יבש לרטוב תרגיל 1 מבני נתונים

מגישים: לירן רדל 314989427

אנה גריגוריבקר 321931396

תיאור מבנה הנתונים:

עץ AVL – AvlTree<Key, Value>: עץ AVL גנרי כפי שנלמד בקורס, הממיין את המידע שנכנס אליו לפי אופרטורי ההשוואה של מחלקת Key, כאשר מבנה הנתונים של העץ מכיל את גודל העץ, כלומר כמה איברים יש בו בשם size, ומצביע לשורש העץ שהוא מסוג AvlNode<Key, Value> .

AvlNode<Key, Value>: צומת בעץ AVL, שמכילה את מפתח ההשוואה Key, ואת המידע שאותו אנחנו רוצים לשמור מסוג Value. כל צומת מכילה את גובהה בעץ, מצביע להורה שלה בעץ, ולבן הימני והשמאלי שלה בעץ.

Pair<Key, Value>: מחלקה בעלת 2 שדות מסוג Key, Value הרלוונטית למימוש מיזוג עצים ולפונקציות המחלקה הראשית.

מבנים פנימיים של התוכנית

Player: מחלקה המייצגת שחקן כדורגל במערכת, בעל שדות פנימיים שיממשו אותנו במימוש הפונקציות: מספר מזהה של השחקן, מספר מזהה של הקבוצה בה משחק, מספר הגולים שהבקיע השחקן, מספר הכרטיסים אותם קיבל השחקן, משתנה בוליאני האומר האם השחקן הינו שוער, מספר המשחקים אותם שיחק השחקן כשהצטרף לקבוצה, מספר המשחקים שהקבוצה שיחקה כאשר הצטרף אלייה, מצביע לקבוצה, מצביע לשחקן הבא InOrder, מצביע לשחקן

הקודם InOrder.

Score: מחלקה שמטרה למיין שחקנים לפי מספר הגולים שהבקיעו. אם שווים, ממיין לפי המספר הנמוך יותר של הכרטיסים שקיבלו. אם שווים, לפי מספר המזהה הגדול יותר. שימושית על מנת ליצור עצים לפי דירוג והשוואת שחקנים לפי הדירוג הרצויי בפונקציות. (כאשר נדבר על דירוג נתכוון ל-Score).

Pair: מחלקת עזר המכילה שדה Key ושדה Value (עקרון דומה למפה) שתעזור לנו במימוש פונקציית מיזוג עצים בעץ AVL.

Team: מחלקה המייצגת קבוצת כדורגל במערכת, בעלת שדות פנימיים שיממשו אותנו במימוש הפונקציות: מספר המזהה של הקבוצה, מספר הנקודות של הקבוצה, שדה value שבעצם הוא הנוסחא: $\sum_{Players} goals - cards$ שמתעדכן עם כל הוצאת\הכנסת שחקן, מספר השוערים בקבוצה, דירוג של השחקן הכי טוב בקבוצה, מצביע לשחקן הכי טוב בקבוצה, עץ הממויין לפי מספר המזהה של שחקנים של כל השחקנים בקבוצה, ועץ הממויין לפי דירוג של כל השחקנים בקבוצה.

Worldcup23a1: המחלקה הראשית שמכילה את כל הפונקציות הרצויות, המכילה את השדות: עץ הממויין לפי מספר מזהה של קבוצות שמכיל את כל הקבוצות במערכת, עץ הממויין לפי מספר מזהה של קבוצות שמכיל את כל הקבוצות החוקיות במערכת (אילו עם לפחות 11 שחקנים ושוער אחד), עץ הממויין לפי מספר מזהה של שחקנים שמכיל את כל השחקנים במערכת, עץ ממויין לפי דירוג שמכיל את כל השחקנים במערכת, הדירוג של השחקן הכי טוב במערכת ומצביע לשחקן הכי טוב במערכת.

סיבוכיות המקום של מבני הנתונים:

במחלקת השחקן יש רק משתנים פרימיטיביים ומצביעים ולכן סיבוכיות המקום שלה היא $O(1)$.

