מבני נתונים 1

234218

תרגיל רטוב

2

מספר

הוגש ע"י:

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

<u>ציון</u>:

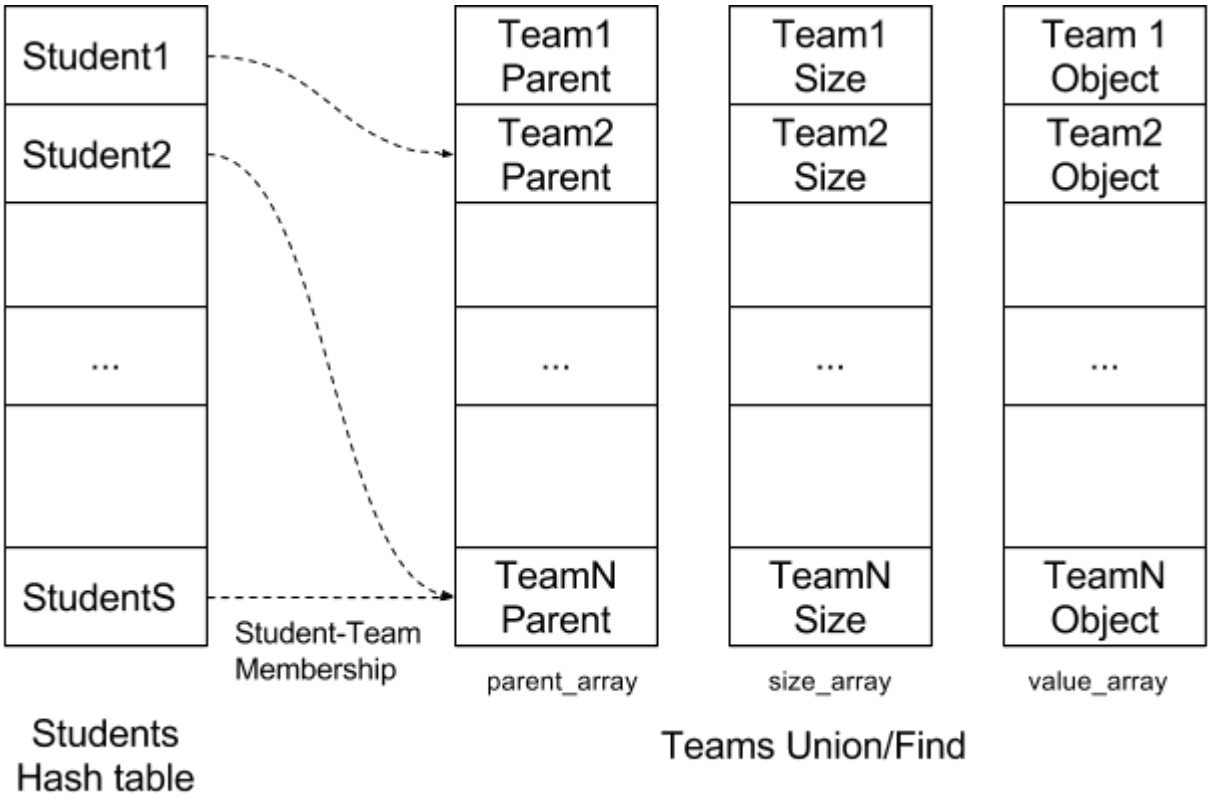
לפני בונוס הדפסה:

כולל בונוס הדפסה:

נא להחזיר לתא מס':

13

המבנה הכללי של הפתרון:



נשתמש ב-2 מבנים עיקריים – Union Find ו-hash table .

נשמור את כל הקבוצות בUnion Find בגודל n. לטובת מימוש הUnion Find, נשתמש ב3 מערכים, כאשר בכל מערך התא באינדקס i שייך לקבוצה i. מערך של int שבמקום ה-i יש את האינדקס של האבא שלו (parent_array), מערך שבמקום ה-i יש את גודל הקבוצה ש-i הוא השורש שלו (size_array) כאשר אם i נמצא תחת קבוצה אחרת – אין למערך זה משמעות עבורו, מערך של ערכים כאשר במקום ה-i יש את הערך ששייך לקבוצה עם המזהה הנ"ל (value_array) – באופן דומה לsize_array אם i נמצא תחת קבוצה אחרת אין משמעות למערך הנ"ל. כאשר כל קבוצה מורכבת באופן דומה לתרגיל הרטוב הקודם, כלומר, בכל קבוצה נשמר: עץ דרגות בינארי מאוזן השומר מצביעים לסטודנטים הנמצאים בקבוצה זו, מספר הניצחונות הכולל של הקבוצה ומי הסטודנט החזק ביותר בקבוצה.

