min_val:

5) insert: ממומשת לפי האלגוריתם שנלמד בהרצאה וראינו שסיבוכיות הזמן והמקום שלה היא $O(log(N))$. וזה המצב למרות שהעץ גנרי כי סיבוכיות הזמן של יצירת האובייקט, וסיבוכיות המקום היא: $O(1)$.

6) deleteVert: מחיקת צומת מסוים מהעץ נעשה לפי האלגוריתם שנלמד בהרצאה וראינו שסיבוכיות הזמן והמקום שלה היא $O(log(N))$. וזה המצב למרות שהעץ גנרי כי סיבוכיות הזמן והמקום של יצירת ושחרור כל האובייקטים(גם עבור Team כי אנו קוראים לפונקציה זו רק כאשר אין בנבחרת שחקנים) היא: $O(1)$. והוספנו עוד שתי פעולות שמעדכנות את השדות vert_with_min_val, vert_with_max_val: מתחילים מהשורש וממשיכים עד הבן הכי שמאלי(המינימלי)/ימני(המקסימלי) בהתאם. סיבוכיות הזמן: $O(log(N))$ ומקום: O(1). ולכן סה"כ סיבוכיות זמן ומקום של הסרת צומת היא: $O(log(N))$.

<u>הערה לגבי insert,deleteVert</u>: לפני הקריאה לפונקציה פנימית רקורסיבית אנו מחפשים אם קיים הצומת בעזרת find (כדי להחזיר שגיאה אם קיימת). ואחרי סיום הרקורסיה אנו מעדכנים את ה-SIZE של העץ והצומת בעל הערך המינימלי והמקסימלי. אבל כל זה קורה בסיבוכיות לוגריתמית במספר הצמתים ולכן לא משנה את הסיבוכיות של הפעולות(החסם האסימפטוטי).

7) sortedArrayToTree: יוצרת עץ AVL חדש עם K צמתים כאשר כל צומת הוא מטיפוס T ומחזירה מצביע אליו. היא בעצם ממירה מערך ממוין של אובייקטים מטיפוס T לעץ AVL. היא מקבלת מערך ושני אינדקסים first,last, קובעת את התא האמצעי $mid = \frac{first+las}{2}$ ויוצרת צומת עם הערך שנמצא בתא mid. באופן

רקורסיבי מבצעת את אותו תהליך עבור הבן הימני כאשר first=mid+1,last=last ועבור הבן השמאלי כאשר first=first, last=mid-1. ובכל קריאה מתבצע מספר קבוע של פעולות. תנאי עצירה first>last. אנו משתמשים בפונקציה זו עם אובייקט T שסיבוכיות הזמן והמקום של יצירתו היא $O(1)$ מכאן נוכל לסמן את הסיבוכיות של כל קריאה ב-C. ולכן סיבוכיות הזמן היא: $\sum_{i=1}^{K} C = KC = O(K)$, וסיבוכיות המקום נסמנה ב-$p(K)$ היא: $p(K) = p_{STACK}(K) + p_{HEAP}(K) = log\, K + KC = O(K)$.

8) PrintInOrder: הפונקציה מבצעת סיור inorder שמתחיל בצומת בעל הערך המינימלי (ולא מהשורש) בעץ בעזרת לולאה. אנו שומרים דגל המציין אם סיימנו לעבור בתת העץ השמאלי או לא ובעזרתו ובעזרת מצביע לצומת האב קובעים אם ללכת שמאלה או ימינה או לעלות לאב (כפי שראינו בתרגול 4). הלולאה מתבצעת עד שנעבור על כל הצמתים בעץ סה"כ N צמתים. בכל איטרציה אנו שומרים את ערך הצומת ה-K בסיור (האובייקט השמור בה) בתא K במערך שהתקבל כארגומנט לפונקציה. וכך נמיר את העץ למערך ממוין בסדר עולה. סיור IN ORDER קורה בסיבוכיות זמן $O(N)$ לפי מה שלמדנו בתרגול, וסיבוכיות המקום היא $O(1)$(אין רקורסיה/ אין הקצאות כי המערך מוקצה מראש).