במחלקת הקבוצה יש משתנים פרימיטיביים, מצביעים ו-2 עצי AVL שכל אחד מהם מכיל את כמות השחקנים בקבוצה, לכן סיבוכיות המקום הינה $O(2n_{TeamID}) \approx O(n_{TeamID})$.

מחלקת הדירוג מכילה 2 משתנים שסיבוכיות המקום שלהם היא $O(1)$ ולכן בסה"כ $O(1)$.

המחלקה הראשית מכילה מצביע ומשתנה פרימיטיבי ו-4 עצים –

2 עצים של כל העובדים במערכת במיונים שונים לכן סיבוכיות המקום היא $O(2n) \approx O(n)$

ו-2 עצים, אחד של כל הקבוצות המערכת ואחד של כל הקבוצות החוקיות במערכת, לכן סיבוכיות המקום היא: $O(k + k_{Legal}) \approx O(k)$.

לכן בסה"כ סיבוכיות המקום של מבני הנתונים הינה $O(n + k)$.

תיאור הפונקציות:

***world_cup_t()*:** נאתחל את ארבעת העצים להיות ריקים בעזרת הקונסטרקטור הדיפולטיבי שלהם, את הדירוג של השחקן המירבי להיות דיפולטיבי (שהכל אפסים) ואת המצביע לשחקן הכי טוב במערכת להיות *nullptr* בסה"כ סיבוכיות של $O(1)$.

***~world_cup_t()*:** נעבור על כל הפרמטרים במבני נתונים, ונמחק אותם מן הזכרון. את הפרימיטיביים זה יקרה ב-$O(1)$, ומחיקת העצים תעשה במעבר על כל הצמתים של העצים ומחיקתם, כאשר הסברנו בסיבוכיות המקום שיש $O(n + k)$ צמתים במבני הנתונים כאשר מחיקת כל צומת קוראת ב-$O(1)$ ולכן הסיבוכיות בסה"כ הינה $O(n + k)$ כנדרש.

add_team: ראשית נבדוק שהקבוצה לא קיימת כבר במערכת על ידי חיפושה בעץ הקבוצות הכללי, פעולה שקוראת ב-$O(logk)$ כפי שנלמד בתרגול. אם קיימת נחזיר כשלון. אחרת, נוסיף קבוצה חדשה ריקה שרק הפרמטרים של מספר הנקודות והמספר מזהה שלה מאותחלים, ונאתחל 2 עצים ריקים בשדות של העצים של הקבוצה בסיבוכיות $O(1)$. לאחר מכן נוסיף את הקבוצה אל עץ הקבוצות הכללי, שלפי מה שנלמד בהרצאה קורה בסיבוכיות $O(logk)$ ואז נחזיר הצלחה. לכן בסה"כ הפעולה קוראת בסיבוכיות של $O(logk)$.

$remove\_team$: ראשית נבדוק האם הקבוצה קיימת במערכת, על ידי חיפושה בעץ

הקבוצות הכללי, פעולה שקוראת ב-$O(logk)$ כפי שנלמד בתרגול. לאחר מכן נבדוק כי הקבוצה ריקה על ידי בדיקת גודל עץ השחקנים שלה, שמכיל את כמות השחקנים בו. בדיקה שקוראת ב-$O(1)$, ואם קיימים שחקנים נחזיר כשלון. אם לא נמצאה, נחזיר כשלון. אחרת, נסיר אותה מן עץ הקבוצות הכללי בסיבוכיות $O(logk)$ כפי שנלמד בהרצאה, ונבדוק האם הקבוצה חוקית, ואם כן נסירה גם מעץ הקבוצות החוקיות בסיבוכיות של $O(\log k_{TeamID})$, ואז נמחק את אובייקט הקבוצה מן המערכת בסיבוכיות של $O(1)$, מאחר והעצים בהכרח ריקים ושאר המשתנים פרימיטיביים ואז נחזיר הצלחה. לכן בסה"כ הפעולה קוראת בסיבוכיות של $O(logk)$, כנדרש.

