יבש לרטוב 2

מגישים:
לירן רדל 314989427
אנה גריגוריבקר 321931396

תיאור מבנה הנתונים:

עץ דרגות AVL - RankTree<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 שדה של גודל העץ, כלומר כמה
איברים יש בו בשם size ,ומצביע לשורש העץ שהוא מסוגRankNode<Key,Value>
בנוסף כל Node כזה מחזיק שדה Weight אשר בעזרתו נוכל לחשב את הדרגה של כל איבר כפי שנלמד בהרצאות, וכך גם הגלגולים ואיזון העץ לאחר כל הכנסה/הוצאה נעשה כפי שנלמד בקורס.

טבלת ערבול HashTable<Value>t :

טבלת ערבול בעלת פונקצית ערבול $h(x) = x \bmod m$ כאשר m הוא גודל הטבלה, נשים לב כי פונקציה זו מקיימת את הנחת הפיזור האחיד הפשוט.
בנוסף נשתמש ב-Chain Hashing במקרה של התנגשויות ובמקרה הצורך נגדיל את הטבלה פי 2 - כאשר הגודל ההתחלתי יהיה 11 (ע"מ למנוע כמה שניתן מצב בו m הוא חזקה שלמה של 10 או 2) (אין צורך בהקטנה כי לא דרושה תמיכה בהוצאה מהטבלה בתרגיל זה) , ההגדלה תיעשה בכל פעם שפקטור העומס יהיה גדול מ-2 , בכך נשמור על מצב בו $n = O(m)$ ולכן פעולת $find(x)$ תיעשה ב- $O(1)$ בממוצע על הקלט ופעולת $insert(x)$ תיעשה ב- $O(1)$ משוערך בממוצע על הקלט כפי שלמדנו בתרגול.

Union Find :

מבנה Union Find הממומש בעזרת עצים הפוכים בו כל איבר מיצג שחקן וכל קבוצה מיצגת קבוצת כדורגל.
המבנה בו נחזיק את כל האיברים שלנו יהיה טבלת ערבול של כלל השחקנים, והמבנה בו נחזיק את הקבוצות יהיה עץ AVL ובו הקבוצות שנמצאות כרגע במערכת. בנוסף, נעשה במימוש איחוד לפי גודל וכיווץ מסלולים לכן הפעולות $union(p,q)$ ו- $find(x)$ כפי שהוגדרו בהרצאה יעשו ב - $O(\log^* n)$ משוערך כפי שלמדנו
כאשר כל פעם אשר תיקרא פונקציית find יבוצע קיצור מסלולים על מסלול החיפוש אל שורש העץ ההפוך כפי שנלמד, וכל איבר במבני נתונים זה יחזיק 2 שדות נוספים שיעזרו לנו במימוש הפונקציות - rS ו-temp_plays.

מבנים פנימיים של התוכנית :

Player: מחלקה המייצגת שחקן כדורגל במערכת, בעל שדות פנימיים שישמשו 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 אליה, שדה שישמש אותנו בחישוב של partial_spirit (שומר מה הspirit הכולל של הקבוצה ברגע הצטרפות השחקן אלייה) ומצביע ל-איבר שלו בUF.

Team: מחלקה המייצגת קבוצת כדורגל במערכת, בעלת שדות פנימיים שיממשו אותנו במימוש הפונקציות: מספר המזהה של הקבוצה, מספר הנקודות של הקבוצה, שדה team_ability שהוא סכום כל ה-ability של כל השחקנים בקבוצה, שדה team_spirit כפי שהוגדר בתרגיל, משתנה בוליאני שמסמל האם הקבוצה פעילה או לא ושדה של מצביע לNode ב-UF ששייך לשחקן הראשון שהצטרף לקבוצה.

Ability: מחלקה שמטרה למיין קבוצות לפי ה- ability של כל קבוצה, אם ערכים אלו שווים, נמיין לפי מזהה הקבוצה בסדר עולה. נשתמש בה ע"מ לממש עץ דרגות בו הקבוצות ממוינות באופן שנדרש בפונקציה get_ith_pointless_ability.

world_cup_t: המחלקה הראשית בתרגיל, מכילה את השדות:

- עץ דרגות AVL המכיל את הקבוצות שנמצאות כרגע בטורניר ממוינות לפי מזהה.
- עץ דרגות AVL המכיל את הקבוצות שנמצאות כרגע בטורניר ממוינות לפי pointless ability.
- עץ דרגות AVL המכיל את הקבוצות שהודחו מהטורניר ממוינות לפי סדר הכנסה (ע"מ למנוע התנגשויות במקרה של מיון לפי מזהה)
- טבלת ערבול המכילה את כלל השחקנים במערכת.
- מבנה Union Find בו כל איבר מיצג שחקן, נשים לב שכמות האיברים חסומה ע"י כמות השחקנים הכוללת.