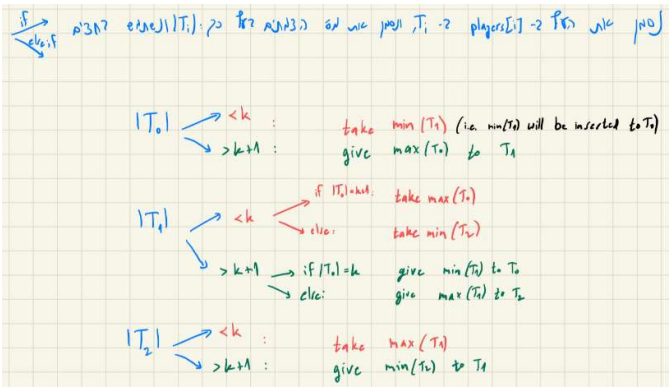
9) mergeSortedArr: מקבלת שני מערכים ממוינים בסדר עולה וממזגת אותם למערך ממוין בסדר עולה חדש **ללא כפילויות** מחזירה אותו ושומרת את גודלו במצביע. נקצה מערך בגודל ששווה לסכום גדלי המערכים ונמיין את 2 המערכים בעזרת אלגוריתם mergeSort עבור מערכים ממוינים (שלמדנו במבוא למדמ"ח) לתוכו. עכשיו יש לנו מערך A המכיל את מיזוג המערכים אך ייתכן כפלויות, כדי להסיר אותם נקצה עוד מערך B ונעבור על מערך A ונעתיק אותו ל-B כך שאם איבר מופיע יותר מפעם אחת נעתיק רק את הופעתו הראשונה, ותוך כדי נשמור גם את מספר האיברים בלי כפילויות ונשמור אותו בתוך פוינטר שהועבר לפונקציה ונחזיר את B. לפי מה שלמדנו בקורס מבוא נקבל כי סיבוכיות הזמן היא: $O(n_1 + n_2) + O(n_1 + n_2) = O(n_1 + n_2)$.
סיבוכיות המקום: $2 * O(n_1 + n_2) = O(n_1 + n_2)$. n_1, n_2 גדלי המערכים)

3. במחלקת Country: במחלקה זו כל השדות הם מטיפוס INT ולכן סיבוכיות הזמן והמקום של ה- d'tor, copy c'tor, c'tor, =operator וגם אופרטורי ההשוואה (שבהם נעשה מספר קבוע של פעולות השוואת INT-ים) היא $O(1)$.
4. במחלקת Contestant: במחלקה זו השדות הם מטיפוס INT או ENUM או מערך בגודל 3 של INT-ים, או מצביע למדינה(שכל שדותיה מספרים) ולכן סיבוכיות הזמן והמקום של ה- d'tor, copy c'tor, c'tor, =operator וגם אופרטורי ההשוואה היא $O(1)$. על מחלקה זו מוגרות פעולות נוספות הקשורות למערך int teamsId[3] שסיבוכיות הזמן והמקום שלהן היא $O(1)$. למשל: החזרת אינדקס נבחרת מסוימת במערך, בדיקה אם יש תא ריק ואז ניתן יהיה להוסיפו לעוד נבחרת, עדכון מזהה קבוצה ממזהה ישן לחדש שנשתמש בה כאשר נאחד נבחרות וכו'.
5. במחלקת Player: במחלקה זו כל השדות הם מטיפוס INT או מצביע למתחרה Contestant ולכן סיבוכיות הזמן והמקום של ה- d'tor, copy c'tor, c'tor, =operator וגם אופרטורי ההשוואה היא $O(1)$.
6. במחלקת Strength: במחלקה זו כל השדות הם מטיפוס INT ולכן סיבוכיות הזמן והמקום של ה- d'tor, copy c'tor, c'tor, =operator וגם אופרטורי ההשוואה היא $O(1)$.
7. במחלקת Team: במחלקה זו כל השדות הם מטיפוס INT או ENUM או מצביע למדינה(שכל שדותיה מספרים) או מערך בגודל 3 של מצביעים SHARED POINTER לעץ (שבהתחלה כלומר כאשר מוסיפים נבחרת לאולימפיאדה, העצים יהיו ריקים כי אין שחקנים בנבחרת) ולכן סיבוכיות הזמן והמקום של ה- copy c'tor, c'tor, =operator וגם אופרטורי ההשוואה היא $O(1)$. עבור ה-d'tor ברוב המקרים סיבוכיות הזמן והמקום היא O(1) כי אנו נמחק את הנבחרת רק כאשר אין בה עוד שחקנים, כלומר רק כאשר העצים ריקים. אך ייתכן שנסיים את התכנית ועדיין יש שחקנים פעילים בנבחרת ואז סיבוכיות הזמן היא כסיבוכיות הזמן והמקום של שחרור 6 עצים בגודל n_{TeamID}← סיבוכיות הזמן: $O(n_{TeamID})$ וסיבוכיות המקום היא $O(log\, n_{TeamID})$.
למחלקה זו הגדרנו גם את הפעולות הבאות:

- addPlayerToTeam: המקבלת שחקן ואינדקס i ומוסיפה את השחקן לעצים strengths[i],players[i] , סיבוכיות הזמן והמקום היא $O(log\, n_{TeamID})$ (לפי ניתוח גודל העץ שעשינו קודם).
- removeFromAddTo: המקבלת שחקן ושני אינדקסים ומוחקת השחקן מהעצים strengths[i],players[i] עבור i מסוים ומוסיפה אותו לעצים strengths[j],players[j] עבור j אחר. סיבוכיות הזמן והמקום היא $4 * O(\log n_{TeamID}) = O(log\, n_{TeamID})$ כי מבצעים 2 הוספות ו-2 מחיקות.
- getStrength: מחזירה את כוח הנבחרת, החישוב נעשה כפי שהוגדר בתרגיל: אם מספר השחקנים לא מתחלק ב-3 אז מחזירה 0 ואחרת בכל עץ במערך strengths יש $\frac{n_{TeamID}}{3}$ בראשון יהיו השחקנים עם מזהה מינימלי ובשני יהיו השחקנים הבאים מבחינת סדר המזהה ובשלישי יהיו השחקנים בעלי מזהה מקסימלי והעצים מסודרים לפי הכוח, ולכן האיבר המקסימלי בכל עץ מכיל את השחקן בכל הכוח המקסימלי באותה קבוצה,

ולכן נסכום את הכוחות של הצמתים המקסימליים ב-3 העצים ונקבל את הנדרש. מכיוון אנו שומרים את צומת זו כשדה בעץ אז יש לנו גישה אליה ב"או" של 1 ולכן סיבוכיות הזמן והמקום היא O(1).

- fix: הפונקציה הזו אחראית על "תיקון" העצים הפנימיים של הנבחרת כדי לשמור על המבנה שלהם לפי התיאור מקודם: אם מספר השחקנים של הנבחרת הוא N=n_{TeamID} אז בכל עץ במערך players יש $\left\lceil\frac{N}{3}\right\rceil$ או $\left\lceil\frac{N}{3}\right\rceil$ שחקנים, כאשר השחקנים בעלי מזהה מינימלי נמצאים בתא 0 והשחקנים בעלי מזהה מקסימלי נמצאים בתא 2 והנותרים נמצאים בתא 1 ובמערך strengths מתקיים שכל השחקנים בעץ strengths[i] הם אותם שחקנים בעץ players[i]. לצורך זה נבצע את האלגוריתם הבא: נסמן: $k = \left\lceil\frac{N}{3}\right\rceil$ אז $k+1 = \left\lceil\frac{N}{3}\right\rceil$, ונעבור פעמיים על העצים players[i] מ-0 ונבצע את האלגוריתם הבא:



נדגיש שעבור כל שינוי בעצים האלה אנו מבצעים את השינוי המתאים גם בעץ strengths[i] המתאים(למשל: אם נעביר שחקן בעל מזהה מקסימלי מעץ 0 לעץ 1 אז נעביר אותו שחקן מעץ 0 לעץ 1 במערך strengths וכך נוודא שהעצים בשני המערכים מכילים את אותם שחקנים). הסבר: אנו קוראים לפונקציה זו כאשר מכניסים/מסירים שחקן מהנבחרת, לפני השינוי, העצים היו תקינים ולאחר הוספה/הסרה מספר הצמתים באחד העצים גדל/קטן ב-1, ולכן למשל אם מספר הצמתים בעץ מסוים הוא קטן מ-k אז הוא שווה ל-k-1. ולכן לאחר כל תיקון מספר הצמתים יהיה בדיוק k. והוא יישאר תקין גם לאחר השינויים הבאים. אך ייתכן שהעץ היה תקין מלכתחילה ולא דרש תיקון אבל העץ הבא כן תיקנו אז ייתכן שפגענו בתקינות העץ הראשון ולכן אנו מבצעים תהליך זה פעמיים כדי לוודא שכל העצים יהיו תקינים בסוף התהליך. הסיבוכיות הזמן והמקום היא $4 * O(\log N) = O(log\ N)$ כי יש לכל היותר 6 קריאות לפונקציה removeFromAddTo, ויש גישה של O(1) לצומת המינימלי והמקסימלי בכל עץ.

<u>הסברים על מימוש הפעולות הנדרשות בצירוף הוכחת סיבוכיות הזמן והמקום:</u>

הערה: בהוכחת סיבוכיות השתמשנו בסימנים n,k,m וכו שהוגדרו בקובץ ת.ב. עבור כל פונקציה.

הערה: אנו נשתמש בגודל הכי גדול שייתכן עבור כל עץ כפי שפירטנו לעיל בתיאור מבנה הנתונים הראשי. ונשתמש בחישובים קודמים של הסיבוכיות שעשינו הנ"ל.