<u>$add\ player$:</u> ראשית נבדוק כי השחקן עם מספר המזהה שלו שיחודי רק לו נמצא כבר במערכת על ידי חיפושו בעץ השחקנים הכללי. אם קיים, נחזיר כשלון. חיפושו קורה ב-$O(logn)$ כפי שנלמד בהרצאה. לאחר מכן נחפש את הקבוצה אלייה ישתייך בעץ הקבוצות הכללי באותו האופן, ואם הקבוצה אינה קיימת נחזיר כשלון, וזה בסיבוכיות של $O(logk)$. לאחר מציאת הקבוצה נעדכן את השדות הרלוונטים שלה לפי השדות של השחקן, ונעדכן את השדות הרלוונטים של השחקן ב-$O(1)$. לאחר מכן, נבדוק האם דירוג השחקן החדש הוא יותר טוב מזה של השחקן הכי טוב בקבוצה ובמערכת על ידי השוואת שדה הניקוד הקיים בקבוצה ובמערכת(לאחר מציאת הקבוצה), ואם כן נעדכן את השדות הרלוונטים. מדובר על פעולות בודדות שלוקחות $O(1)$. לאחר מכן נכניס את השחקן ל-2 העצים של הקבוצה בסיבוכיות של $O(logn_{TeamID})$ כפי שנלמד בהרצאה, ול-2 העצים של השחקנים הכללי במערכת באותו האופן בסיבוכיות של $O(logn)$. לאחר מכן נבדוק האם חוקיות הקבוצה השתנתה, כעת מאחר ונוסף שחקן היא יכולה להיות חוקית, ואם כן נכניסה אל עץ הקבוצות החוקיות בסיבוכיות של $O(logk_{Legal})$. לבסוף, נחפש את האיברים הבאים InOrder לשחקן שלנו, כלומר האיברים הצמודים אל השחקן במערך הInOrder שנקבל אם נבצע פעולת InOrder על עץ השחקנים הכללי לפי דירוג, בעזרת פונקצית עזר שנמצאת בעץ. חיפוש השחקן הבא בתור InOrder לוקחת לנו $O(logn)$, והיא עובדת בצורה הבאה:(נתאר עבור next InOrder, עבור prev האלגוריתם דומה: נבדוק את הבן הימני של האיבר בעץ. אם קיים, אז נלך אל הבן הימני ונחזיר את האיבר המינימלי בתת העץ הזה. אחרת, האיבר הבא בתור הוא אחד האבות של האיבר, לכן נטייל מעלה המעלה העץ עד שנתקל באיבר שהוא הבן הימני של ההורה שלו, והוא בהכרח האיבר שאנו מחפשים. מאחר ובמקרה הגרוע מטייל מן עלה אל השורש, פעולה זאת תקרה בסיבוכיות של $O(logn)$, כנדרש. לאחר מציאת השחקנים נעדכן את השדות של השחקן הבא InOrder והשחקן הקודם InOrder של השחקן שהכנסנו, ואז נחזיר הצלחה. לכן בסה"כ סיבוכיות פעולת הכנסת שחקן חדש למערכת לוקחת לנו:

$$(logk) + O(logn) + O(1) + O(logn_{TeamID}) + O\left(k_{Legal}\right) = O(logn + logk)$$

<u>$\boldsymbol{remove\ player}$:</u> ראשית נבדוק האם השחקן קיים כבר המערכת, על ידי חיפושו בעץ השחקנים הכללי בסיבוכיות של $O(logn)$ כפי שנלמד בהרצאה. אם לא נמצא נחזיר כשלון. לאחר מציאתו, נלך לקבוצה אלייה שייך בעזרת המצביע לקבוצה אותו מחזיק, ונשנה את ערך הקבוצה, ונסירו מן עצי השחקנים של הקבוצה בסיבוכיות של $O(logn_{TeamID})$י שנלמד בהרצאה, ובאותו האופן נסירו מן עצי