סיבוכיות המקום של מבני הנתונים:

עבור המחלקות Player, Team ו- Ability כל אחת מהן מחזיקה מספר קבוע של משתנים פרימיטיביים ומצביעים ולכן סיבוכיות המקום של כל אחת מהן תהיה $O(1)$.
עבור המחלקה world_cup_t - נסמן 'k - כמות הקבוצות שנמצאות כרגע בטורניר , k'' - כמות הקבוצות שהודחו מהטורניר, $k = k' + k''$, n - כמות השחקנים הכוללת שהיתה בטורניר. במחלקה יש 2 עצים המכילים את הקבוצות שנמצאות כרגע בטורניר כלומר סיבוכיות מקום של $O(k') \approx O(k)$. עץ המכיל את הקבוצות שהודחו, סיבוכיות מקום $O(k'') \approx O(k)$. טבלת ערבול של כלל השחקנים,בטבלה m תאים אשר מכילים סה"כ n ערכים, בכך שהשתמשנו בטבלה בגודל דינאמי ופקטור העומס שלנו היה חסום ע"י קבוע דאגנו שבכל זמן נתון $m = O(n)$ לכן סה"כ סיבוכיות המקום של הטבלה היא $O(n)$.
Union Find בו כל איבר הוא שחקן , סיבוכיות מקום $O(n)$.
סה"כ סיבוכיות מקום של $O(n+k)$.

תיאור הפונקציות:

world_cup_t () - נאתחל שלושה עצים ריקים, טבלת ערבול ריקה ו-Union find ללא איברים בעצים ההפוכים, כל אחד ב - $O(1)$, סה"כ סיבוכיות של $O(1)$.

~world_cup_t() - נעבור על כל הפרמטרים במבני נתונים, ונמחק אותם מן הזכרון. עבור העצים שסיבוכיות המקום של כל אחד מהם היא $O(k)$ זה יעשה בסיבוכיות של $O(k)$ כפי שלמדנו, כשמחיקת העץ תעשה במעבר InOrder
על האיברים בעץ ומחיקתם לפי כך.
עבור טבלת הערבול נעבור על כל הטבלה ונמחק כל שחקן ב- $O(1)$, נשתמש בכך שכל שחקן מחזיק מצביע לאיבר שלו ב-UF ונמחק את האיבר, גם כן ב- $O(1)$, ישנם סה"כ m תאים בטבלת הערבול המחזיקים בסה"כ n איברים, לכן מחיקה של כלל הטבלה תעשה ב- $O(n+m)$,נשים לב שכיוון שהגדלת הטבלה פי 2 נעשית רק כאשר $\frac{n}{m} = \alpha > 2 \implies n > 2m$ נשמור על מצב בו $m = O(n)$ ולכן מחיקת כלל הטבלה תעשה ב-$O(n)$.
סה"כ סיבוכיות של $O(n+k)$.

add_team - נבדוק אם הקבוצה קיימת כבר בעץ הקבוצות ב- $O(\log k)$ על ידי חיפושה בעץ הקבוצות הממוינות לפי מזהה כפי שנלמד בתרגול, ואם קיימת נחזיר שגיאה. אם לא, ניצור אובייקט חדש של קבוצה ב-$O(1)$ ונכניס אותו לשני העצים הרלוונטיים ב- $O(\log k)$ - עץ הקבוצות שממויין לפי מזהה הקבוצות, והעץ קבוצות הממויין לפי ה-Ability של הקבוצות(שימוש במחלקת Ability כדי למיין), סה"כ $O(\log k)$.

remove_team - מחיקה של הקבוצה מעצי הקבוצות(הממויינות לפי מזהה ולפי Ability) כל אחד ב- $O(\log k)$ כפי שנלמד בהרצאה, והכנסה לעץ הקבוצות שנמחקו ב - $O(\log k)$ כפי שנלמד בהרצאה, מאחר ואנו עדיין רוצים לשמור את המידע של הקבוצות במערכת. סה"כ סיבוכיות של $O(\log k)$.

