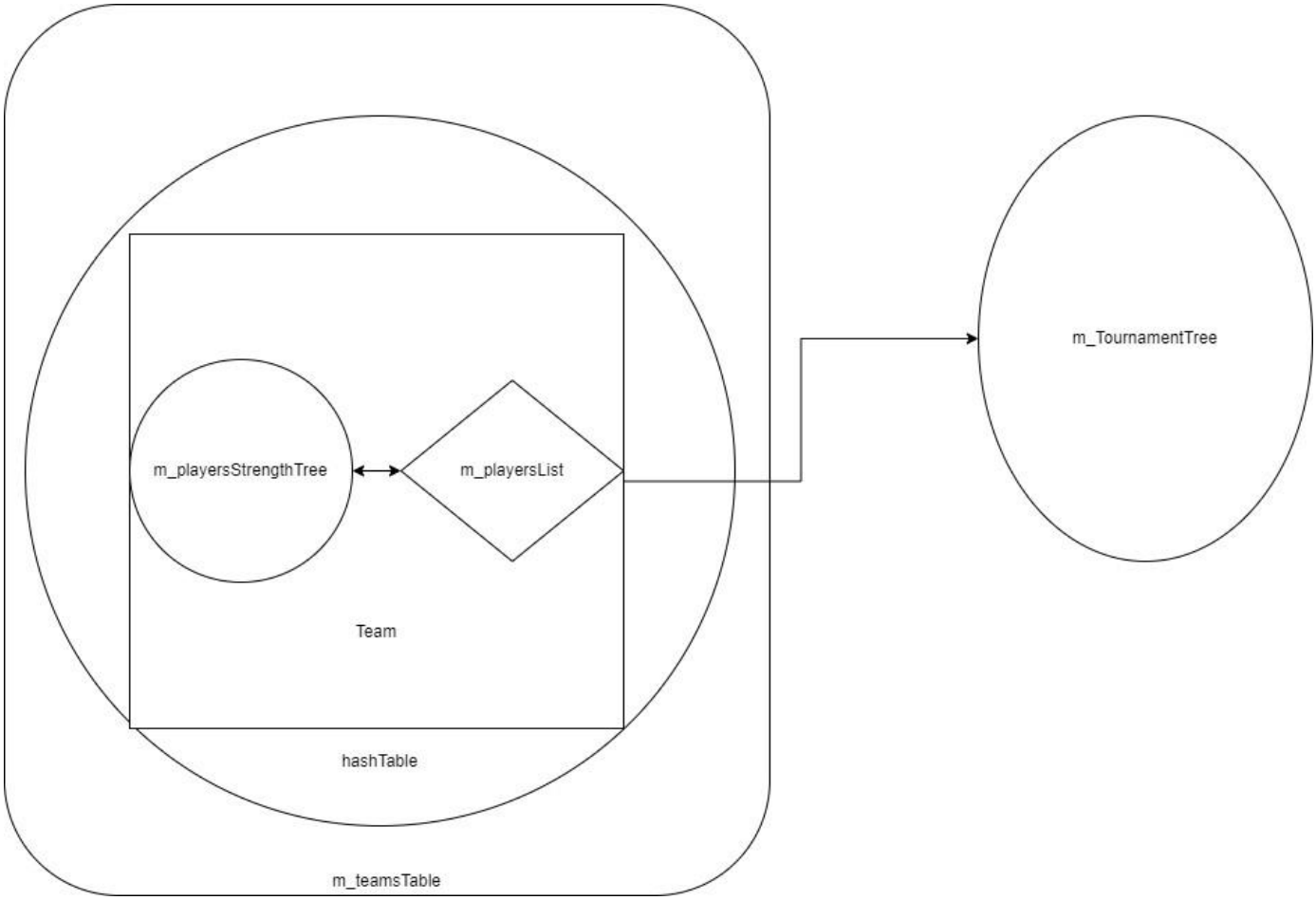
Dry2-wet2

נתחיל בתרשים של מבני הנתונים שנחזיק:



מבנה הנתונים olymipcs_t יורכב בצורה הבאה:

- m_teamsTable- טבלת ערבול המכילה את הנבחרות יפורט בהמשך
- m_tournamentTree- עץ דרגות אשר בו נעדכן את האיברים בו למקרה שנצטרך לבצע טורניר
- m_HighestRank- דירוג הנבחרת הגבוהה ביותר אותו נתחזק בעזרת עץ הטורניר
- m_numOfTeam-מספר הנבחרת הכולל במערכת

נחזיק מחלקה אשר נקרא לה contestant שבה נייצר שחקנים חדשים האיבר יחזיק את הכוח והid הייחודי(נפרט בהמשך) של השחקן

בנוסף לכך נחזיק את המחלקה Team אשר תכיל את הדברים הבאים:

- m_playersList- רשימה מקושרת שבעזרתה נוציא את השחקן האחרון שנכנס
- m_playersStrength- עץ דרגות של הכוחות של השחקנים בנבחרת
- m_myTournamentTeam- האובייקט של הנבחרת בעץ הטורניר
- m_teamPower- הכוח של הנבחרת
- m_teamRank- הדירוג של הנבחרת
- m_numOfWins- מספר הניצחונות של הנבחרת

