מבני נתונים 1

234218

הוגש ע"י:

מספר

תרגיל רטוב

1

305664039

שי אגרוסקין

מספר זהות

שם

203762042

מורן רובינשטיין

מספר זהות

שם

ציון:

לפני בונוס הדפסה:

כולל בונוס הדפסה:

נא להחזיר לתא מס':

41-42

**כתובות מייל:** שי- agroskinshay@campus.technion.ac.il

מורן- moranr@campus.technion.ac.il

**מבני הנתונים:**

* **Oasis:**

1. **players\_by\_id**
   * מבנה הנתונים: AvlTree<Player\*,Player::Comp\_by\_id> - עץ AVL של שחקנים ממויין לפי player id
   * בכל צומת יש מצביע לשחקן שמכיל את השדות הבאים:
     + *id –* מזהה השחקן (int)
     + *coins –* מספר המטבעות שצבר (int).
     + *num\_of\_challenges* – מספר האתגרים שביצע (int)
   * סיבוכיות מקום: O(n)
2. **players\_by\_coins**
   * מבנה הנתונים: AvlTree<Player\*,Player::Comp\_by\_coins> - עץ AVL של שחקנים ממויין לפי coins, מיון משני לפי id.
   * בכל צומת יש מצביע לשחקן שמכיל את השדות הבאים:
     1. *id –* מזהה השחקן (int)
     2. *coins –* מספר המטבעות שצבר (int).
     3. *num\_of\_challenges* – מספר האתגרים שביצע (int)
   * סיבוכיות מקום: O(n)
3. **best\_player**
   * מבנה הנתונים: מצביע לשחקן שהשלים הכי הרבה אתגרים (Player\*)
   * סיבוכיות מקום: O(1)
4. **clans**
   * מבנה הנתונים: AvlTree<Clan\*,Clan::Comp\_by\_id> - עץ AVL של שבטים ממויין לפי id
   * בכל צומת יש מצביע לשבט שמכיל את השדות הבאים:
     1. *id –* מזהה השבט (int)
     2. *players\_by\_coins –* עץ AVL של שחקני השבט ממויין לפי coins (AvlTree<Player\*,Player::Comp\_by\_coins>).
     3. *players\_highest\_score* – מצביע לשחקן שהשלים הכי הרבה אתגרים בקבוצה ((Player\*
     4. *compare –* אופרטור השוואת שחקנים לפי מספר האתגרים (Player::comp\_by\_score)
   * סיבוכיות מקום: O(n+k)

***סה"כ סיבוכיות מקום: O(n+k)***

**הפעולות:**

1. void\* init() – מאתחל מבנה נתונים ריק (במקרה של כישלון מחזיר NULL)
   * אתחול players\_by\_id – עץ AVL ריק - סיבוכיות זמן: O(1)
   * אתחול players\_by\_coins – עץ AVL ריק - סיבוכיות זמן: O(1)
   * אתחול best\_player ל-NULL - סיבוכיות זמן: O(1)
   * אתחול clans – עץ AVL ריק – סיבוכיות זמן: O(1)

***סה"כ סיבוכיות זמן: O(1)***

1. StatusType addPlayer(int *playerID*, int *initialCoins*) - הוספת שחקן בעל המזהה *playerID* ובעל סכום התחלתי של initialCoins, אשר לא משתייך לשבט
   * בדיקת תקינות קלט -סיבוכיות זמן: O(1)
     1. הקלט לא תקין 🡸 החזרת INVALID\_INPUT – סיבוכיות זמן: O(1)
     2. הקלט תקין 🡸 בדיקת קיום שחקן בעל המזהה PlayerID ב-players\_by\_id – סיבוכיות זמן: O(log n)
        1. השחקן קיים 🡸 החזרת FAILURE – סיבוכיות זמן: O(1)
        2. השחקן לא קיים 🡸 יצירת Player בעל הנתונים הנ"ל – סיבוכיות זמן: O(1)
           1. היצירה נכשלה 🡸 החזרת ALLOCATION\_ERROR - סיבוכיות זמן: O(1)
           2. היצירה הצליחה 🡸

הוספת השחקן ל-players\_by\_id ול-players\_by\_coins – סיבוכיות זמן: O(log n)

עדכון best\_player במידת הצורך – סיבוכיות זמן: O(1)

החזרת SUCCESS – סיבוכיות זמן: O(1)