add_player - ראשית נחפש את השחקן לפי המזהה בטבלת הערבול כפי שנלמד בהרצאה ב-O(1) בממוצע על הקלט על מנת לבדוק שאינו כבר נמצא במערכת, אם כן נחזיר שגיאה. לאחר מכן נחפש את הקבוצה של השחקן בעץ הקבוצות הממוינות לפי id בסיבוכיות של $O(logk)$ כפי שנלמד בהרצאה, ונבדוק שהיא קיימת אחרת נחזיר שגיאה. לאחר מכן ניצור אובייקט של השחקן בטבלת הערבול, כאשר ההכנסה שלו קוראת ב-O(1) בממוצע על הקלט משוערך מאחר והמערך דינאמי ועלול לגדול כפי שנלמד בהרצאה. נאתחל בהתאם לקבוצה את השדות הנוספים של השחקן בביצוע פעולות בודדות ב-O(1). לאחר מכן ניצור איבר ב-UF ובהתאם האם הקבוצה ריקה או לא נעדכן את שדה ראש הקבוצה שלה. לאחר מכן נשנה את השדה org_team_spirit_join שיעזור לנו בהמשך להיות הפרמוטציה הכללית של הקבוצה של השחקן כפול היפוך השדה הנוסף של ראש הקבוצה בעץ ההפוך של הקבוצה של השחקן על מנת למנוע בעיות בחישובים בהמשך האלגוריתם של partial_spirit. לאחר מכן נסיר את הקבוצה מעץ הקבוצות לפי Ability, נעדכן את ה-ability הכולל שלה מאחר ולאחר הוספת השחקן הוא השתנה ואז נכניסה מחדש בסיבוכיות של $O(logk)$ כפי שנלמד

בהרצאה. בסה"כ הסיבוכיות הינה $O(logk)$ משוערך בממוצע על הקלט כנדרש.

play_match - מציאת שתי הקבוצות בעץ הקבוצות ב - $O(\log k)$ וביצוע מספר פעולות אשר חסום ע"י קבוע,
ביניהן עדכון של שדה ה-points ושדה המשחקים ששוחקו ע"י הקבוצה בהתאם לנדרש. סה"כ $O(\log k)$.

num_played_games_for_player - מציאה של השחקן בטבלת הערבול - $O(1)$ בממוצע על הקלט, וגישה לאיבר השחקן ב-UF וקריאה לפונקצית עזר העובדת באופן דומה לפונקציית ה – $find(x)$ ב-union find רק שכעת נחזיק משתנה אותו נרצה להחזיר בסוף פעולת הפונקציה, בכל מעבר באיבר לאורך מסלול החיפוש ב-$find(x)$ נבצע חיבור של המשתנה עם שדה נוסף שנשמור בכל node, לו קראנו temp_plays, נבצע פעולה זו בכל node לאורך מסלול החיפוש בדומה לשיטת הארגזים אשר נלמדה בתרגול 9. לאחר מכן נחזיר את סכום השדות הנוספים שסכמנו לאורך מסלול החיפוש ועוד כל המשחקים שהקבוצה של השחקן שיחקה ועוד מספר המשחקים שהשחקן שיחק כשהצטרף למערכת פחות מספר המשחקים שהשחקן שיחק כאשר הצטרף אל הקבוצה הראשונית בה שיחק, מה שבהתחשב לתחזוק שביצענו בשאר הפעולות אמור להחזיק את מספר המשחקים שהשחקן שיחק המעודכן כנדרש. החיפוש לוקח $O(log*n)$ כפי שנלמד בהרצאה. את השדה הנוסף נדאג לעדכן בכל פעולת $union(p,q)$ ו-$find(x)$ כך שפעולה כזאת תחזיר את התשובה הנכונה. בנוסף, ע"מ לעמוד בדרישות הסיבוכיות נבצע כיווץ מסלולים בפונקציה זו,כיווץ המסלולים יעשה כפי שלמדנו בהרצאה, עדכון השדה הנוסף יעשה באופן הבא: ראשית נחפש את שורש העץ ההפוך, וכאשר אנחנו עוברים על כל איבר (חוץ מהשורש) נסכום את השדה הנוסף של כל איבר לתוך משתנה זמני שייסכם לאורך מסלול החיפוש שנקרא לו לצורך ההמחשה sum. לאחר מציאת השורש, נבצע שוב את החיפוש החל מהאיבר הראשון במסלול החיפוש כאשר כל איבר יחובר אל השורש, ונחליף את השדה הנוסף שלו temp_plays ב- sum, ולאחר מכן נפחית את השדה הנוסף של האיבר מן sum על מנת שבאיבר הבא זה לא ייחשב. כך ניצור מצב שבו כל איבר שחובר על השורש ישירות, השדה הנוסף שלו מכיל את סכום כל השדות הנוספים של האיברים הבאים אחריו. מדובר על חיפוש פעמיים מן האיבר אל השורש לכן הסיבוכיות נותרה $O(log*n)$.
נשים לב כי כיוון ופונקצית העזר הזו ממומשת באותו אופן כמו פונקציית $find(x)$ הכוללת כיווץ מסלולים, שערוך שלה עם פונקציות הפועלות כמו $union(p,q)$ עם איחוד לפי גודל יביא לסביכויות משוערכת של $O(\log^* n)$, סה"כ נקבל סיבוכיות של $O(\log^* n)$ משוערך בממוצע על הקלט.