- m_teamCounter- מונה אשר בעזרתו ניתן id ייחודי לכל שחקן בנבחרת כאשר נכנס שחקן חדש נגדיל אותו באחד וכאשר יצא שחקן נקטין באחד (באיחוד נטפל בו אחרת)
- m_numOfPlayersInTeam-מספר השחקנים בקבוצה

כעת נסביר על החוליות בעץ הדרגות שבנינו כל איבר בעץ בנוסף ל key וvalue ושדות שאנחנו מחזיקים לתחזק עץ avl תקין נחזיק שדות לתחזוק עץ הדרגות:

- m_amount- מספר האיברים בתת העץ של האיבר יעזור לנו למצוא את האינדקס של כל איבר בעץ (נשתמש בכך לעץ טורניר ועץ כוחות של נבחרת)
- M_extra- כמות הניצחונות שנרצה להוסיף לתתי העצים של האיבר (נשתמש בכך לעץ הטורניר)
- M_maxRankBeforeExtras- הדירוגים המקסימלי של תתי העצים של האיברים ללא תוספת האקסטרה לתתי העצים(נשתמש בכך לעץ טורניר)
- M_maxRankAfterExtras- הדירוג המקסימלי של תתי העצים של האיבר לאחר תוספת האקסטרה הדרושה לתת העץ(נשתמש בכך לעץ טורניר)
- נשים לב כי m_maxRankAfterExtras בשורש העץ יכיל את הדירוג המקסימלי לאחר הוספת אקסטרה של כל הנבחרות כי כולם נמצאים בתתי העצים שלו לכן הוא תמיד יכיל את הדירוג המקסימלי של הנבחרות
- M_myRank- הדירוג איתו האיבר נכנס לטורניר (נשתמש בכך לעץ טורניר) לא מתעדכן בגלגולים
- M_sumTill- סכום האקסטרות עד הצומת (נשתמש בעץ טורניר)

נסביר על אופן שימור השדות שמשתנים בעת הוספה והסרה :

- עבור הוספה: עבור הוספה נוסיף את האיבר החדש כמו שהוסבר לנו בשיעורים כעלה כאשר נגיע למקומו נכניס אותו עם ערך אקסטרה של אפס וsumTill=0 וm_maxRankAfter=m_maxRankBefore=m_myRank כלומר כל הדירוגים יהיו שווים לדירוג הכניסה שלו כי לא השתתף בטורניר כעת נעלה במעלה מסלול החיפוש עד שנגיע לשורש בכל איבר נבדוק האם maxRankAfter וmaxRankBefore השתנו ביחס להכנסה החדשה ונפמפם את זה למעלה
- עבור החסרה בהתחלה נבצע חפוש עבור האיבר המוסר במהלך החיפוש נסכום את כל ערכי האקסטרה עד אותו איבר זה ישמש אותנו לעדכון לאחר ההסרה כעת כי שלמדנו בשיעורים נחפש את האיבר העוקב שיוכל להתחלף איתו נשים לב כי בעת ההחלפה תיהיה השפעה על הבנים של אותו איבר וגם לא נרצה שתיהיה השפעה על סכום האקסטרות שיקבלו תתי העצים של האיבר המוסר לכן בעת החיפוש עבור האיבר המחליף נבצע סכימה של ערכי האקסטרה עד אותו איבר המחליף ובעת ההחלפה נעדכן את ערך האקסטרה כך שהמחליף יקבל את ההפרש באקסטרות ונחסיר את זה מתתי העצים של המוחלף כי לא היו לקבל את ההפרש. נשים לב כי גם במידה ויש בנים לאיבר המחליף הם יושפעו מכך שיחליף את מקומו לכן נוסיף להם את ערך האקסטרה של המחליף לפני ההחלפה כדי שישאר להם ערך אקסטרה תקין , ביחס לsumTill המחליף יקבל את הערך של האיבר המוסר ונעדכן בהתאם את הבנים של האיבר המחליף כי הם מקבלים תוספת לאקסטרה אז נחסיר את זה מsumTill שלהם.כעת לאחר ההחלפה נבצע עליה למעלה מהמקום החדש של האיבר המוסר נסיר אותו ונעדכן עבור כל איבר את maxRankAfterוmaxRankBefore עד שנגיע לשורש.
- עבור גלגולים עבור האיבר המתגלגל נבדוק נוסיף את ערכו לערך האקסטרה של ההורה שלו ונסיר ממנו את ערך אקסטרה הישן של ההורה שלו (לפני הוספה) עבור גלגול rightRotating אם יש לאיבר בין ימני נוסיף לו את הערך האקסטרה הישן של ההורה של האיבר המתגלגל גם כן במידה וזה גלגול leftRotating נבצע זאת לבן שמאלי במידה ויש את שאר השדות נסביר נתחזק כפי שיתואר למטה.

