**הסבר על מבני הנתונים:**

1. עץ הAVL:

מימוש עץ AVL כפי שנלמד בכיתה עם השינויים הבאים:

* שדה המידע של העץ הוא טמפלייטי, כדי לממש עצים מטיפוסים שונים (שחקנים/ קבוצות)
* לכל node בעץ מצביע לשני בניו וגם לאביו.

מידע חשוב על המחלקות team ו- player (לצורך הוכחת סיבוכיות בהמשך)

1. קבוצה:

* לכל קבוצה מצביע לtop\_scorer שלה.
* לכל קבוצה שני עצים עבור השחקנים שלה (אחד ממוין לפי id והשני ממוין לפי goals, cards and ids (כפי שנדרש עבור קביעת הדירוג בין השחקנים)
* שדות עבור מספר המשחקים, הכרטיסים ומספר השוערים בקבוצה

**הסבר על הפעולות:**

1. **World\_cup\_t()**
2. **~world\_cup\_t()**
3. **Add\_team:**

הוספת הקבוצה לעץ הקבוצות (הוספה של איבר לעץ AVL לוקחת log(k) במקרה הגרוע כאשר k הוא מספר הקבוצות במערכת.)

1. **Remove\_team()**

הוצאת הקבוצה מעץ הקבוצות (הוצאה של איבר מעץ AVL לוקחת log(k) במקרה הגרוע כאשר k הוא מספר הקבוצות במערכת.)

1. **Add\_player:**

הוספת השחקן לעצים הבאים:

* עצי כל השחקנים במערכת (אחד הממוין לפי id והשני לפי שערים וכרטיסים.)  
   (סיבוכיות של log(n) כאשר n הוא מספר השחקנים במערכת)
* עץ השחקנים של הקבוצה שלו (log(n1) כאשר n1 הוא מספר השחקנים בקבוצה שלו (בוודאות קטן מn)
* עץ השחקנים בקבוצה שלו הממוין לפי השערים והכרטיסים (כפי שהוסבר לעיל, סיבוכיות של log(n1)).
* עץ הקבוצות הכשירות לשחק במידה ובהוספת השחקן השתנתה כשירות קבוצתו (זאת בסיבוכיות של log(k) כך שk הוא מספר הקבוצות הכשירות לשחק, כך שk הוא מספר הקבוצות הכשירות לשחק, ובהכרח k<n מכיוון שקיימות לכל היותר n/11 קבוצות כשירות. כי בכל קבוצה כשירה ישנם 11 שחקנים.)

השוואת השחקן לtop\_scorer בקבוצתו ובמערכת והחלפתו בעת הצורך (סיבוכיות של o(1) מכיוון שהמצביעים לשחקן הטוב ביותר שמורים במערכת)

עדכון שכניו של השחקן (השמורים לכל אחד מהשחקנים במערכת כשדות של המחלקה player), זאת בעזרת פונקציה למציאת שכנים הממומשת בAVLTREE.H), פונקציה זו רצה משורש העץ לכל היותר עד אחד מעליו, והיות שגובה העץ הוא log(n) כך שn הוא מספר הצמתים בעץ, סיבוכיות הפונקציה היא גם log(n).

לכן, ככלל, סיבוכיות הוספת השחקן היא o(log(n)) כאשר n הוא משסר השחקנים במערכת.

1. **Remove\_player:**

הוצאת השחקן מארבעת העצים שתוארו לעיל כעצים שאליו מוסיפים את השחקן. בסיבוכיות של o(log(n)) כך שn הוא מזפר השחקנים במערכת וזהו גם מספר השחקנים בעץ הכי גדול מבין הארבעה)

אם הוא היה top\_scorer אזי נחליף את מצביעי הtop\_scorer הרלוונטים (בקבוצה ובמערכת) לשכן של האיבר משמאל (סיבוכיות של o(1) מכיוון שכל אלו שמורים כשדות של השחקן, הקבוצה והמערכת)

עדכון סטטוס הקבוצה באם הוצאת השחקן פגע בכשירות שלה לשחק (סיבוכיות של o(log(k)) במקרה הגרוע שהוסבר לעיל בחלק של הוספת הקבוצה לעץ הקבוצות הכשירות לשחק)

1. **Update\_player:**

שינוי פרטי השחקן עלול לשנות את דירוגו ביחס לשאר בשחקנים ולכן נוציאו תחילה מהעצים הרלוונטים, נעדכן את פרטיו ואז נוסיפו חזרה.  
-זאת בסיבוכיות של log(n) במקרה הגרוע כאשר n הוא מספר השחקנים במערכת, מסיבות שפורטו לעיל עבור הכנסה והוצאה של שחקן לעצים.

עדכון מצב הכשירות של הקבוצה אליה משתייך השחקן (הכנסה או הוצאה מעץ הקבוצות הכשירות לשחק) וזאת בסיבוכיות של log(k) כאשר k הוא מספר הקבוצות הכשירות לשחק, והראינו לעיל מדוע בהכרח log(k) < log(n), כאשר n הוא מספר השחקנים במערכת.

כמו כן, נעדכן את שכניו של השחקן, בסיבוכיות של log(n) כפי שהוצג בסעיפים קודמים, ואת המצביעים לtop\_scorer במידה וחל שינוי עקב עדכון פרטיו של השחקן. זאת בסיבוכיות של o(1) מכיוון שאלו שמורים כשדות של המחלקות השונות.

1. **Play\_match:**
2. **Get\_num\_played\_games:**
3. **Get\_team\_points:**
4. **Unite\_teams:**
5. **Get\_top\_scorer:**
6. **Get\_all\_players\_count:**
7. **Get\_all\_players:**
8. **Get\_closest\_player:**
9. **Knockout\_player:**