מבני נתונים – תרגיל בית 2

נמרוד קדיש ואלכס בלגודרסקי

4.5.17

חלק יבש

1 תיאור מבנה הנתונים

<u>כללי:</u>

מבנה הנתונים הראשי יקרא $Xmen$. מבנה הנתונים יהיה מורכב משלושה עצי AVL עיקריים. בעצים אלו ישמרו אובייקטים שטיפוסם יהיה צוות או סטודנט שיתוארו בהמשך.

*מצורף שרטוט של מבנה הנתונים בסוף החלק היבש.

<u>אובייקטים מרכזיים</u>

- אובייקט סטודנט:

אובייקט זה מייצג סטודנט במערכת $Xmen$. הוא יכיל את השדות הבאים:

- ID - מזהה הסטודנט במערכת.

- GRADE - הכיתה שבה נמצא הסטודנט

- PWR - כוחו של הסטודנט

- מצביע לאובייקט צוות שאליו הוא שייך בעץ ה- AVL של הצוותים שממוין לפי מס' צוות.

- אובייקט צוות:

אובייקט זה מייצג צוות במערכת $Xmen$. הוא יכיל את השדות הבאים:

- ID – מס' צוות, מזהה הצוות במערכת.

- ID וPOWER של הסטודנט החזק ביותר בצוות.

- מצביע לעץ של כל הסטודנטים הנמצאים בצוות ממוין לפי POWER (ואז ID).

<u>עצים עיקריים</u>

- עץ AVL צוותים(ממוין לפי מס' צוות):

  עץ זה יכיל את כל הצוותים הקיימים במערכת ממויינים על פי מס' הצוות שלהם.

- עץ AVL סטודנטים(ממוין לפי ID):

  עץ זה יכיל את כל הסטודנטים הקיימים במערכת ממויינים על פי STUDENTID.

- עץ AVL סטודנטים (ממוין לפי POWER ואז ID):

  עץ זה יכיל את כל הסטודנטים הקיימים במערכת ממויינים על פי הPOWER שלהם.

- תא סטודנט הכי חזק:

  תא זה יחזיק בכל שלב את הid ואת הpower של הסטודנט הכי חזק הנמצא במערכת כעת.

2 תיאור הפעולות

$Init()$:

1. יצירה ואתחול של שלושת עצי ה-AVL העיקריים. לוקח O(1) (יצירת צומת שורש ריק בכל עץ)
2. הקצאה ואתחול של תא הסטודנט החזק ביותר גם לוקח O(1).

ניתוח סיבוכיות: בסה"כ פעולת אתחול מבנה הנתונים לוקחת O(1).

$AddStudent()$:

1. ניצור ונאתחל אובייקט סטודנט, אתחול זה לוקח O(1) פעולות (אתחול מס' קבוע של תאי זיכרון) ונוסיף אותו כאיבר חדש לעץ הסטודנטים הממוינים על פי ID (פעולה זו לוקחת O(log(n)) כאשר n הוא מס' הסטודנטים במערכת).
2. נאתחל אובייקט סטודנט נוסף זהה ונוסיף כאיבר חדש לעץ הסטודנטים הממוינים על פי POWER (פעולה זו לוקחת O(log(n)) כאשר n הוא מס' הסטודנטים במערכת).
3. נבדוק את תא הסטודנט הכי חזק הכללי ונעדכן אותו בהתאם(כלומר אם הסטודנט שהוספנו בעל POWER גדול יותר אזי נחליף את הערכים המתאימים). פעולות אלה לוקחות O(1).

ניתוח סיבוכיות: מספר הפעולות שנבצע הוא במקרה הגרוע O(log(n)) כנדרש.

AddTeam:

1. ניצור ונאתחל אובייקט צוות, אתחול זה לוקח O(1) פעולות (אתחול מס' קבוע של תאי זיכרון ואתחול עץ AVL)
ונוסיף כאיבר חדש לעץ הצוותים הממוינים לפי ID של הצוות (פעולה זו לוקחת O(log(k)) כאשר K הוא מס' הצוותים במערכת.)