ניתן הסבר קצר לאופן עדכון השדות בגלגולים והוספה\הסרה גם באופן הבא

- נבצע סכימה של כל האקסטרה עד אותו איבר שמכניסים או מוציאים נקרא לסכום sumExtra
- בכל איבר במסלול החיפוש נבצע עדכון שלm_sumTill כך שיהיה sumExtra עד אותה צומת ללא האקסטרה של האיבר הנוכחי
- כאשר נגיע לאיבר במידה והוא מוסר ניגש למצביע שיש לנו לTeam ונעדכן את מספר הניצחונות שלנו לm_numOfWins+sumExtra
- נעדכן m_extra/m_sumTill כפי שהוסבר מעלה
- נעדכן את m_amount כך ש m_amount=node->Right.m_amount+node->Left.m_amout+1
- נעדכן את maxRankBefore=max(myRank,node->Left.maxRankBefore,node->Right.maxRankBefore)

- נעדכן m_maxRankAfter=max(updatedRank,node->Right.maxRankAfter,node->Left.maxRankAfter)
- נמשיך כך כל עוד נגיע לשורש כלומר נבצע שני מסלולי חיפוש ועוד כמות מסויימת של פעולת בזמן קבוע כלומר סה"כ נקבל סיבוכיות זמן O(log(n)) במקרה הגרוע כאשר n מספר האובייקטים בעץ

עבור טבלת הערבול נסביר את הפעולות המרכזיות בה נתחיל ונאמר כי עבור התנגשויות אנו נחזיק עץ בכל תא בטבלה עבור הנבחרות המתנגשות:

עבור ההסבר נגדיר a_factor שהוא פקטור העומס

- עבור הכנסה נבצע בעזרת פונקציות הערבול מודולו שבה בחרנו נבדוק איפה צריך להכניס נבחרת חדשה ונכניס אותה לתא שלה ונוסיף אותה נעדכן את פקטור העומס בהתאם O(1) במקרה הגרוע
- נבדוק אם פקטור העומס גדול או שווה 1 במידה וכן הטבלה מלאה ויש להגדיל אותה O(1) במקרה הגרוע
- כעת נגדיל את הטבלה וניצור טבלה חדשה ע"פ ההרצאה הקצאת מערך היא בO(1) במקרה הגרוע
- נבצע rehashing שבו נעביר את האיברים לעצים חדשים בטבלה החדשה עבור כל נבחרת למקום החדש שלה כי פונקציית המודולו השתנה לפי השיעורים סיבוכיות הזמן המשוערכת היא O(1) כל עוד נשמור על פקטור עומס קטן מאחד
- נעדכן שדות בהתאם למעבר O(1) במקרה הגרוע

לכן סה"כ נקבל עבור הכנסה לטבלת הערבול זמן משוערך של O(1)

- עבור הוצאה נבצע את אותו הדבר רק שבבדיקה נבדוק אם מספר הנבחרות הכולל הוא קטן שווה לרבע ממספר התאים בטבלה במידה וכן נקטין את הטבלה ונבצע rehashing בהתאם

לכן סה"כ גם עבור הוצאה נקבל זמן משוערך של O(1)

- מבחינת סיבוכיות מקום נשים לב כי בזמן rehashing בהעברה נחזיק שני טבלאות ערבול כך שגודל כל אחת הוא O(n) כאשר n מספר הקבוצות וכמו כן מערך בגודל מספר התאים בטבלת הערבול הישנה לפני הרחבה/הקטנה את טבלת ערבול הישנה ואת המערך נמחק בסוף פעולת הreHasing וסיבוכיות המקום תישאר בהתאם

- נחזיק גם רשימה מקושרת עבור השחקנים היא תיהיה רשימה מקושרת דו כיוונית כך שנחזיק את הראש והזנב שלה ולכל איבר ברשימה יהיה מצביע לאובייקט Contestant שהוא שייך לא ומצביע לאיבר לפניו ואחריו ברשימה

כעת נעבור להסבר על הפעולות במבנה olympics_t

<u>Olympic_t()</u>

- נאתחל את טבלת הערבול m_teamsTable איתחלו הטבלה הוא בזמן קבוע כי נאתחל את הטבלה בגודל קבוע לכן O(1) במקרה הגרוע
- נאתחל את עץ הטורניר לפי השיעורים איתחול טבלה הוא בO(1) במקרה הגרוע
- נאתחל משתנים בO(1) במקרה הגרוע

לכן סה"כ נקבל סיבוכיות זמן O(1) במקרה הגרוע

<u>~olympics_t()</u>

- נקרא לדיסטרקטור דיפולטי אשר יקרא לדיסטרקטור של טבלת הערבול ושל עץ הטורניר

- עבור הטבלה הוא ימחוק תחילה את השחקנים שנמצאים בנבחרות מעץ הכוחות ורשימת השחקנים כאשר k מספר השחקנים במערכת אזי סה"כ O(2k) לכן סה"כ סיבוכיות זמן O(k)
- כעת יעבור על הנבחרות בטבלה וימחק אותן כאשר יש n נבחרות במערכת לכן סה"כ סיבוכיות זמן O(n)
- כמו כן ימחק את האיברים מעץ הטורניר כאשר לכל נבחרת יש איבר בעץ נבחרות לכן סה"כ סיבוכיות זמן O(n)