הערה: בהתחלת כל פונקציה, אנו בודקים שהארגומנטים שהועברו לפונקציה תקינים לפי התנאים שהוגדרו בקובץ התרגיל. בדיקות אלה קורות בסיבוכיות זמן ומקום של O(1).

1. Olympics(): מאתחלת את האולימפיאדה להיות ריקה ע"י קריאה ל- c'tor של המחלקה avlTree המאתחל את שורש העץ ל- null עבור כל עץ ב-Olympics (3 עצים).
<u>סיבוכיות הזמן</u>: O(1) – כסיבוכיות הזמן של ה-c'tor של העץ וכנ"ל לגבי <u>המקום</u> => O(1).
2. ~Olympics(): הפעולה משחררת את האולימפיאדה ע"י קריאה ל-d'tor של כל אחד מהעצים. <u>סיבוכיות הזמן</u>:
$$O(k) + O(n) + O(m) = O(k + n + m)$$
<u>סיבוכיות המקום</u>:
$$O(logk) + O(logn) + O(logm) = O(logk + logn + logm) = O(\log knm)$$
כאשר k, n, m הם מספר הנבחרות במערכת, מספר השחקנים במערכת, מספר המדינות במערכת. בהתאמה.
3. add_country(int countryId, int medals): פונקציה זו מוסיפה מדינה חדשה לאולימפיאדה. בהתחלה אנו מוודאים שאין מדינה עם המזהה countryId ע"י קריאת findPtr על העץ **countries**,אם קיימת כזו נחזיר שגיאה. ואם לא, נוסיף את המדינה החדשה למערכת ע"י קריאה ל- insert לעץ **countries**.
<u>סיבוכיות הזמן והמקום</u>:
$$2 \cdot O(log\ k) = O(log\ k)$$
4. remove_country(int countryId): הפונקציה מסירה את המדינה בעלת המזהה countryId מהאולימפיאדה. בהתחלה בודקים אם קיימת מדינה עם מזהה זה ע"י קריאת findPtr על העץ **countries**, נסמן ערך החזרה ב-toDelete ומחזרים שגיאה לפי ההנחיות בקובץ. אם אין שגיאה, אז נמחק את המדינה (עם המזהה הנתון) מהעץ **countries**.

סיבוכיות הזמן והמקום:

$$2 \cdot O(log\, k) = O(log\, k)$$

5. add_team(int teamId, int countryId, Sport sport): פונקציה זו מוסיפה נבחרת חדשה לאולימפיאדה. בהתחלה אנו מוודאים שאין נבחרת עם המזהה teamId וגם שקיימת מדינה עם מזהה countryId ע"י קריאת findPtr על העצים **countries ,teams** בהתאמה ,אם הדרישות(לפי הנחיות התרגיל) לא מתקיימות נחזיר שגיאה. ואם לא, נוסיף את הנבחרת החדשה לאולימפיאדה ע"י קריאה ל- insert לעץ **teams**. ונעדכן את מספר הנבחרות teamsNum במדינה.

סיבוכיות הזמן והמקום:

$$2 \cdot O(log\, m) + O(\log k) = O(logm + log\, k)$$

6. remove_team(int teamId): פונקציה זו מסירה את הנבחרת בעלת המזהה teamId מהאולימפיאדה. בהתחלה בודקים אם קיימת נבחרת עם מזהה זה ע"י קריאת findPtr על העץ **teams**, ובודקים אם יש שחקנים פעילים בנבחרת זו ומחזירים FAILURE בהתאם. אם אין שגיאה, אז נעדכן את מספר הנבחרות במדינה המתאימה דרך גישה מהירה למדינה מעץ המדינות $O(1)$ לשדה country, ואחר כך נמחק את הנבחרת מהעץ **teams**.

סיבוכיות הזמן והמקום: נשים לב כי העצים ב-players,strengths ריקים ולכן נשחרר אותם בזמן קבוע:

$$2 \cdot O(log\, m) = O(log\, m)$$

7. add_contestant(int contestantId, int countryId, Sport sport, int strength): פונקציה זו מוסיפה מתחרה חדש לאולימפיאדה. בהתחלה אנו מוודאים שאין מתחרה עם המזהה הנתון וגם שקיימת מדינה עם מזהה countryId ע"י קריאת findPtr על העצים **countries ,contestants** בהתאמה ,אם הדרישות(לפי הנחיות התרגיל) לא מתקיימות נחזיר שגיאה. ואם לא, נוסיף את המתחרה החדש לאולימפיאדה ע"י קריאה ל- insert לעץ **contestants**. ונעדכן את מספר המתחרים במדינה.

