# מבני נתונים 1

# 234218

## תרגיל רטוב

| |
|---|
| 1 |

מספר

הוגש ע"י:

| | |
|---|---|
| חגי קריטי | 301781613 |
| שם | מספר זהות |
| תם נלסון | 204805824 |
| שם | מספר זהות |

<u>ציון</u>:

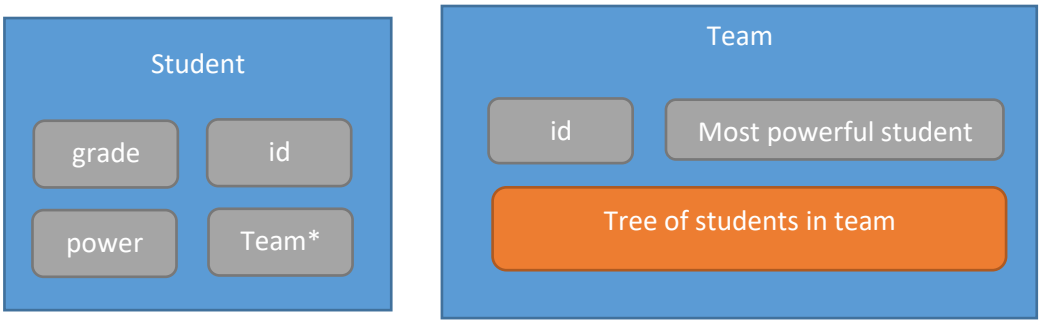
לפני בונוס הדפסה:

כולל בונוס הדפסה:

נא להחזיר לתא מס':

13

יש לנו שני class-ים עיקריים במערכת: Student ו- Team.




עבור הסטודנט, הוא מחזיק את הid, grade, power שלו, ובנוסף מחזיק פוינטר לרשומה של הTeam שלו.

בכל קבוצה יש את הid שלה, את הסטודנט החזק ביותר (student id שלו) ועץ AVL מסודר לפי power בסדר יורד וstudent id בסדר עולה.

הסטודנטים מסודרים ב-2 עצי AVL. אחד מסודר לפי id (idTree), ואחד מסודר לפי power בסדר יורד ו-student id בסדר עולה (powerTree).

הקבוצות גם מסודרות בעץ AVL שמסודר לפי team id (teamTree).

בנוסף, שומרים int שהוא mostPowerfulStudent גובלי של המערכת.

Init() :

באיתחול מאתחלים 3 עצים ריקים (idTree, powerTree, teamTree) וint אחד, ולכן הפעולה היא O(1).

**AddStudent() :**

מחפשים בעץ לראות אם הסטודנט כבר קיים (O(logn))

אם לא, מוסיפים את הסטודנט שמקבלים ל-2 עצי AVL (O(2logn))

בנוסף, מוצאים את הסטודנט החזק ביותר באמצעות הליכה לאיבר הגדול ביותר בpowerTree ושומרים את הid שלו בmostPowerfulStudent (O(logn))

כלומר 3*log(n) שזה O(logn) פעולות.

**AddTeam() :**

מחפשים אם הקבוצה כבר קיימת בעץ (O(logk))

מוסיפים קבוצה חדש לעץ הקבוצות. מאתחלים עץ ריק בתוכו (O(1)), ומוסיפים אותו לעץ AVL (O(logk)

כלומר סה"כ זה O(logk) פעולות.

**MoveStudentToTeam () :**

לחפש את הקבוצה בעץ ((O(logk), ומחפשים אם הסטודנט קיים ((O(logn).

אם שניהם קיימים, הולכים לסטודנט ((O(logn), ומוציאים אותו מהקבוצה הישנה שלו ((O(1 למצוא את הקבוצה בteamTree (זה פוינטר), ואז (O(logn כדי להוציא את הסטודנט מהעץ של הקבוצה, ועוד (O(logn כדי למצוא את הסטודנט הכי חזק באותה קבוצה שוב (מוצאים את האיבר הגדול ביותר בעץ של הקבוצה)).

מחפשים את הקבוצה החדשה (O(logk), מוסיפים את הסטודנט לעץ סטודנטים שלה ((O(logn), מחפשים מחדש את הסטודנט החזק ביותר ((O(logn) ומחליפים את הפוינטר לקבוצה של הסטודנט לקבוצה החדשה שלו ((O(1).