לכן סה"כ נקבל סיבוכיות זמן O(n+k) במקרה הגרוע כאשר n מספר הנבחרות במערכת וk מספר השחקנים

Add_team(teamId)

- נבדוק תקינות קלט O(1) במקרה הגרוע
- נבצע בדיקה כי לא קיימת נבחרת בטבלת הנבחרות ע"פ ההרצאה חיפוש נבחרת הוא בO(1) משוערך
- במידה ולא קיים נייצר נבחרת חדשה ואובייקט חדש שלה לעץ הטורניר סה"כ סיבוכיות זמן O(1) במקרה הגרוע
- נכניס את הנבחרת לטבלת הערבול עבור הכנסה O(1) משוערך כפי שהסברנו על אלגוריתם הכנסה למעלה
- נעדכן את מספר מספר הנבחרות הכולל O(1)
- הסיבה להיותה של הסיבוכיות המשוערכת הינה שבמקרה ואנחנו מכניסים איבר חדש לטבלת הערבול (שהם הקבוצות) נוכל להגיע למצב שכמות האיברים שווה לגודל הטבלה , ובכדי לשמור על פקטור העומס כדי שהסיבוכיות תהיה לפי הפיזור האחיד נצטרך להגדיל את הטבלה פי 2 , כלומר נצטרך לבצע rehashing. וכפי שלמדנו בהרצאה זה מבוצע בסיבוכיות O(1) משעורך.

לכן נקבל בסה"כ סיבוכיות זמן משוערכת של O(1) זמן בממוצע על הקלט כפי שנדרש

Remove_team(teamId)

- נבדוק תקינות קלט O(1) במקרה הגרוע
- נבצע בדיקה כי הנבחרת קיימת בטבלת הערבול O(1) משוערך
- נסיר את האיבר המתאים בעץ הטורניר של הנבחרת כפי שלמדנו הסרת איבר מעץ avl הוא O(log(n)) במקרה הגרוע כאשר n מספר הנבחרות
- נסיר את כל השחקנים מעץ הכוחות ואת האיברים מהרשימה המקושרת נצטרך לעבור עלk איברים במקרה הגרוע לכן סה"כ סיבוכיות זמן O(k) במקרה הגרוע
- נסיר את הנבחרת מטבלת הערבול כאשר הסרת איבר לפי אלגוריתם שלמדנו בשיעורים היא O(1) במקרה המשוערך
- נבצע התאמה וrehashing בטבלת הערבול המידת הצורך כאשר הrehashing הוא בO(1) במקרה המשוערך

נשים לב כי במקרה הלא משוערך ה rehashing יקח nlog(n) זמן במקרה הגרוע כי אנו מחזיקים עצים לכן במקרה הגרוע הסיבוכיות הזמן הכללית היא O(nlog(n)+k) אך במקרה המשוערך נקבל O(log(n)+k+1) כלומר O(log(n)+k) סיבוכיות זמן משוערכת

- הסיבה להיותה של הסיבוכיות המשוערכת הינה שבמקרה ואנחנו מסירים איברים מטבלת הערבול (שהם הקבוצות) נוכל להגיע למצב שכמות האיברים בקבוצה הינה רבע מגודל הטבלה ולכן נצטרך לבצע rehashing. וכפי שלמדנו בהרצאה זה מבוצע בסיבוכיות O(1) משעורך.

לפני שנמשיך אסביר על פונקציה שנשתמש בה והיא findAndSum()

- נבצע סיור שבו נחפש איבר בעץ הטורניר כאשר יש n נבחרות בעץ לכן בעץ avl החיפוש יקח O(log(n)) סיבוכיות זמן במקרה הגרוע
- בזמן הסיור נסכום את כל האקסטרה עד הגעה לאיבר כך נדע את סכום האקסטרות שיש להוסיף לאיבר

לכן סיבוכיות הזמן של הפונקציה O(log(n) במקרה הגרוע.

עוד פונקציה שנשתמש בה היא select() לפונקציה נביא את מספר האינדקס של האיבר שנרצה למצוא והיא תעזור לנו במציאת הכוח החציוני

- הפונקציה תבצע סיור כך שתבדוק את אם מספר האיבר בתת העץ השמאלי קטן מסכום האיברים בתת העץ השמאלי של האיבר הנוכחי אם כן נפנה שמאלה
- אם הוא שווה למספר האיברים בתת השמאלי פחות אחד אז הוא זה האיבר
- אחרת נפנה ימינה
- נשים לב כי נבצע חיפוש במקרה הגרוע בגובה העץ כלומר סיבוכיות זמן O(log(n) במקרה הגרוע

Add_player(int teamId,int playerStrength)