סיבוכיות הזמן והמקום:

$$2 \cdot O(log\, n) + O(\log k) = O(logn + log\, k)$$

8. remove_contestant(int contestantId): פונקציה זו מסירה את המתחרה בעל המזהה הנתון מהאולימפיאדה. בהתחלה בודקים אם קיים מתחרה עם מזהה זה ע"י קריאת findPtr על העץ **contestants**, ומחזרים שגיאה לפי ההנחיות בקובץ. אם אין שגיאה, אז נעדכן את מספר המתחרים במדינה המתאימה דרך גישה מהירה למדינה מעץ המדינות $O(1)$ לשדה country, ואחר כך נמחק את המתחרה מהעץ **contestants**.

סיבוכיות הזמן והמקום:

$$2 \cdot O(log\, m) = O(log\, m)$$

9. addPlayerToTeamHelper: פונקציית עזר שמוסיפה שחקן לנבחרת, מניחים שהכל תקין. בהתחלה בודקים לאיזה עץ ב- players צריך להוסיף את השחקן לפי המזהה שלו: אם הוא קטן מהמזהה (הצומת) המקסימלי ב-players[0] נוסיף אותו לעץ זה, ואם הוא גדול מהמזהה (הצומת) המינימלי ב-players[2] נוסיף אותו לעץ זה, אחרת נוסיף אותו לעץ players[1]. נעדכן את מספר השחקנים בנבחרת ואח"כ נתקן את העצים בעזרת הפ' fix().

סיבוכיות הזמן והמקום: $O(1) + O(log\, n_{TeamID}) + O(log\, n_{TeamID}) = O(log\, n_{TeamID})$.

10. add_contestant_to_team(int teamId,int contestantId): בהתחלה נבדוק תקינות: נבדוק אם המתחרה והנבחרת הנתונים נמצאים בעצים teams,contestants ע"י קריאה ל-findPtr . ובודקים אם שניהם שייכים לאותה מדינה ומשחקים את אותו ספורט, וגם אם המתחרה כבר נמצא בנבחרת/ אם הוא כבר ב-3 נבחרות ונחזיר FAILURE בהתאם. בהנחה שהכל תקין נוסיף את השחקן לנבחרת ע"י קריאה ל-addPlayerToTeamHelper ונעדכן שהמתחרה שייך לנבחרת במערך teamsId. סיבוכיות הזמן והמקום:

$O(1) + O(log\, m) + O(log\, n) + O(log\, n_{TeamID}) = O(log\, n + \log m)$.

11. remove_contestant_from_team(int teamId, int contestantId): בהתחלה נבדוק תקינות: נבדוק אם המתחרה והנבחרת הנתונים נמצאים בעצים teams,contestants ע"י קריאה ל-findPtr ונבדוק אם המתחרה נמצא בנבחרת ונחזיר FAILURE בהתאם. בהנחה שהכל תקין נסיר את השחקן מהנבחרת: נבדוק באיזה עץ הוא נמצא ב-strengths, players ע"י ביצוע חיפוש ב-3 העצים(ב-strengths יהיה בעץ בעל אותו אינדקס) ונסיר אותו, נעדכן את מספר השחקנים בנבחרת ואח"כ נתקן את העצים בעזרת הפ' fix(). סיבוכיות הזמן והמקום:

$$O(1) + O(log\, m) + O(log\, n) + 3 * O(log\, n_{TeamID}) + O(log\, n_{TeamID}) = O(log\, n + \log m)$$

12. update_contestant_strength(int contestantId, int change): בהתחלה נבדוק תקינות: נבדוק אם המתחרה הנתון נמצא בעץ contestants ע"י קריאה ל-findPtr . ובודקים אם הכוח שלו אחרי שינוי קטן מ-0 ונחזיר FAILURE בהתאם. בהנחה שהכל תקין אז עבור כל נבחרת שהמתחרה נמצא בה(לכל היותר 3 נבחרות) נבצע את האלגוריתם הבא: נמצא את הנבחרת בעץ teams נסיר את השחקן עם המזהה הנתון מהנבחרת ואח"כ נוסיף אותו שוב לנבחרת עם הכוח החדש ע"י קריאה ל-add_contestant_to_team , remove_contestant_from_team.

סיבוכיות הזמן והמקום היא:

$$(1) + O(log\, n) + 3 * (O(log\, m) + O(log\, n + \log m)) = O(log\, n + \log m)$$

13. (int contestantId)get_strength : בהתחלה נבדוק תקינות : נבדוק אם המתחרה הנתון נמצא בעץ contestants ע"י קריאה ל-findPtr . ונחזיר FAILURE בהתאם. בהנחה שהכל תקין ומצאנו את המתחרה אז נחזיר את הכוח שלו. <u>סיבוכיות הזמן והמקום</u> היא : $O(log\, n)$ כסיבוכיות החיפוש בעץ.