***סה"כ סיבוכיות זמן: O(log n)***

1. StatusType addClan(int *clanID*) - הוספת שבט חדש עם המזהה *clanID* 
   * בדיקת תקינות קלט -סיבוכיות זמן: O(1)
     1. הקלט לא תקין 🡸 החזרת INVALID\_INPUT – סיבוכיות זמן: O(1)
     2. הקלט תקין 🡸 בדיקת קיום שבט בעל המזהה clanID ב-clans – סיבוכיות זמן: O(log k)
        1. השבט קיים 🡸 החזרת FAILURE – סיבוכיות זמן: O(1)
        2. השבט לא קיים 🡸 יצירת Clan בעל הנתונים הנ"ל – סיבוכיות זמן: O(1)
           1. היצירה נכשלה 🡸 החזרת ALLOCATION\_ERROR - סיבוכיות זמן: O(1)
           2. היצירה הצליחה 🡸

הוספת השבט ל-clans – סיבוכיות זמן: O(log k)

החזרת SUCCESS – סיבוכיות זמן: O(1)

***סה"כ סיבוכיות זמן: O(log k)***

1. StatusType joinClan(int *playerID,* int *clanID*) - הוספת השחקן playerID לשבט*clanID* 
   * בדיקת תקינות קלט -סיבוכיות זמן: O(1)
     1. הקלט לא תקין 🡸 החזרת INVALID\_INPUT – סיבוכיות זמן: O(1)
     2. הקלט תקין 🡸 בדיקת קיום שחקן בעל המזהה playerID ב-players\_by\_id ושבט בעל המזהה clanID ב-clans ושהשחקן לא משתייך לאף שבט – סיבוכיות זמן: O(log n + log k)
        1. השחקן או השבט לא קיימים או שהשחקן משתייך כבר לשבט 🡸 החזרת FAILURE – סיבוכיות זמן: O(1)
        2. השחקן והשבט קיימים והשחקן לא משתייך לאף שבט 🡸 הוספת השחקן לשבט – סיבוכיות זמן: O(log n)
           1. ההוספה נכשלה 🡸 החזרת ALLOCATION\_ERROR - סיבוכיות זמן: O(1)
           2. ההוספה הצליחה 🡸

עדכון *players\_highest\_score* ו- best\_player במידת הצורך – סיבוכיות זמן: O(1)

החזרת SUCCESS – סיבוכיות זמן: O(1)

***סה"כ סיבוכיות זמן: O(log n + log k)***

1. StatusType completeChallenge(int *playerID,* int *coins*) – השחקן בעל המזהה playerID השלים אתגר בהצלחה ולכן קיבל מטבעות
   * בדיקת תקינות קלט -סיבוכיות זמן: O(1)
     1. הקלט לא תקין 🡸 החזרת INVALID\_INPUT – סיבוכיות זמן: O(1)
     2. הקלט תקין 🡸 בדיקת קיום שחקן בעל המזהה playerID ב-players\_by\_id – סיבוכיות זמן: O(log n)
        1. השחקן לא קיים 🡸 החזרת FAILURE – סיבוכיות זמן: O(1)
        2. השחקן קיים 🡸 עדכון המטבעות ומספר האתגרים של השחקן – סיבוכיות זמן: O(1)
           1. העדכון נכשל 🡸 החזרת ALLOCATION\_ERROR - סיבוכיות זמן: O(1)
           2. העדכון הצליח 🡸

עדכון *players\_highest\_score* ו- best\_player במידת הצורך – סיבוכיות זמן: O(1)

החזרת SUCCESS – סיבוכיות זמן: O(1)

***סה"כ סיבוכיות זמן: O(log n)***

1. StatusType getBestPlayer(int *clanID*, int *playerID*) – החזרת מזהה השחקן שהשלים הכי הרבה אתגרים בקבוצה *clanID* (במידה וחיובי) או במערכת (במידה ושלילי)
   * בדיקת תקינות קלט -סיבוכיות זמן: O(1)
     1. הקלט לא תקין 🡸 החזרת INVALID\_INPUT – סיבוכיות זמן: O(1)
     2. הקלט תקין 🡸
        + אם clanID<0: עדכון playerID ב-best\_player והחזרת SUCCESS – סיבוכיות זמן: O(1)

***סה"כ סיבוכיות זמן: O(1)***

* + - * אם clanID>0: בדיקת קיום שבט בעל המזהה clanID ב-clans – סיבוכיות זמן: O(log k)
      1. השבט לא קיים 🡸 החזרת FAILURE – סיבוכיות זמן: O(1)
      2. השבט קיים 🡸 עדכון playerID ב-*players\_highest\_score* של השבט והחזרת SUCCESS – סיבוכיות זמן: O(1)