- נבדוק תקינות קלט O(1) במקרה הגרוע
- נבדוק כי הנבחרת קיימת בטבלה O(1) משוערך ו ((O(log(n) במקרה הגרוע כי ניגש לאיבר בטבלה שבה נמצא בזמן קבוע ואז נחפש אותו בעץ שנמצא באיבר
- נאתחל שחקן חדש בעזרת הcounter שיש לנו בנבחרת אליה הגיע והכוח שלו ונקדם את הcounterב 1 סה"כ O(1) במקרה הגרוע
- נוסיף את השחקן לרשימה המקושרת כך שתמיד נוסיף אותו לזנב כדי שנוכל להוציא אותו במידת הצורך הוספת איבר לזנב כאשר הזנב ידוע לנו תיקח O(1) במקרה הגרוע
- נוסיף את השחקן לרשימת הכוחות של השחקנים של הנבחרת ע"פ אלגוריתם מההרצאה הכנסה לעץ תיקח ((O(log(k כאשר k מספר השחקנים במקרה הגרוע
- נקרא לfindAndSum=sumExtra ונעדכן את הסכום כך שnumOfWins=numOfwins+sumExtra סה"כ סיבוכיות זמן ((O(log(n) במקרה הגרוע
- נסיר את האיבר המתאים לנבחרת בעץ הטורניר ((O(log(n) במקרה הגרוע
- נמצא את הכוח החציוני החדש של הנבחרת ע"י select() סה"כ O(log(n) במקרה הגרוע
- נעדכן את הכוח והדירוג החדש של הנבחרת O(1) במקרה הגרוע
- נכניס איבר חדש של הנבחרת לעץ הטורניר עם הערכים החדש ((O(log(n) במקרה הגרוע
- נעדכן את m_HighestRank=root.maxRankAfterExtra כפי שהסברנו השורש יכיל את הערך וניתן לגשת לשורש בזמן קבוע לכן סה"כ O(1) במקרה הגרוע

קיבלנו סה"כ ((O(log(n)+log(k במקרה הגרוע

Remove_newest_player(int teamId)

- נבדוק תקינות קלט O(1) במקרה הגרוע
- נבדוק כי נבחרת קיימת בטבלה O(log(n) במקרה הגרוע
- נסיר את השחקן מהרשימה המקושרת הוא נמצא בזנב הרשימה ויש לנו אותה לכן סה"כ O(1) במקרה הגרוע
- נעדכן אתteamCounter פחות אחד כי נסיר את השחקן שהיה עם הid הזה O(1) במקרה הגרוע
- נסיר את השחקו מעץ הכוחות סה"כ סיבוכיות זמן ((O(log(k במקרה הגרוע
- נקרא לfindAndSum=sumExtra ונעדכן את הסכום כך שnumOfWins=numOfwins+sumExtra סה"כ סיבוכיות זמן ((O(log(n) במקרה הגרוע
- נמצא את הכוח החציוני החדש של הנבחרת ע"י select() סה"כ O(log(n) במקרה הגרוע
- נסיר את האיבר המתאים לנבחרת מעץ הטורניר O(log(n) במקרה הגרוע
- נעדכן את הכוח והדירוג החדש של הנבחרת O(1) במקרה הגרוע
- נכניס איבר חדש של הנבחרת לעץ הטורניר עם הערכים החדש ((O(log(n) במקרה הגרוע
- נעדכן את m_HighestRank=root.maxRankAfterExtra כפי שהסברנו השורש יכיל את הערך וניתן לגשת לשורש בזמן קבוע לכן סה"כ O(1) במקרה הגרוע

נקבל בסה"כ סיבוכיות זמן במקרה הגרוע של ((O(log(n)+log(k

play_match(teamId1,teamId2)

- נבדוק תקינות קלט O(1) במקרה הגרוע
- נבדוק כי הקבוצות קיימות O(log(n) במקרה הגרוע

- נבדוק ביחס לכוח של נבחרת שנמצא בשדה מי יותר גדול וכך נדע מי ניצח אם הכוח שווה אז הנבחרת עם id קטן יותר ניצחה סה"כ O(1) במקרה הגרוע לבדיקת השדות
- נעדכן את מספר הניצחונות של הנבחרת המנצחת ב1 O(1) במקרה הגרוע
- נקרא לfindAndSum=sumExtra ונעדכן את הסכום כך שnumOfWins=numOfwins+sumExtra סה"כ סיבוכיות זמן O(log(n)) במקרה הגרוע
- נמצא את הכוח החציוני החדש של הנבחרת ע"י select() סה"כ O(log(n) במקרה הגרוע
- נסיר את האיבר המתאים לנבחרת מעץ הטורניר O(log(n) במקרה הגרוע
- נעדכן את הכוח והדירוג החדש של הנבחרת O(1) במקרה הגרוע
- נכניס איבר חדש של הנבחרת לעץ הטורניר עם הערכים החדש O(log(n)) במקרה הגרוע
- נעדכן את m_HighestRank=root.maxRankAfterExtra כפי שהסברנו השורש יכיל את הערך וניתן לגשת לשורש בזמן קבוע לכן סה"כ O(1) במקרה הגרוע
- נחזיר את הid של הקבוצה המנצחת O(1) במקרה הגרוע