עבור כל קבוצה, נשמור עץ דרגות כפול בינארי מאוזן ששומר מצביעים לסטודנטים הנמצאים בקבוצה זו. בנוסף בכל צומת נשמור את מספר הצמתים מתחת לצומת מכל צד ואת סכום הכוחות של הסטודנטים מתחת לצומת מכל צד (הדרגות בעץ).

בנוסף למבנה זה נשמור hash table שמקשר בין Student_id לבין הסטונדטים. נשתמש בhash table המשתמש בdouble hashing הממומש בתוך מערך דינאמי.

סה"כ הזיכרון של המבנה:

זיכרון עבור הunion find: O(n).

זיכרון עבור הhash table: O(k).

זיכרון כולל של העצים בקבוצות: O(k) (ישנם סה"כ k סטודנטים במערכת, כאשר כל אחד נמצא בקבוצה אחת בלבד – סך כל העצים שמאחסנים סטודנטים הוא לא יותר מO(k)).

סה"כ סיבוכיות הזיכרון היא O(n+k) כנדרש.

הוכחות סיבוכיות לפעולות המבנים:

<u>טבלת ערבול:</u>

<u>Init():</u>

איתחול מערך בגודל התחלתי קבוע - O(1)

איתחול מונה האיברים המלאים ל-0: O(1)

סה"כ: O(1)

<u>Find():</u>

נבצע חיפוש בטבלת ערבול בגודל m כלשהו, שבה מספר האיברים המלאים f הוא לכל היותר m ולכל הפחות $\frac{m}{4}$. לכן פקטור העומס $a = \frac{f}{m} = O(1)$ ולפי מה שנלמד בהרצאה סיבוכיות הזמן במקרה הממוצע היא O(1).

<u>Insert():</u>

תחילה מחפשים את המפתח במערך, ואם הוא קיים מחזירים שגיאה. לאחר מכן מחפשים את הרשומה הפנויה (ריקה או מסומנת כמחוקה) הראשונה, לפי איך שנלמד בהרצאה על ערבול נשנה. כשנמצאה הרשומה, מעתיקים את הערך הנתון ושמים אותו ברשומה שמצאנו. לאחר מכן מקדמים את מונה האיברים המלאים.

אם מספר האיברים המלאים שווה לגודל המערך, מבצעים הגדלה של המערך פי 2: מקצים מערך בגודל 2m ומעתיקים אליו את כל הרשומות המלאות מהמערך הקודם, תוך כדי ערבול מחדש של המפתחות.

סיבוכיות:

חיפוש מפתח קיים במערך: O(1) בממוצע על הקלט (לפי Find())

חיפוש רשומה פנויה: O(1) בממוצע על הקלט, לפי מה שנלמד בהרצאה על double hashing.

העתקה של ערך לתוך המערך: O(1)

הגדלה אפשרית של המערך והעתקת האיברים: O(m)

סה"כ: O(1) בממוצע על הקלט ללא הגדלה, O(m) עם הגדלה.

<u>Remove():</u>

נחפש את המפתח במערך. אם הוא לא קיים נחזיר שגיאה, ואם הוא קיים נסמן את הרשומה שלו כמחוקה. לאחר מכן מקטינים את מונה האיברים המלאים.

אם מספר האיברים המלאים שווה לרבע מגודל המערך וגודל המערך גדול מהגודל ההתחלתי, מבצעים הקטנה של המערך פי 2: מקצים מערך בגודל $\frac{m}{2}$ ומעתיקים אליו את כל הרשומות המלאות מהמערך הקודם, תוך כדי ערבול מחדש של המפתחות.

סיבוכיות:

חיפוש מפתח קיים במערך: O(1) בממוצע על הקלט

סימון רשומה כמחוקה:O(1)

הקטנה אפשרית של המערך והעתקת האיברים: O(m)

סה"כ: O(1) בממוצע על הקלט ללא הקטנה, O(m) עם הקטנה.