14. (int countryId)get_medals : בהתחלה נבדוק תקינות : נבדוק אם המדינה הנתונה נמצאת בעץ countries ע"י קריאה ל-findPtr . ונחזיר FAILURE בהתאם. בהנחה שהכל תקין ומצאנו את המדינה אז נחזיר את כמות המדליות שלה. <u>סיבוכיות הזמן והמקום</u> היא : $O(log\, k)$ כסיבוכיות החיפוש בעץ.

15. (int teamId)get_team_strength : בהתחלה נבדוק תקינות : נבדוק אם הנבחרת הנתונה נמצאת בעץ teams ע"י קריאה ל-findPtr . ונחזיר FAILURE בהתאם. בהנחה שהכל תקין ומצאנו את הנבחרת אז נחזיר את הכוח של הנבחרת שנחשב בעזרת הפ' ()getStrength לפי מה שהוגדר בתרגיל(המימוש שלה הוגדר לעיל). <u>סיבוכיות הזמן והמקום</u> היא : $O(log\, m) + O(1) = O(\log m)$ כסיבוכיות החיפוש בעץ וחישוב הכוח.

16. (int teamId1, int teamId2)unite_teams : פונקציה זו מאחדת את שתי הנבחרות שקיבלה לנבחרת אחת בעלת המזהה teamId1. בהתחלה בודקים אם שתי הנבחרות קיימות בעץ **teams** ע"י קריאה ל-findPtr אם הן קיימות אז נבדוק שהן בעלות מדינה/ספורט זהה , ומחזירים שגיאה לפי ההנחיות בקובץ. כעת אם לפחות אחת מהנבחרות אינה ריקה :

נמיר כל עץ שחקנים מבין 3העצים ב-players למערך ממוין בעזרת סיור in order (ע"י קריאה ל-PrintInOrder) ואז נמזג אותם למערך אחד ממוין(לפי מזהה) ללא כפילויות בעזרת אלגוריתם Merge Sort(ע"י קריאה לפ' mergeSortedArr) נקרא לו ב- unitedPlayers ונבצע תהליך זה גם עבור כל עץ ב- Strength שממוין לפי כוח, נקרא לו ב- unitedStrengths. ונעדכן את מערך ה-teamsId של כל אחד מהמתחרים בעזרת גישה מהירה לתוך מצביע contestant (של השחקן), כעת נקצה מערך חדש כגודל ששווה למספר השחקנים בשתי הנבחרות יחד, שיכיל את השחקנים ממוינים לפי כוח כך שבשליש התחתון של המערך הוא יכיל את השחקנים בעלי מזהה מינימלי וכך הלאה, כלומר השחקנים הנמאים בשליש הראשון/השני/השלישי במערך זה יהיו אותם שחקנים הנמצאים בשליש הראשון/השני/השלישי במערך unitedPlayers אך במערך זה הם יהיו מסודרים לפי כוח ולא לפי מזהה(כדי לבנות מערך זה ניעזר במערך unitedStrengths). אחר כך נמיר כל אחד מהמעריכים unitedPlayers והמערך החדש ל-3 עצים(שליש i יומר לעץ i) ע"י קריאה ל-sortedArrayToTree ונשמור אותם בתוך השדות players, strengths בהתאם. ואז נעדכן את מספר השחקנים בכל נבחרת בהתאם.

נשים לב כי התהליך לשמירה נעשה רק עבור $3 \cdot \left\lfloor \frac{n_{teamId1}+n_{teamId2}}{3} \right\rfloor$ השחקנים הראשונים, במקרה שמספר השחקנים בשתי הנבחרות אינו מתחלק ב-3, אז יש עוד 1-2 שחקנים שצריך להוסיף, נוסיף אותם ע"י קריאה ל-addPlayerToTeamHelper. ומחיקת הנבחרת השנייה מהאולימפיאדה (ע"י קריאה ל-remove_team) בסוף מוודאים ששחררנו את כל הזיכרון שהוקצה עבור הפונקציה.