סה"כ קיבלנו O(log(n)) במקרה הגרוע

Num_wins_for_team(int teamId)

- נבדוק תקינות קלט O(1) במקרה הגרוע
- נבדוק כי הנבחרת קיימת O(log(n)) במקרה הגרוע
- נקרא לfindAndSum=sumExtra סה"כ סיבוכיות זמן O(log(n)) במקרה הגרוע
- נחזיר את סכום הניצחונות לקיימים לאיבר בנבחרת +sumExtra סה"כ O(1) במקרה הגרוע

סה"כ קיבלנו O(log(n)) במקרה הגרוע

Get_highest_ranked_team()

- נבדוק כי קיימים נבחרות ע"י שדה m_numOfTeams אם לא קיימים נחזיר -1 סה"כ O(1) במקרה הגרוע
- נבדוק אם השורש של עץ הטורניר לא מאותחל לאיבר במצב שאין נבחרות בטורניר לכן אין נבחרת עם שחקן אבל יש נבחרות בכללי אז נחזיר 0 סה"כ O(1) במקרה הגרוע
- אחרת נחזיר m_highestRankTeam סה"כ O(1) במקרה הגרוע

סה"כ קיבלנו O(1) במקרה הגרוע

Unite_teams(teamId1,teamId2)

- נבדוק תקינות קלט O(1) במקרה הגרוע
- נבדוק כי שני הנבחרות קיימות O(log(n)) במקרה הגרוע
- במידה ושני הנבחרות ריקות נסיר את הנבחרת השנייה מטבלת הערבול O(nlog(n) במקרה הגרוע וO(1) במקרה המשוערך
- נעדכן את כל השחקנים בid החדש שלהם לאחר מעבר לנבחרת החדשה ע"י מעבר כל האיברים ברשימה סה"כ O(k2) במקרה הגרוע כאשר k2 מספר השחקנים בנבחרת השנייה ששחקניה עוברים לנבחרת הראשונה
- נקרא לfindAndSum=sumExtra עבור הנבחרת הראשונה ונעדכן את הסכום כך שnumOfWins=numOfwins+sumExtra סה"כ סיבוכיות זמן O(log(n)) במקרה הגרוע
- נסיר את שתי הנבחרות מעצי הטורניר O(log(n)) במקרה הגרוע
- כעת נקרא לפונקציה makeUnite אשר תפקידה לאחד את עצי הכוחות המורכב ממספר פעולות נסביר אותן
- נאתחל שני מערכים כל אחד בגודל הנבחרת שהוא מייצג O(1) במקרה הגרוע
- נקרא לפונקציה putStrengthInArray אשר מכניסה למערך את איברי העץ כוחות של כוחות נבחרת ע"פ הסדר שלהם בעץ ע"י ביצוע סיור inOrder על האיברים בעץ לכן נקבל סיבוכיות זמן O(k1/k2) כאשר k1/k2 מספר השחקנים בכל נבחרת

- כעת עבור הנבחרת השנייה נקרא updateStrengthId אשר תעדכן גם אתid שמחזיקים עצי הכוחות לid המעודכנים כדי שלא תיווצר חפיפה בין id נעשה זאת ע"י מעבר על כל האיברים במערך של הנבחרת השנייה סה"כ O(k2) במקרה הגרוע
- נאתחל מערך בגדול המשותף של שני הנבחרות O(1) במקרה הגרוע
- כעת נקרא לפונקציה mergeStrength אשר תקבל שני מערכים כל אחד מכיל את הכוחות של הנבחרת ממויינים וימזג ביניהם יש בסה"כ k1+k2 כוחות וע"פ איך שיצרנו את המערכים האלה דאגנו לכך שכל אחד מהם יהיה ממוזג לכן כדי למזג אותם נצטרך לעבור על כל איבר רק פעם אחת ולכן O(k1+k2)
- makeUnitedId- בעזרת כך שהמערך שלנו ממוין נשתמש בזה בפונקציה זאת כדי לבנות עץ כמעט מלא ע"י בנייה רקורסיבית כך שבהתחלה נבחר את השורש להיות האיבר האמצעי במערך המתמודדים מימינו הם תת העץ הימני ומשמאלו הם תת העץ השמאלי וכך נמשיך ברקורסיה לכל המערך ונקבל עץ כמעט מלא מאוזן בסיבוכיות זמן O(log(k1+k2))
- makePostOrderUpdateValue- כעת כשיש לנו עץ מאוזן נדאג לעדכן את השדות שחשובים לנו לשמירת האיזון של העץ בהמשך ונעשה זאת ע"י סיור postOrder ע"פ הרצאה סיור זה לוקח O(k1+k2)