לסיכום, מספר הפעולות שנבצע הוא במקרה הגרוע O(log(k)) כנדרש.

MoveStudentToTeam:

1. תחילה נחפש את הסטודנט על פי הID בעץ הסטודנטים הממוין לפי ID. (פעולה זו לוקחת O(log(n)) כאשר n הוא מס' הסטודנטים במערכת).
2. ניגש למצביע לצוות שלו ונבדוק האם הוא NULL:
   - אם המצביע הוא NULL:

1. נחפש את הצוות שאליו ברצוננו להוסיף את הסטודנט בעץ הצוותים הממוין לפי ID של הצוותים. אם הצוות לא קיים, נחזיר שגיאה בהתאם ונסיים. (פעולה זו לוקחת O(log(k)) כאשר K הוא מס' הצוותים במערכת).
2. ניגש לעץ שמחזיק את האובייקט של הצוות שמצאנו (O(1)) ונוסיף בו את הסטודנט (פעולה זו לוקחת O(log(m)=O(log(n)) כאשר n הוא מס' הסטודנטים במערכת, ו-m<n מס' הסטודנטים ששייכים לצוות המסויים שמצאנו).
3. נשנה את המצביע כעת כך שיצביע על אובייקט הצוות החדש שלו, ואת המצביעים של הסטודנט בעצי הסטודנטים כך שיצביעו על הצוות החדש (חיפוש ושינוי המצביעים לוקח O(2*log(n)) במקרה הגרוע.)
4. בנוסף נבדוק את תא הסטודנט החזק ביותר בצוות החדש ונעדכן בהתאם (אם הסטודנט שהוכנס כרגע בעל POWER גדול יותר, אז הוא יהיה כעת הסטודנט החזק ביותר ונשנה את ערכי התא בהתאם – O(1))

   - אם המצביע שונה מNULL:

1. אם הצוות החדש הוא גם הצוות הנוכחי, נסיים ונחזיר הצלחה.
2. ניגש לצוות שאליו מצביע המצביע ונסיר מעץ הסטודנטים של הצוות הנ"ל את הסטודנט שברצוננו להעביר (פעולה זו לוקחת O(log(m)=O(log(n)) כאשר n הוא מס' הסטודנטים במערכת, ומ מס' הסטודנטים ששייכים לצוות המסויים שמצאנו).
*אם הוא היה הסטודנט החזק ביותר בצוות, ניגש לעץ הסטודנטים של הצוות הנוכחי שממוין לפי POWER לאיבר התחתון הימני ביותר ונרשום בתא הסטודנט החזק ביותר בצוות את נתוני הסטודנט שנמצא בו. O(log(m))=O(log(n)) כנ"ל.
3. נחפש את הצוות שאליו ברצוננו להוסיף את הסטודנט בעץ הצוותים הממוין לפי ID של הצוותים. (פעולה זו לוקחת O(log(k)) כאשר K הוא מס' הצוותים במערכת).
4. ניגש לעץ שמחזיק את האובייקט של הצוות שמצאנו ונוסיף בו את הסטודנט (פעולה זו לוקחת O(log(m)=O(log(n)) כאשר n הוא מס' הסטודנטים במערכת, ומ מס' הסטודנטים ששייכים לצוות המסויים שמצאנו).

5. נשנה את המצביע כעת כך שיצביע על אובייקט הצוות החדש שלו, ואת המצביעים של הסטודנט בעצי הסטודנטים כך שיצביעו על הצוות החדש (חיפוש ושינוי המצביעים לוקח O(logn) במקרה הגרוע.)
6. בנוסף נבדוק את תא הסטודנט החזק ביותר בצוות הנוכחי ונעדכן בהתאם(אם הסטודנט שהוכנס כרגע בעל POWER גדול יותר, אז הוא יהיה כעת הסטודנט החזק ביותר ונשנה את ערכי התא בהתאם – O(1))

ניתוח סיבוכיות: העברת הסטודנט הינה O(log(n)+log(k)) כנדרש.