***סה"כ סיבוכיות זמן: O(log k)***

1. StatusType getScoreBoard(int *clanID,* int\*\* *players,* int\* *numOfPlayers*) – החזרת השחקנים שנמצאים בשבט clanID (במידה וחיובי) או במערכת (במידה ושלילי), ממוינים על פי כמות המטבעות שלהם.
   * בדיקת תקינות קלט -סיבוכיות זמן: O(1)
     1. הקלט לא תקין 🡸 החזרת INVALID\_INPUT – סיבוכיות זמן: O(1)
     2. הקלט תקין 🡸
        + אם clanID<0:
        1. עדכון numOfPlayers לפי ה-size של העץ players\_by\_coins של המערכת – סיבוכיות זמן: O(1)
        2. הקצאת מערך בגודל numOfPlayers ש-players יצביע אליו
           1. ההקצאה נכשלה 🡸 החזרת ALLOCATION\_ERROR - סיבוכיות זמן: O(1)
           2. ההקצאה הצליחה 🡸

עדכון המערך באמצעות מעבר על players\_by\_coins של המערכת והכנסת השחקנים ממוינים על פי כמות המטבעות שלהם – סיבוכיות זמן: O(n)

החזרת SUCCESS – סיבוכיות זמן: O(1)

***סה"כ סיבוכיות זמן: O(n)***

* + - * אם clanID>0:

1. בדיקת קיום שבט בעל המזהה clanID ב-clans – סיבוכיות זמן: O(log k)
   * + - 1. השבט לא קיים 🡸 החזרת FAILURE – סיבוכיות זמן: O(1)
         2. השבט קיים 🡸

עדכון numOfPlayers לפי ה-size של העץ players\_by\_coins של השבט – סיבוכיות זמן: O(1)

הקצאת מערך בגודל numOfPlayers ש-players יצביע אליו

ההקצאה נכשלה 🡸 החזרת ALLOCATION\_ERROR - סיבוכיות זמן: O(1)

ההקצאה הצליחה 🡸

עדכון המערך באמצעות מעבר על players\_by\_coins של השבט והכנסת השחקנים ממוינים על פי כמות המטבעות שלהם – סיבוכיות זמן: O()

החזרת SUCCESS – סיבוכיות זמן: O(1)

***סה"כ סיבוכיות זמן: O***

1. StatusType uniteClans(int *clanID1*, int *clanID2*) – איחוד שני השבטים clanID1 ו-clanID2 כך שכל השחקנים בשבט החדש השלימו אתגרים
   * בדיקת תקינות קלט -סיבוכיות זמן: O(1)
     1. הקלט לא תקין 🡸 החזרת INVALID\_INPUT – סיבוכיות זמן: O(1)
     2. הקלט תקין 🡸 בדיקת קיום שבטים בעלי המזהים clanID1 ו-clanID2 ב-clans – סיבוכיות זמן: O(log k)
        1. אחד מהשבטים לא קיים 🡸 החזרת FAILURE – סיבוכיות זמן: O(1)
        2. שני השבטים קיימים 🡸
           1. מעבר על העצים players\_by\_coins של שני השבטים והעברתם לשני מערכים – סיבוכיות זמן O(n)
           2. ביצוע merge בין שני המערכים כך שהשחקנים שיעברו לשבט החדש הם רק אלה שביצעו אתגרים (אלה שלא יהיו מחוסרי שבט) – O(n)
           3. יצירת עץ ריק בגודל מספר השחקנים שביצעו אתגרים, והכנסת האיברים מהמערך הממוזג לעץ שבנינו – סיבוכיות זמן: O(n)
           4. יצירת שבט חדש:

עדכון id לפי השבט הגדול יותר – סיבוכיות זמן: O(1)

עדכון *players\_highest\_score* לפי הגבוה בין שני השבטים – סיבוכיות זמן: O(1)

הגדרת העץ players\_by\_coins להיות העץ שיצרנו בסעיף c – סיבוכיות זמן: O(1)

* + - * 1. הסרת שני השבטים מהעץ clans והוספת השבט החדש – סיבוכיות זמן: O(log k)
        2. החזרת SUCCESS – סיבוכיות זמן: O(1)

***סה"כ סיבוכיות זמן: O(n + log k)***

1. void Quit() – שחרור המבנה
   * מעבר על players\_by\_id ועל players\_by\_coins, ושחרור הזכרון שהקצנו לשחקנים + הריסת שני העצים - סיבוכיות זמן: O(k)
   * מעבר על clans, ושחרור הזכרון שהקצנו לשבטים + הריסת העץ – סיבוכיות זמן: O(n+k) כי כל שחקן שייך לכל היותר לשבט אחד
   * הצבת ערך NULL במבנה הנתונים – סיבוכיות זמן: O(1)

***סה"כ סיבוכיות זמן: O(n+k)***