add_player_cards - מציאה של השחקן בטבלת הערבול - $O(1)$ בממוצע על הקלט כפי שנלמד בהרצאה, ומציאה של הקבוצה שלו בעזרת הפונקציה $find(x)$ במבנה ה-UF ב-$O(\log^* n)$ משוערך ע"מ לבדוק אם היא עדיין פעילה בטורניר (בראש העץ ההפוך נמצא תמיד המצביע המעודכן לקבוצת השחקנים בכל עץ הפוך המייצג קבוצה במבני הנתונים UF).
לאחר מכן במידת הצורך נעדכן את שדה הכרטיסים אצל השחקן דרך טבלת הערבול. סה"כ סיבוכיות של $O(\log^* n)$ משוערך בממוצע על הקלט.

get_player_cards - מציאה של השחקן בטבלת הערבול - $O(1)$ בממוצע על הקלט כפי שנלמד בהרצאה והחזרה של השדה הרצוי.

סה"כ $O(1)$ בממוצע על הקלט.

get_team_points - מציאת הקבוצה בעץ הקבוצות לפי מזהה ב - $O(\log k)$ על ידי חיפושה כפי שנלמד בהרצאה והחזרה של השדה הרצוי, נשים לב כי דאגנו שהוא יעודכן בהתאם בכל משחק של הקבוצה.
סה"כ $O(\log k)$.

get_ith_pointless_ability - מציאת האיבר בעל האינדקס ה-i בעץ הדרגות המכיל את הקבוצות ממוינות לפי Ability (pointless ability) כפי שהוגדר בתיאור הפונקציה, המציאה תעשה ע"י פונקציית $select(i)$ הממומשת כפי שלמדנו בתרגול בסיבוכיות של $O(\log k)$, סה"כ סיבוכיות של $O(\log k)$.

get_partial_spirit - נמצא את השחקן בטבלת הערבול ב- $O(1)$ בממוצע על הקלט כפי שנלמד בהרצאה לפי מזהה השחקן. לאחר מכן נפעיל פונקצית עזר שעובדת באופן דומה לפונקציית ה – $find(x)$ ב-union find רק שכעת נחזיק משתנה אותו נרצה להחזיר בסוף פעולת הפונקציה, בכל מעבר באיבר לאורך מסלול החיפוש ב-$find(x)$ נבצע הרכבה של המשתנה אשר יאותחל לשדה ששמרנו אצל השחקן
org_team_spirit_join שאומר על ה-spirit הכולל של הקבוצה ברגע הצטרפות השחקן ושל שדה נוסף אותו נשמור בכל Node ב Union Find, הנקרא rS' בכל הרכבה כזו המשתנה יהיה מימין ע"מ שסדר ההכפלה יתאים להגדרת ה-partial_spirit, ואת ההכפלה נבצע עד שנגיע לשורש העץ ההפוך(כולל השדה הנוסף של שורש העץ) ונחזיר את התוצאה שאמורה לתת לנו את הpartial_spirit המעודכן של השחקן כנדרש. את השדה הנוסף נדאג לעדכן בכל פעולת $union(p,q)$ ו-$find(x)$ כך שהרכבה כזאת תחזיר את התשובה הנכונה. ע"מ לעמוד בדרישות הסיבוכיות נבצע כיווץ מסלולים בפונקציה זו כאשר נבצע את פעולת החיפוש של שורש העץ ההפוך. כיווץ המסלולים יעשה כפי שלמדנו בהרצאה, עדכון השדה הנוסף יעשה באופן הדומה לשיטת הארגזים הנלמד בתרגול 9: נחזיק משתנה לו נקרא sum_spirit שיאותחל לפרמוטציה הטבעית.
ראשית נחפש את שורש העץ ההפוך, וכאשר אנחנו עוברים על כל איבר (חוץ מהשורש) נכפיל את השדה הנוסף rS מימין של כל איבר לתוך משתנה זמני שיוכפל לאורך מסלול החיפוש שנקרא לו לצורך ההמחשה total_rS. לאחר מציאת השורש, נבצע שוב את החיפוש החל מהאיבר הראשון במסלול החיפוש כאשר כל איבר יחובר אל השורש, ונחליף את השדה הנוסף שלו rS ב- total_rS , ולאחר מכן נכפול משמאל בהופכי של השדה הנוסף של האיבר מן total_rS על מנת שבאיבר הבא זה לא ייחשב. כך ניצור מצב שבו כל איבר שחובר על השורש ישירות, השדה הנוסף שלו מכיל את סכום כל השדות הנוספים של האיברים הבאים אחריו. מדובר על חיפוש פעמיים מן האיבר אל השורש לכן הסיבוכיות נותרה $O(log*n)$.
נשים לב כי כיוון ופונקצית העזר הזו ממומשת באותו אופן כמו פונקציית $find(x)$ הכוללת כיווץ מסלולים, שערוך שלה עם פונקציות הפועלות כמו $union(p,q)$ עם איחוד לפי גודל יביא לסביכויות משוערכת של $O\left(\log^* n\right)$, סה"כ נקבל סיבוכיות של $O\left(\log^* n\right)$ משוערך בממוצע על הקלט.