GetMostPowerful:

לפונקציה זו 2 אופני פעולה שונים התלויים בקלט teamID:

- במידה וteamID קטן מ-0 נחזיר את ה-ID של הסטודנט החזק ביותר במערכת, השמור בתא הזיכרון, ונסיים. (O(1))

- אחרת, נחפש את הצוות בעץ הצוותים בעזרת הteamID שלו. פעולה זו לוקחת O(log(k)) כאשר K הוא מס' הצוותים במערכת). ניגש אל הצומת של העץ שמצאנו, נחזיר את מזהה הסטודנט החזק ביותר בצוות הנוכחי – (אם לא מאותחל – כלומר אין סטודנטים בעץ אז נחזיר 1- ואחרת נחזיר על פי ערך התא השמור באובייקט (O(1))

ניתוח סיבוכיות: מציאת הסטודנט החזק ביותר בקבוצה מסויימת הוא O(1) או O(log(k)) כנדרש.

RemoveStudent:

1. תחילה נמצא את הסטודנט בעזרת הID שלו בעץ הסטודנטים הממוין לפי הID. (פעולה זו לוקחת O(log(n)) כאשר n הוא מס' הסטודנטים במערכת). נשמור בצד את המצביע לצוות שבו הוא נמצא ואת עצמתו.
2. נסיר את האובייקט מעץ הסטודנטים הממוין לפי ID ולאחר מכן נסירו מן העץ הממויין לפי POWER. כל אחת מפעולות ההסרה לוקחת O(log(n)) כאשר n הוא מס' הסטודנטים במערכת.
3. ניגש לצוות שבו נמצא הסטודנט(על פי המצביע ששמרנו קודם\אם אין לו צוות נדלג על שלב זה) – גישה כזאת תעשה ב(O(1. נסיר מן העץ את הסטודנט הרלוונטי (פעולה זו לוקחת O(log(m)=O(log(n)) כאשר n הוא מס' הסטודנטים במערכת, ml מס' הסטודנטים ששייכים לצוות המסויים שמצאנו).
4. לבסוף, נבדוק האם הסטודנט הנ"ל הוא הסטודנט החזק ביותר במערכת, במידה ולא נסיים. אחרת, ניגש לעץ הסטודנטים הממויין לפי POWER ונלך "ימינה" עד הסוף בעץ עד שנגיע לעלה, מכיוון שהעץ הוא עץ AVL ובפרט עץ חיפוש זהו הסטודנט החזק ביותר כעת, נשמור אותו בתא הסטודנט החזק ביותר. המסלול הנ"ל הוא במקרה הגרוע ביותר באורך O(log(n)) כפי שהוכח בהרצאה.
5. נבצע פעולה דומה לשלב 4 בעץ הסטודנטים שבצוות שמצאנו, אם הסטודנט אינו הסטודנט החזק ביותר נסיים. אחרת נלך בעץ הסטודנטים שנמצאים בצוות "ימינה" עד הסוף, וכעת זהו הסטודנט החזק ביותר בצוות, נעדכן את הסטודנט החזק ביותר בצוות ונסיים. (פעולה זו לוקחת

(O(log(m)=O(log(n)) כאשר n הוא מס' הסטודנטים במערכת, ומ מס' הסטודנטים ששייכים לצוות המסויים שמצאנו).

ניתוח סיבוכיות: מספר הפעולות שנעשה במקרה הגרוע ביותר הוא (O(log(n)) .

GetAllStudentsByPower:

1. לפונקציה זו שני אופני פעולה שונים התלויים בקלט teamID:
   - אם ערך זה שלילי אז:
   נלך לעץ הסטודנטים הממויין לפי POWER ונבצע סיור inorder הפוך, כך נקבל את הסטודנטים ממויינים כנדרש. פעולה זו לוקחת O(n) כאשר n הוא מס' הסטודנטים במערכת.
   - אחרת, אם teamID>0:
   נחפש את הצוות בעץ הצוותים הממויין (פעולה זו לוקחת (O(log(k)) . נבצע סיור inorder הפוך בעץ הסטודנטים של הצוות שמצאנו וכך נקבל את היצורים ממוינים כנדרש. (פעולה זו לוקחת O(m) כאשר m הוא מס' הסטודנטים בצוות TEAMID) ולכן במקרה זה סיבוכיות הפעולה היא O(log(k)+m) כנדרש.
   - אם teamID=0 נחזיר שגיאה בהתאם (O(1)).