השחקנים הכלליים בסיבוכיות של $O(logn)$. לאחר מכן נבדוק האם הקבוצה הקבוצה של השחקן הייתה חוקית פעם וכעת אינה, ואם אינה יותר נסירה מן עץ הקבוצות החוקיות בסיבוכיות של $O(logk_{Legal})$שתמש בעובדה כי יש קשר ישיר בין מספר השחקנים הכללי לבין מספר הקבוצות החוקיות, מאחר וכל קבוצה חוקית מכילה לפחות 11 שחקנים, ולכן מתקיים $n \geq 11k_{Legal}$, ולכן בהכרח מתקיים לפי הגדרה:$O(logk_{Legal}) = O(logn)$. לאחר מכן נבדוק האם השחקן שהסרנו היה השחקן הכי טוב בקבוצה שלו\ במערכת, ואם כן נחפש את האיבר המקסימלי בעץ השחקנים של הקבוצה לפי דירוג\עץ השחקנים הכללי לפי דירוג בהתאמה ונשימו כהשחקן הכי טוב בהתאמה ולפי הצורך. חיפוש איבר מקסימלי בעץ משמעותו ללכת לבן הכי ימני בעץ, כלומר קורה בסיבוכיות של $O(logn)$ או $O(logn_{TeamID})$ גם וגם (כשאנו יודעים בכל מקרה כי $n \geq n_{TeamID}$). ולבסוף, נלך את השחקן הבא *InOrder* והשחקן הקודם *InOrder* (אם קיימים ואינם *nullptr*, כלומר השחקן שהוסר הינו עם הדירוג הכי נמוך או גבוה או היחידי במערכת) ועבור השחקן הבא *InOrder* נשנה את השחקן הקודם *InOrder* שלו להיות השחקן הקודם *InOrder* של השחקן שהסרנו, וההיפך נבצע עבור השחקן הקודם *InOrder* של השחקן אותו אנו מסירים, מאחר ואם נסתכל על זה כרשימה מקושרת זה אותו עקרון כמו להסיר איבר ו"שינויי מצביעים", בסיבוכיות $O(1)$, ואז נחזיר הצלחה. כל הפעולה לוקחת בסה"כ:

$O(logn_{TeamID}) + O(logn) + O(1) = O(logn)$, כנדרש.

***update player stats*:** ראשית נבדוק שהשחקן קיים במערכת על ידי חיפושו בעץ השחקנים הכללי בסיבוכיות $O(logn)$ כפי שנלמד בהרצאה, ואם אינו נמצא אז נחזיר כשלון. לאחר מכן נוציא את השחקן מן העץ של כל השחקנים בקבוצה לפי דירוג, ומן העץ של כל השחקנים הכללי לפי דירוג, מאחר וכעת נעדכן את דירוג השחקן. לאחר הוצאת השחקן משם, נעדכן את הסטטיסטיקה שלו (מספר הכרטיסים, גולים ומשחקים ששיחק בעצמו) ב-$O(1)$, ונכניסו שוב לתוך העצים שהוצאנו אותו משם. הכנסה והוצאה של השחקן קוראת בסיבוכיות כפי שנלמדה בהרצאה של-$O(logn) + O(logn_{TeamID})$

לאחר מכן נבדוק האם היה השחקן הטוב ביותר, והאם נשאר כזה, ואם לא היה האם הוא כעת לפי המצביעים בקבוצה שלו ושל המערכת, ונעדכן בהתאם, ב-$O(1)$. נעדכן גם את ערך הקבוצה מאחר וסטטיסיטקת השחקן השתנתה ב-$O(1)$. לבסוף, באותו אופן כמו בפעולת ההכנסה כפי שהאלגוריתם הוסבר שם, נעדכן את שדות השחקן הבא *InOrder* והקודם *InOrder* מאחר ומיקומו בעץ השתנה(יש לציין שבהסרת השחקן מן העצים גם עדכנו באותו האופן כמו בפעולת ההסרה מהעץ) בעזרת פעולת העזר שהסוברה מקודם, ואז נחזיר הצלחה. בסה"כ קורה בסיבוכיות של $O(logn_{TeamID}) + O(logn)$.

לכן פונקציה זאת קוראת סבה"כ בסיבוכיות של: $O(logn_{TeamID}) + O(logn) + O(1) = O(logn)$.