סה"כ, עשינו (O(logn + logk פעולות.

**<u>GetMostPowerful() :</u>**

אם מחפשים את האדם החזק ביותר בקבוצה:

מחפשים את הקבוצה בteamTree ((O(logk) ומחזירים את הid של הסטודנט החזק ביותר שלה.

סה"כ (O(logk.

אם מחפשים את האדם החזק ביותר במערכת:

מחזירים את הid השמור בmostPowerfulStudent.

סה"כ (O(1.

**<u>RemoveStudent() :</u>**

מחפשים את הסטודנט בעץ הסטודנטים idTree ((O(logn).

משתמשים בפוינטר לקבוצה שלו כדי להסיר אותו מהקבוצה, כלומר הוצאה שלו מתוך העץ של הקבוצה ((O(logn) והחלפת הסטונדט החזק ביותר בקבוצה ((O(logn).

משתמשים בpower והid שלו כדי להוציא אותו מהעץ powerTree ((O(logn), ומחפשים מחדש את הסטודנט החזק ביותר במערכת ((O(logn).

מוציאים אותו מidTree ((O(logn)

סה"כ: (O(logn

**<u>getAllStudentByPower() :</u>**

אם זה עבור קבוצה מסויימת:

מוצאים את הקבוצה בעץ הקבוצות ((O(logk)

עושים סיור inorder בעץ סטודנטים שלה ((O(n_team), ואז מקצים מערך בגודל זה וממלאים אותו ע"י סיור inorder נוסף בעץ ((O(n_team).

סה"כ (O(logk + n_team

אם זה עבור כלל המערכת:

עורכים סיור inorder בpowerTree וסופרים כמה רשומות יש ((O(n), לאחר מכן מקצים מערך בגודל זה וממלאים אותו ע"י סיור inorder נוסף בעץ ((O(n)

סה"כ (O(n.

**IncreaseLevel() :**

עוברים על idTree וסופרים כמה סטודנטים יש בgrade ((O(n)).

מקצים מערך בגודל זה, ושומרים בו את הסטודנטים בgrade ((O(n)).

עוברים על המערך ומעדכנים לכל סטודנט את הpower ((O(n)

כעת עושים merge למערך וpowerTree באופן הבא ((O(n):
עוברים על המערך עם איטרטור (currentArray), ועוברים על העץ עם 2 איטרטורים (writeTree, readTree) בסיור inorder.
בכל שלב, משווים את הסטודנט בcurrentArray ובreadTree וכותבים למקום בwriteTree את הגדול מביניהם. לאחר הכתיבה מקדמים את האיטרטור שכתבנו את ערכו, כאשר readTree מדלג על רשומות בעץ בעלות grade זהה לgrade שעובר עדכון. את האיבר האחרון בעץ (הסטודנט החזק ביותר) שומרים בmostPowerfulStudent.

כעת עוברים על העץ teamTree בסיור inorder ((O(k). כאשר עבור כל קבוצה עושים תהליך זהה למתואר מעלה לעץ ששייך לקבוצה, כאשר את האיבר האחרון שומרים בסטודנט החזק ביותר של הקבוצה. נשים לב כי המעבר הנ"ל לא יוצר סיבוכיות של (O(kn משום שבסה"כ בכל הצוותים ישנם n איברים בעצים ולכן הפעולה לוקחת (O(n+k.

סה"כ (O(n+k פעולות.

**Quit() :**

עוברים ומשחררים את 3 העצים, עבור העצים של הסטודנטים זה (O(n ועבור העץ של הקבוצות, צריך לשחרר את העץ הפנימי גם, אבל כמו שנאמר בIncreaseLevel המעבר על כלל העצים הפנימיים הוא סה"כ n ולכן זה ((O(n+k).

סה"כ (O(n+k.

נשים לב כי, במקרה הכי גרוע, יש במערכת מקום של 2 עצים של סטודנטים כל אחד בגודל n, עץ של קבוצות בגודל k כאשר בתוכן יש סה"כ עוד שווה ערך לעץ אחד בגודל n ועוד מערך בגודל n. כלומר הסיבוכיות מקום אינה עולה על (O(n+k.