- כעת נדאג לרשימות המקושרת של השחקנים מכיוון שנרצה לשמור על הסדר כך שהשחקנים שמצטרפים מהנבחרת השנייה הצטרפו מאוחר יותר לפי סדר כניסתם לנבחרת נחבר בין זנב הרשימה המקושרת של הרשימה הראשונה לראש הרשימה המקושרת של הנבחרת השנייה נעשה זאת בO(1) במקרה הגרוע כי יש לנו כבר את השדות לביצוע המיזוג
- נעדכן את מספר השחקנים בנבחרת המאוחדת O(1) במקרה הגרוע
- כעת ניתן להסיר את הנבחרת השנייה מטבלת הנבחרות במקרה הגרוע הפעולה היא ב O(nlog(n)) סיבוכיות זמן אבל במקרה המשוערך נקבל O(1)
- נעדכן את הכוח והדירוג החדש של הנבחרת O(1) במקרה הגרוע
- נכניס איבר חדש של הנבחרת לעץ הטורניר עם הערכים החדש O(log(n)) במקרה הגרוע
- נעדכן את m_HighestRank=root.maxRankAfterExtra כפי שהסברנו השורש יכיל את הערך וניתן לגשת לשורש בזמן קבוע לכן סה"כ O(1) במקרה הגרוע
- נקבל כי מבחינת סיבוכיות זמן קיבלנו סיבוכיות זמן משוערכת של O(log(n)+k1+k2)
- הסיבה להיותה של הסיבוכיות המשוערכת הינה שבמקרה ואנחנו מסירים איברים מטבלת הערבול (שהם הקבוצות) נוכל להגיע למצב שכמות האיברים בקבוצה הינה רבע מגודל הטבלה ולכן נצטרך לבצע rehashing. וכפי שלמדנו בהרצאה זה מבוצע בסיבוכיות O(1) משעורך.
- נשים לב כי מבחינת סיבוכיות מקום אנחנו נחזיק בזמן פעולת הפונקציה מערכים בגודל k1 k2 וk1+k2 ונייצר עץ חדש בגודל k1+k2 לכן נשמרת בזמן הפעולה סיבוכיות מקום O(n+k) בסוף פעולת הפונקציה נשחרר את העצים והמערכים שיותר לא צריך להשתמש בהם

לפני שנסביר על אופן פעולת פונקציית הטורניר נסביר על פונקציות שנשתמש בהן לטובת הפונקצייה

findClosestRight(int RightBound,Node<int,TeamInTournamet>* winner)

- הפונקציה תחזיר לנו את האינדקס של האיבר שנמצא במקום הכי ימני משמאל לטווח הימני כלומר ע"פ הסידור שביצענו לעץ הטורניר לפי כוחות זה יהיה האיבר המנצח כי הוא יהיה האיבר הכי ימני בטווח הטורניר
- נבצע זאת ע"י סיור נבדוק עם הכוחות באיבר כרגע גדול מהאיבר מהטווח הימני אז נפנה שמאלה
- אחרת נמשיך את הסיור ע"י בדיקה של תתי העצים הימני והשמאלי כך שאם נפנה שמאלה נוסיף גם את כמות האיברים משמאל לאיבר כולל אותו
- במידה והגענו לאיבר המבוקש נשמור מצביע אליו כדי שנוכל להחזיר את המזהה שלו בסוף הפונקציה טורניר
- בסה"כ נבצע סיור לאורך הגובה של העץ כלומר סה"כ O(log(n)) במקרה הגרוע

addValue(Node<int,TeamInTournamet>* node,int extra)

- פונקציה שבה נעדכן את האקסטרה עבור איבר במסלול החיפוש
- נעשה זאת בצורה זהה לאלגוריתם המתחסנים שהוסבר בתרגול
- עבור פנייה ימינה שלא לאחר רצף פניות ימינה נוסיף את האקסטרה לשדה האקסטרה
- אם ביצעת שמאלה אחרי פנייה ימינה נוסיף -אקסטרה
- אם הגעת לאיבר שלא אחרי פנייה ימינה נוסיף אקסטרה

- אם יש לנו תת עץ ימני נוסיף לתת העץ -אקסטרה
- נוסיף ל sumTill בהתאם לערך האקסטרה שנוסיף לאיבר נוסיף אותו להורה במידה וקיים
- בסה"כ נבצע מסלול חיפוש כך שבמקרה הגרוע סיבוכיות הזמן O(log(n))

rightMax(Node<int,TeamInTournament>* node)