***play match*:** ראשית נבדוק האם הקבוצות קיימות המערכת על ידי חיפושן בעת הקבוצות הכללי. אם אחת מהן אינה נמצאה או שתיהן, נחזיר כשלון. לפי מה שנלמד בהרצאה קורה בסיבוכיות של $O(logk)$. לאחר מכן נבדוק האם הקבוצות חוקיות על ידי בדיקה שיש להן לפחות 11 שחקנים ושוער אחד בעזרת שדה של מספר השוערים בקבוצה ובדיקת גודל עץ השחקנים(שדה שקיים בעץ ותמיד מעודכן וניתן לגשת אליו ישירות), ונוושא ששתיהן חוקיות אחרת נחזיר כשלון. (לכן $O(1)$). לאחר מכן נחשב עבור כל קבוצה את הערך שלה, העזרת השדה *value* של כל קבוצה כפי שתואר בפירוט מבני הנתונים, שהוא

הסיגמה של השחקנים שתמיד מעודכן, ובעזרת מספר הנקודות של השחקן. חישוב זה לכל קבוצה לוקח $O(1)$. לאחר נכן נשווה בין ערכי הקבוצות, ובהתאם לבקשת הפונקציה נעדכן את מספר הנקודות של כל קבוצה ונוסיף 1 למספר המשחקים ששיחקה, עדכון שדות שניתן לגשת אליהן ישירות וזה גם קורה ב-$O(1)$, ולבסוף נחזיר הצלחה. לכן פונקציה זאת בסה"כ קוראת בסיבוכיות של:$= O(logk) + O(1)$ $O(logk)$, כנדרש.

<u>get num played games:</u> ראשית נבדוק האם השחקן עם המזהה המתקבל קיים במערכת. אם לא נחזיר כשלון. פעולה זאת קוראת בסיבוכיות של $O(logn)$ בחיפוש על עץ השחקנים הכללי לפי מספר מזהה של השחקנים כפי בנלמד בהרצאה. לאחר מציאת השחקן, נלך אל השדה של הקבוצה של השחקן(שמחזיק מצביע אלייה) של מספר המשחקים שהקבוצה שיחקה מאז שנוצרה, ונבצע ב-*O(1)* את החישוב הבא שייתן לנו את מספר המשחקים העדכני של השחקן: מספר המשחקים של השחקן ששיחק כשנוצר + מספר המשחקים של הקבוצה מאז שנוצרה – מספר המשחקים שהקבוצה שיחקה כשהשחקן הצטרף אלייה, ואז נחזיר מספר זה, ולבסוף נחזיר הצלחה. בסה"כ סיבוכיות של $O(logn)$.

<u>get team points:</u> נחפש את הקבוצה בעץ הקבוצות הכללי בסיבוכיות של $O(logk)$ כפי שנלמד בהרצאה, ואז ניקח את השדה של מספר הנקודות של הקבוצה ב-$O(1)$ ונחזירו, ובסוף נחזיר הצלחה. בסה"כ סיבוכיות של $O(logk)$ כנדרש.

<u>unite teams:</u> ראשית נבדוק שהקבוצות לא קיימות על ידי חיפושן בעץ הקבוצות הכללי בסיבוכיות של $O(logk)$ כדי שנלמד בהרצאה (במקרה של אי מציאת אחת מהן או שהקבוצה עם המזהה החדש קיימת ואינה אחת מ-2 הקבוצות נחזיר כשלון). לאחר מציאתן, נאחד אותן לתוך קבוצה חדשה עם המזהה החדש (אם המזהה הוא של אחת הקבוצות אז נחלשב לזה בהתאמה) בצורה הבאה:

ערך הקבוצה וניקודה יהיה סכום של 2 הקבוצות ביחד, כך גם מספר השוערים ונעדכן את השחקן הכי טוב בקבוצה בהתאמה, פעולות אלה ב-$O(1)$. לאחר מכן נאחד את העצים של השחקנים בכל קבוצה (העץ לפי המזהה של השחקנים והעץ לפי דירוג השחקנים) באופן שבו נלמד בתרגול בסיבוכיות של $O(n_{Team1ID} + n_{Team2ID})$. לאחר מכן נבדוק האם הקבוצה החוקית(מאחר ויש מצבים שבהם נוספו שחקנים ושוערים כשמקודם לו היו) ואם כן נכניסה אל עץ הקבוצות החוקיות בסיבוכיות של $O(logk_{Legal})$, ובסוף נחזיר הצלחה. בסה"כ הפעולה קוראת בסיבוכיות של:

$O(logk + n_{Team1ID} + n_{Team2ID})$, כנדרש.