buy_team - ראשית נמצא את הקבוצות בעץ הקבוצות ב- $O(\log k)$ על ידי חיפושן בעץ הקבוצות הממוינות לפי מזהה כפי שנלמד בהרצאה, לאחר מכן נקרא לפונקציית $union(p,q)$ כאשר אנו מעבירים להן את הקבוצות הפונות ישירות לראשי העצים ההפוכים שלהן, ומבצעת איחוד לפי גודל כפי שלמדנו ומעדכנת את השדות הנוספים בהתאם באופן הבא:
עבור ה-spirit:

- אם הקבוצה הרוכשת היא הגדולה:

נכפול בשורש הקבוצה הנרכשת את השדה הנוסף rs משמאל כך:

$$BuyingRoot(rs).\ inv()*Buying(TeamSpirit)*BoughtRoot(rs)$$

- אם הקבוצה הנרכשת היא הגדולה:

נכפול בשורש הקבוצה הנרכשת את השדה הנוסף rs משמאל כך:

$$Buying(TeamSpirit)*BoughtRoot(rs)$$

נכפול בשורש הקבוצה הרוכשת את השדה הנוסף rs כך:

$$BoughtRoot(spiritPath)_{new}.\ inv()*Buying(TeamSpirit)$$

עבור המשחקים ששוחקו:

- אם הקבוצה הרוכשת היא הגדולה:

$$Bought(r\_games) + = Bought(All\_games\_played) - Buyer(All\_games\_played) - Buyer(r\_games)$$

- אם הקבוצה הנרכשת היא הגדולה:

$$Bought(r\_games) + = Bought(All\_games\_played) - Buyer(All\_games\_played)$$

$$Buyer(r\_games) + = Buyer(All\_games\_played) - Bought(All\_games\_played) - Bought(r\_games)$$

עדכונים אלו יבטיחו כי בעת סכימה/הרכבה לאורך מסלול החיפוש נקבל את הערך הרצוי.
לבסוף נעדכן את השדות של הקבוצה הרוכשת (הנקודות שלה, ה-ability הכולל שלה, והשדה האם קיים שוער) ונכניסה מחדש את עץ הקבוצות לפי Ability(כאשר לפני שינויי ה-ability הכולל הסרנו אותה על מנת למנוע כפילויות) כאשר הכנסה והוצאה מהעץ קוראות בסיבוכיות של $O(logk)$ כפי שנלמד בהרצאה.
נשים לב שכיוון שפונקציה זו פועלת באותו אופן כמו פונקציית $union(p,q)$ עם איחוד לפי גודל (מלבד כמות פעולות קבועה) שערוך שלה עם פונקציות הפועלות כמו $find(x)$ הכוללת כיווץ מסלולים,יביא לסביכויות משוערכת של $O(\log^* n)$, סה"כ נקבל סיבוכיות של $O(\log k + \log^* n)$ משוערך.

* כל הפונקציות המשוערכות למעט add_player משוערכות אחד עם השנייה מאחר וחלקן משתמשות בפונקציות Union וחלקן ב-Find, וכפי שנלמד בהרצאה השיערוך של שתיהן הינו $O(log*n)$ ולכן כשנשערך אותך נקבל כי הסיבוכיות של כל אחת מהפונקציות הינה נכונה בצורה משוערכת כנדרש.