<u>ניתוח סיבוכיות :</u> 2 חיפושים בעץ, המרת 12 עצים השייכים לשתי הנבחרות ל-2 מערכים ממוינים, גדלי העצים ביחד חסומים ע"י $2(n_{TeamId1} + n_{TeamId2})$ וסיבוכיות ההמרה למערך היא לינארית במספר הצמתים בעץ. ולכן סיבוכיות הזמן והמקום של פעולה זו היא לינארית במספר השחקנים בשתי הנבחרות. לעבור על כ"א מהמערכים כדי לעדכן שדה/ לשמור את נתוניו למערך 3, גם זה לינארי בגדלי המערכים. המרת שני המערכים לעצים-לינארי בגודל המערך. הוספת לכל היותר 2 שחקנים ע"י הפ' הנ"ל ושחרור מערכים. נסמן ב-C קבוע המסמן את מספר הפעולות הלינאריות במספר השחקנים אז <u>סיבוכיות הזמן והמקום</u> היא :

$$2 \cdot O(\log m) + C * O(n_{TeamId1} + n_{TeamId2}) + 2 * O(\log(n_{TeamId1} + n_{TeamId2}))$$
$$= O(\log m + n_{teamId1} + n_{teamId2})$$

17. (int teamId1, int teamId2)play_match : בהתחלה נבדוק תקינות : נבדוק אם הנבחרות הנתונות נמצאת בעץ teams ע"י קריאה ל-findPtr ונבדוק אם שתי הנבחרות משחקות את אותו ספורט ונחזיר FAILURE בהתאם. בהנחה שהכל תקין ומצאנו את הנבחרות אז נחשב את הניקוד של כל נבחרת : מספר המדליות במדינה שהנבחרת משחקת בשבילה + הכוח של הנבחרת שנחשב בעזרת הפ' get_team_strength. אם הניקוד מסיים אז המשחק הסתיים בתיקו ולא השתנה כלום, אחרת נוסיף מדליה אחת למדינה של הנבחרת המנצחת.

<u>סיבוכיות הזמן והמקום</u> היא : $2 * (O(log\, m) + O(1) + O(1)) = O(\log m) = O(\log k + \log m)$ כסיבוכיות החיפוש בעץ וחישוב הכוח.

18. (int teamId)austerity_measures : בהתחלה נבדוק תקינות : נבדוק אם הנבחרת הנתונה נמצאת בעץ teams ע"י קריאה ל-findPtr ונבדוק אם יש בה פחות מ-3 שחקנים ונחזיר FAILURE בהתאם. בהנחה שהכל תקין ומצאנו את הנבחרת אז נבדוק אם מספר השחקנים בה מתחלק ב-3 או שווה ל-3 במקרה שלא מתחלק/במקרה ששווה אז נחזיר 0 כנדרש(אם נעיף 3 שחקנים אז הנבחרת את תהיה ריקה וכוחה 0). כעת, יש לפחות 6 שחקנים ומספרם מתחלק ב-3 ולכן בכל עץ ב- strengths,players יש בדיוק $\frac{n_{TeamID}}{3}$. כעת נעבור על כל אפשריות ההסרה של 3 מתחרים שתיתן כוח מקסימלי עבור הנבחרת ועבור כל אפשרות נחשב את הכוח החדש של הנבחרת ובסוף נחשב את המקסימום על כל האפשריות וזה יהיה הכוח המקסימלי שיכול להיות לנבחרת לאחר העפת 3 שחקנים. מכאן והלאה נתייחס לכל עץ ב- players כקבוצה(נשים לב כי קבוצות אלה הן אותן קבוצות

שהוגדרו בקובץ התרגיל בפ׳ get_team_strength) ונאמר ״נסיר״ למרות שאנחנו לא באמת נסיר שחקן מנבחרת אלא נחשב את כוח הנבחרת בלעדיו. מספר אפשריות ההסרה של 3 שחקנים מ-3 קבוצות הוא 10: מסירים שחקן 1 מכל קבוצה(1), מסירים 3 שחקנים מקבוצה מסוימת (3), מסירים שני שחקנים מקבוצה אחת ושחקן אחד מקבוצה אחרת(6). כדי לקבל את הכוח המקסימלי עבור כל אפשרות אנו נסיר תמיד את השחקנים בעלי כוח מינימלי. נחשב את הכוח עבור כל אפשרות באופן הבא:

- נבין מי הן הקבוצות החדשות לאחר ההסרה: הן יכילו את אותם שחקנים פרט לאלה שהסרנו שהם בעלי כוח מינימלי ולכן בפועל לא ישפיעו על כוח הנבחרת עבור מספר שחקנים גדול מספיק (12 שחקן ויותר) כי אז השחקן בעל כוח מקסימלי שנכנס לחישוב כוח הנבחרת יישאר בקבוצה. וייתכן שנוסף לקבוצה עד 2 שחקנים מקבוצה שאחריה/לפניה או שהסרנו ממנה עד 2 שחקנים שעברו לקבוצה שלפניה/אחריה כדי לשמור על גודל $\frac{n_{TeamID}-1}{3}$ עבור כ״א מהקבוצות. ונשים לב שהשחקנים שיעברו מקבוצה לקבוצה הם השחקנים שבקצוות הקבוצה, כלומר בעלי מזהה מינימלי/מקסימלי.
- נחשב את כוח הנבחרת החדש: עבור כל קבוצה אנו יודעים את השחקן בעל הכוח המקסימלי ואת כוחות השחקנים שהתווספו לקבוצה ואת השחקנים שעברו ממנה לקבוצה אחרת, נחשב את המקסימום של הקבוצה בלי השחקנים שעברו לקבוצה אחרת: הוא יהיה השחקן בעל הכוח המקסימלי הראשון/השני/ השלישי (למשל אם במקרה שהשחקן שעבר לקבוצה אחרת היה השחקן עם הכוח המקסימלי אז עכשיו השחקן בעל הכוח המקסימלי הוא זה שהיה בעל הכוח השני הגדול ביותר) שנחשב אותו ע״י קריאה לפ׳ GetThirdMax,GetSecondMax,GetMaxPtr על העצים ב-strengths הממוינים לפי כוח, ונחשב את כוח השחקים שעברו לקבוצה: כפי שהזכרנו אלה הם השחקנים בקצוות הקבוצות כלומר השחקנים בעלי מזהה מינימלי/מקסימלי שגם אותם נחשב ע״י קריאה לפ׳ GetSecondMax,GetMaxPtr אך הפעם על העצים ב-players הממוינים לפי מזהה. ונחשב את המקסימום ביניהם(בין הכוח של הקבוצה לאחר הסרת והעברת שחקנים לבין כ״א מכוחות השחקנים שנוספו לקבוצה) וזה יהיה הכוח של השחקן הכי חזק בקבוצה זו.
- לאחר שחישבנו את הכוח של השחקן הכי חזק בכ״א מהקבוצות לאחר השינוי נסכום אותם וזה יהיה הכוח של הנבחרת לפי הגדרה עבור אפשרות זו.

מספר הערות: השחקנים שייתכן שיעברו מקבוצה לקבוצה הם: השחקן בעל מזהה מקסימלי מקבוצה 1(בעץ players[0]), שני השחקנים בעלי מזהה מינימלי ושני שחקנים בעלי מזהה מקסימלי מקבוצה 2, השחקן בעל מזהה מינימלי מקבוצה 3. כל שחקן שעובר לעץ הבא/ הקודם ולא עובר שני עצים קדימה/ אחורה. יש טיפול מיוחד עבור המקרים הבאים: אם מספר השחקנים בנבחרת הוא 6(אז לא ניתן למשל להסיר 3 שחקנים מאותה קבוצה כי בכל קבוצה יש שני שחקנים), אם מספר השחקנים הוא 9(אז בכל קבוצה יש 3 שחקנים, אם נסיר את כל השחקנים לא יישארו שחקנים מהקבוצה המקורית והשחקנים שבקבוצה יהיו השחקנים שהועברו אליה).

<u>*ניתוח סיבוכיות*</u>: נשים לב שחישוב המקסימום/ המינימום הראשון/השני/השלישי נעשה בזמן קבוע, וחישוב הכוח החדש עבור כל אפשרות גם נעשה בזמן קבוע(יש לנו את כל הכוחות – 3 כוחות לכל היותר – שייתכן שהן הכוח המקסימלי בכל קבוצה מחשבים את המקסימום ביניהם, עושים זאת עבור 3 קבוצות ואח״כ סוכמים), בנוסף יש מספר קבוע של השוואות ותנאי IF ומספר האפשריות 10- קבוע ולכן פעולת חישוב הכוח הגדול ביותר שייתכן לנבחרת לאחר העפת 3 מתחרים נעשה ב-O(1). ואין הקצאות, ולכן <u>סיבוכיות הזמן והמקום</u>: היא כסיבוכיות החיפוש בעץ: $O(log\ m) + O(1) = O(log\ m)$.

<u>סיבוכיות המקום של המבנה:</u> נסמן ב- n_{TeamID} הוא מספר השחקנים השייכים לנבחרת כלשהיא. המבנה מכיל 3 עצים שכ״א מהם תואר לעיל וגם גודלו פורט לעיל, נקבל כי סיבוכיות המקום היא:

$$0(k) + O(n) + O(m) + O\left(\sum_{for\ each\ teamID\ in\ the\ system} 2n_{TeamID}\right) = O(k+m+n)$$

כאשר המעבר האחרון נובע מכך ש-$\sum_{for\ each\ teamID\ in\ the\ system} 2n_{TeamID} \leq 6n$.