<u>get top scorer:</u> נחלק למקרים: אם $teamID > 0$, אז נחפש את הקבוצה בעץ הקבוצות הכללי בסיבוכיות של $O(logk)$ כפי שנלמד בהרצאה. אם לא נמצאה, נחזיר כשלון. לאחר מציאתה נבדוק שיש שחקנים על ידי בדיקת גודל העץ של כל השחקנים בקבוצה, ואם אין שחקנים נחזיר כשלון. אם קיימים שחקנים, נחזיר את השדה בקבוצה של השחקן עם הניקוד הכי גבוה שדאגנו לתחזק ב-$O(1)$, לכן

$O(logk)$ בסה"כ עבור מקרה זה. אחרת, אם $teamID < 0$, נבדוק האם יש שחקנים לפי גודל העץ של כל השחקנים, ואם אין נחזיר כשלון. אחרת באותו האופן, נחזיר את המזהה של השחקן עם הניקוד המירבי בעזרת השדה במחלקה הראשית שלנו שדאגנו לתחזק ב-$O(1)$, כנדרש.

<u>get all players count:</u> נחלק למקרים: אם $teamID > 0$, אז נחפש את הקבוצה בעץ הקבוצות הכללי לבדוק שהיא קיימת בסיבוכיות של $O(logk)$ כפי שנלמד בהרצאה, ואם לא נמצאה אז נחזיר כשלון. אחרת, פשוט נחזיר את גודל העץ (כפי שהוסבר גודל שתוחזק וניתן לגשת אליו ב-$O(1)$) של כל השחקנים בקבוצה, לכן $O(logk)$ בסה"כ עבור מקרה זה. אחרת אם $teamID < 0$, באפן דומה ניגש אל עץ כל השחקנים הכללי של המחלקה הראשית ונחזיר את גודלו ב-$O(1)$, כנדרש.

<u>get all players:</u> נחלק למקרים: אם $teamID > 0$, אז נחפש את הקבוצה בעץ הקבוצות הכללי לבדוק שהיא קיימת בסיבוכיות של $O(logk)$ כפי שנלמד בהרצאה, ואם לא נמצאה אז נחזיר כשלון. אחרת, נבצע מעבר *InOrder* על עץ כל השחקנים בקבוצה המדורג לפי הדירוג שמבוקש בפונקציה שדאגנו לתחזק לתוך המערך שהוקצה כפי שנלמד בהרצאה בסיבוכיות של $O(n_{TeamID})$, ולכן בסה"כ סיבוכיות של $O(logk + n_{TeamID})$. אחרת אם $teamID < 0$, נבצע מעבר *InOrder* על עץ כל השחקנים במערכת המדורג לפי הדירוג שמבוקש בפונקציה שדאגנו לתחזק לתוך המערך שהוקצה כפי שנלמד בהרצאה בסיבוכיות של $O(n)$ ככמות השחקנים הכללית במערכת, ולכן בסה"כ $O(n)$ כנדרש.

<u>get closest player:</u> ראשית, נבדוק שהקבוצה קיימת במערכת על ידי חיפושה בעץ הקבוצות הכללי בסיבוכיות של $O(logk)$ כפי שנלמד בהרצאה, ואז נחפש את השחקן בעץ כל השחקנים לפי מזהה בקבוצה באותו האופן בסיבוכיות של $O(logn_{TeamID})$. אם השחקן ו\או הקבוצה לא נמצאו, נחזיר כשלון. נבדוק גם האם זה השחקן היחידי בכל המערכת על ידי בדיקת גודל עץ כל השחקנים במערכת ב-$O(1)$, ואם כן נחזיר כשלון. לאחר מכן, נשווה בין השדות של השחקן של השחקן הבא *InOrder* והשחקן הקודם *InOrder* מי מהם יותר קרוב לפי הסטטיסטיקה של השאלה אל השחקן, ונחזיר את מזהה השחקן המתאים, בדיקה שלוקחת פעולות בודדות ולכן הסיבוכיות שלה היא $O(1)$. (מאחר וכפי שפורט בפעולת הכנסת שחקן, אנו דואגים תמיד למצוא את השחקן הבא והקודם *InOrder* על עץ השחקנים הכללי לפי דירוג, כלומר את השחקנים ששניהם בפוטנציאל הכי קרובים אל השחקן שלנו כשאנחנו מתחשבים בעובדה שנלמדה בהרצאה כי מעבר *InOrder* ממיין את העץ לפי גודל באופן של "מערך", לכן השחקן הבא ב"מערך" והקודם ב"מערך" הינם השחקנים שהכי קרובים מבחינת הדירוג שלפיו מדורג העץ אל השחקן הרצויי. אם אחד מהשחקנים שאליו אנו מצביעים הוא *nullptr*, פשוט נחזיר את מזהה השחקן השני, ומובטח לנו שתמיד יהיה אחד מאחר ויש לנו לפחות 2 שחקנים).