$IncreaseLevel$:

1. נעבור על העץ הכללי של הסטודנטים הממוינים לפי ID ב-in-order. עבור כל סטודנט, נבדוק האם הוא שייך ל-grade: אם כן, נעלה את עצמתו. נשים לב כי לא נשנה את טופולוגית העץ, מכיוון שהוא אינו ממויין לפי POWER. בסה"כ מעבר in-order על העץ ייקח O(n) פעולות והעדכון O(1) לכל סטודנט במקרה הגרוע – בסה"כ O(n).
2. עבור כל עץ סטודנטים הממוין לפי power (העץ הכללי ובנוסף k עצים – אחד לכל צוות):
   a. נעבור על העץ בסיור in-order ונספור את כמות הסטודנטים שנמצאים ב-grade. בעץ הכללי זה מתבצע ב-O(n) וב-k העצים של הצוותים יש בסה"כ n סטודנטים ונעבור על כל אחד פעם אחת בלבד ולכן O(n+k) בסה"כ.
   b. ניצור שני מערכים בעזרת סיור in-order נוסף. O(n) במקרה הגרוע:
      i. <u>מערך A</u>: יכיל את כל הסטודנטים שנמצאים ב-grade.
      ii. <u>מערך B</u>: יכיל את כל שאר הסטודנטים בעץ.
   c. נעבור על מערך A ונגדיל את ה-power של כל הסטודנטים ב-powerIncrease. O(n)במקרה הגרוע.
   d. נבצע merge בין שני המערכים ונשמור את ה-ID וה-power של התא האחרון במערך החדש (זהו הסטודנט החזק ביותר בעץ החדש) ונעדכן בהתאם. Merge מתבצע בסיבוכיות $O(n_1+n_2)$ כאשר ה-n-ים מייצגים גודל המערכים ולכן O(n) במקרה הגרוע.
   e. נבנה עץ חדש מתוך המערך שהתקבל, שיחליף את העץ הישן. בהרצאה ראינו איך לבצע זאת ב-O(n).

   ניתוח סיבוכיות: בסה"כ האלגוריתם יתבצע ב-O(n+k) במקרה הגרוע, כנדרש.

:Quit

*נציין שמחיקת עץ AVL בעל n איברים לוקחת O(n) מכיוון שנדרש מעבר על כל האיברים שבעץ.

1. תחילה נמחק את עץ הסטודנטים הממויינים לפי הID. פעולה זו לוקחת O(n) כאשר n הוא מס' הסטודנטים(כל מחיקה של צומת בעץ גם מוחקת את האובייקט המשוייך אליו – פעולה זו במקרה של אובייקט סטודנט תיקח O(1)).

2. נמחק את עץ הסטודנטים הממויינים לפי POWER. פעולה זו לוקחת O(n) כאשר n הוא מס' הסטודנטים(כל מחיקה של צומת בעץ גם מוחקת את האובייקט המשוייך אליו – פעולה זו במקרה של אובייקט סטודנט תיקח O(1)).

3. בנוסף נמחק את עץ הצוותים. מחיקת צומת בעץ הצוותים גוררת את מחיקת הצוות, ומחיקה של כל הסטודנטים בעץ הסטודנטים של הצוות(מחיקת סטודנט בצוות לוקחת O(1) פעולות. נשים לב שבסה"כ לכל הצוותים יחד יש n סטודנטים ולכן מעבר על כל הצוותים ועל כל הסטודנטים שלהם יקח O(n+k).

4. לבסוף נמחק את תא הסטודנט החזק ביותר במערכת. (פעולה זו לוקחת O(1).

לסיכום, בסה"כ סיבוכיות הפעולה היא O(n+k) כנדרש.