- מטרת הפונקציה לעדכן את maxRankAfterExtra כאשר נחפש את המנצח המקסימלי בסיבוב הנוכחי בטורניר
- נבצע סיור עד המנצח המקסימלי בסיבוב הנוכחי תוך שנסכום את האקסטרה עד אליו sumExtra=findAndSum() סה"כ O(log(n)) במקרה הגרוע
- נחזיק משתנה sumRight-יחזיק את הדירוג החדש של תת העץ הימני של האיבר
- newNodeRank-הדירוג החדש של האיבר
- maxTemp- יחזיק את המקסימום בדירוג עד אותו רגע
- כעת נשים לב כי עבור האיברים שנמצאים מימין למסלול החיפוש הם לא נמצאים בסיבוב הנוכחי של הטורניר לכן לא יקבלו אקסטרה לדירוג
- עבור כל איבר נעדכן:
- newNodeRank=myRank+sumExtra
- sumLeft=node->Left.maxRankBeforeExtra+sumExtra+node->Left.extra
- נעדכן maxRankAfter=max(sumLeft,newNodeRank,node->Right.maxRankAfter)
- נעדכן maxTemp
- נעדכן sumExtra-=node.extra
- במידה ויש הורה נמשיך להורה
- בסוף במידה וmaxTemp>m_highestRank נעדכן אותו להיות m_highestRank=maxTemp
- נבצע את המסלול ההפוך מהאיבר לשורש כך שנבצע שני מסלולי חיפוש לכן נקבל O(log(n)) סיבוכיות זמן במקרה הגרוע
- נשים לב כי עבור הוספת הניצחונות הניצחונות ישפיעו על תתי העץ הימנים נרצה למצוא את מי שהטורניר השפיע עליו הכי הרבה ע"י כך שנעשה חיפוש למנצח של הטורניר נתפוס את הדירוג המקסימלים של תתי העץ השמאליים שלו לפני הוספת ניצחונות כי עידכנו את השדות בהתאם ואת האיברים המסלול החיפוש נעדכן ע"פ איך שהוסבר מעלה לכן נוכל להגיע להשפעה שתקרה עבור כל משתתף בטורניר וע"י עדכון השדות אנחנו נדע עבור כל תת עץ מה הוא המדורג הכי גבוהה שלו לפני הוספת הניצחונות ובהינתן שאנחנו מעדכנים את הניצחנות כפי שהסברנו נקבל כי עבור כל תת עץ נדע מה הדירוג המקסימלי לאחר ביצוע הטורניר.

במהלך ביצוע הפעולה נבצע חיפוש עבור האיבר המקסימלי וממנו לשורש בסך הכל שני חיפושים לכן סה"כ O(log(n)) סיבוכיות זמן במקרה הגרוע

Stimulate_match(int I ,int HighBound,Node<int,TournamentInTeam>* winner)

- מטרת הפונקציה לבצע את הטורניר
- עבור כל סיבוב נמצא את המפסיד המקסימלי כאשר שנמצא באינדקס HighBound-i כאשר HighBound הוא אינדקס מנצח הטורניר וi מספר המנצחים בשלב זה בעזרת פונקציית select O(log(n)) במקרה הגרוע
- נבצע addValue(winner,1) כאשר winner הוא מנצח הטורניר O(log(n)) במקרה הגרוע
- נבצע addValue(leftBoundNode,-1) כאשר leftBoundNode הוא המפסיד המקסימלי שמצאנו O(log(n)) במקרה הגרוע
- נבצע rightMax O(log(n)) במקרה הגרוע
- נבצע את הפונקציה כל עוד i>0 וכל סיבוב נחלק את i/2 לכן סה"כ log(i) סיבובים

לכן בסה"כ סיבוכיות זמן(log(i)*log(n))Oבמקרה הגרוע

כעת נסביר על פונקציית הטורניר

playTournament(int lowPower,int highPower)

- נבצע בדיקות קלט בO(1)

- נקרא לפונקציה findClosestRight ונעדכן את winner ((O(log(n במקרה הגרוע
- נקרא לפונקציה findClosestRight עבור lowPower-1 כך שנקבל את האינדקס של האיבר הכי ימני משמאל לאיבר הראשון בטורניר סה"כ ((O(log(n במקרה הגרוע
- נבדוק אם ההפרש בין האינדקסים הוא לוג של 2 נעשה זאת בעזרת פונקציית isPower
- הפונקציה בודקת אם תוצאת מודולו 2 אינה שווה 0 בשלב מסוים נחזיר false אחרת נמשיך לחלק כל עוד המספר קטן מ1 סה"כ עבור i נקבל log(i) איטרציות לכן סיבוכיות זמן ((O(log(i כאשר i מספר נבחרות בטווח
- נקרא לפונקציה stimulateMatch סיבוכיות זמן במקרה הגרוע ((O(log(i)*log(n
- נחזיר מזהה של winner O(1)

לכן סה"כ סיבוכיות זמן ((O(log(i)*log(n במקרה הגרוע

<u>סיבוכיות מקום:</u>

אנחנו מחזיקים מבני נתונים הבאים:

1. רשימת שחקנים אשר בסה"כ יש k שחקנים לכן O(k) סיבוכיות מקום
2. עץ כוחות שחקנים יש k שחקנים לכן O(k) סיבוכיות מקום
3. טבלת ערבול אשר מכילה בסה"כ n נבחרות בתוכה לכן O(n) סיבוכיות מקום
4. עץ טורניר אשר מכיל n נבחרות לכל היותר לכן O(n) סיבוכיות מקום

לכן נקבל בסה"כ סיבוכיות מקום O(n+k)