לכן בסה"כ סיבוכיות הפעולה הינה: $O(logk + logn_{TeamID})$, כנדרש.

<u>knockout_winner</u> **:** ע"מ לממש את הפונ' תחזקנו במחלקה *world_cup* עץ *AVL* ובו רק הקבוצות הכשרות לפי הגדרת הפונ', נשים לב שגודל העץ, נסמן אותו ב-*x* חסום ע"י מס' כל הקב' וע"י מספר כל השחקנים חלקי 11, לכן$x \le \min\{\frac{n}{11}, k\} \leftarrow \log x \le \log\min\{\frac{n}{11}, k\}$ מכאן שכל חיפוש בעץ שנעשה באופן שלמדנו בהרצאה יעשה ב$O(\log\min\{n, k\})$ נרצה למלא מערך ובו כל הקב' הכשרות בטווח הרצוי מסודרות בסדר עולה לפי המזהה שלהן, נעשה זאת ע"י פונ' עזר *limitedInOrder* אשר מחפשת את האיבר הראשון שנמצא בטווח ע"י חיפוש בעץ בינארי, ואז קוראת לפונ' רקורסיבית שמקבל איבר בעץ אשר נמצא בטווח ואת הטווח ומשבצת את האיברים במערך בהתאם, אם נגיע לאיבר שאינו בטווח הפונ' בודקת אם יש איבר בטווח הרצוי בבן השמאלי של האיבר(ע"י חיפוש בעץ בינארי)(אם הוא גדול מהטווח), אם כן היא קוראת לעצמה שוב עם הבן השמאלי,באופן דומה נבדוק את הבן הימני אם האיבר קטן מהטווח. לאחר החיפוש ההתחלתי שנעשה ב$O(\log\min\{n, k\})$, הפונ' הרקורסיבית תבקר ב-*r* איברים עקב כך שאין קריאות מיותרות בגלל שלפני כל קריאה בדקנו אם יש בכל תת עץ איבר בטווח (מה שנעשה באמצעות חיפוש בעץ *AVL* כמו שלמדנו בהרצאה), נשים לב שבדיקה האם יש איבר בטווח בתת עץ של איבר שאינו בטווח תעשה לכל היותר כמות סופית של פעמים. כעת, נשתמש בפונ' אשר מקבלת מערך של קבוצות שמשחקות בסיבוב, מסודרות לפי מזהה, ומערך ריק ובו ישובצו המנצחות בסיבוב. נעבור בלולאה על כל הקבוצות ונשווה כל זוג לפי הפרמטרים בסיבוכיות *O(r)* , את הקבוצה המנצחת נשבץ במקום הרצוי במערך המנצחות. לאחר מכן נקרא שוב לפונ' ברקורסיה, כעת מערך המנצחות הופך למערך הקב' שמשחקות בסיבוב, נשים לב שבכל קריאה לפונ' כמות הקבוצות תקטן פי 2 עד שנשאר עם קבוצה אחת, מכאן $T(n) = n + T(n/2)$

ו- $T(1) = 1$ ע"י הצבות חוזרות ונשנות ושימוש בנוסחה של טור הנדסי מתכנס נקבל $T(n) = 2n$

כלומר הסיבוכיות של הפונ' היא ב- $O(r)$, סה"כ נקבל שהסיבוכיות הכוללת היא $O(\log(\min\{n, k\}) + r)$ כנדרש.