הוכחת סיבוכיות משוערכת לפעולות טבלת הערבול:

נשים לב שאחרי כל הגדלה או הקטנה של המערך, מתקיים שמספר התאים המלאים במערך הוא $\frac{m}{2}$ כאשר m הוא גודל המערך לאחר ההגדלה\הקטנה. בדומה למערך דינאמי שראינו בתרגול, ניתן לחלק כל רצף פעולות באורך m ל-k תת-רצפים, כשבסוף כל רצף אנחנו מבצעים הגדלה או הקטנה של המערך. נראה כעת שלכל תת רצף באורך l מתקיים שהסיבוכיות של l הפעולות היא O(l) בממוצע על הקלט.

נסמן ב m את גודל המערך בתחילת הרצף הנוכחי. לפני הפעולה הראשונה מספר התאים המלאים במערך הוא לכל היותר $\frac{m}{2}$ (או שהתבצע שינוי גודל בפעולה האחרונה ברצף הקודם, או שאנחנו בתחילת הרצף והמערך ריק) לכן עד שינוי הגודל הבא של המערך יש לכל הפחות $\frac{m_l}{4}$ פעולות (למקרה של הקטנה), ז"א ש$l =$ $\frac{m}{4} = O(l) \Longrightarrow \Omega\left(\frac{m}{4}\right)$. כל פעולה חוץ מהפעולה האחרונה לוקחת O(1) בממוצע על הקלט מכיוון שלא מתבצע שינוי גודל, והפעולה האחרונה לוקחת O(m) פעולות. לכן סה"כ מספר הפעולות הוא:

$$\Sigma_1^{l-1} 1 + O(m) \leq O(l) + 4O(l) = O(l)$$

וסה"כ הפעולות על כל הרצפים: $\Sigma_{i=1}^{k} O(l_i) = O(l_1 + \cdots + l_k) = O(m)$

לכן עבור כל רצף של m פעולות, הסיבוכיות של פעולות טבלת הערבול הן O(1) משוערך בממוצע על הקלט.

Union\Find:

מבנה הUnion/Find מומש בצורה של עץ הפוך וקיצור מסלולים בפעולת find, לכן לפי ההרצאה הפעולות עליו הן בסיבוכיות משוערכת של O(log*n)

נפרט כעת על סיבוכיות זמן הריצה של כל פונקציה:

Init(n):

מאתחלים hash table ריק – O(1)

מאתחלים מבנה של Union Find: O(n)

הקצאת 3 המערכים בזיכרון: O(1)

מעבר כל מערך ואיתחולו באופן הבא: value_array מקבל קבוצה ריק בכל תא O(n), size_array מקבל 1 בכל תא O(n), parent_array מקבל ערך המסמל "לא קיים" בכל תא O(n).

סה"כ : O(n)

AddStudent(ID, team, power):

מוסיפים את הסטודנט לhash table : O(1) משוערך בממוצע על הקלט.

מוצאים את הקבוצה של הסטודנט: חיפוש בUnion find O(log*n) משוערך

מוסיפים את הסטודנט לקבוצה שלו: הכנסה לעץ דרגות בינארי מאוזן O(logk)

עדכון הסטודנט החזק ביותר בקבוצה: חיפוש איבר מקסימלי בעץ דרגות בינארי מאוזן: O(logk)

סה"כ O(log*n + logk) משוערך בממוצע על הקלט

RemoveStudent(ID):

מוצאים את הסטונדט בhash table: O(1) משוערך בממוצע על הקלט.

לוקחים את האינדקס של הקבוצה מתוך המידע של הסטודנט: O(1)

מוצאים את הקבוצה של הסטודנט: חיפוש בUnion find O(log*n) משוערך

מסירים את הסטודנט מהקבוצה: הסרה שלו מהעץ דרגות בינארי מאוזן: O(logk) וחיפוש מחדש של הסטודנט המקסימלי בקבוצה: O(logk)

מסירים את הסטודנט מהhash table: O(1) משוערך בממוצע על הקלט.

בסך הכל: O(log*n + logk) משוערך בממוצע על הקלט

<u>JoinTeams(Team1, Team2):</u>

מוצאים כל קבוצה בUnion Find: 2O(log*n) משוערך

יש לבצע איחוד של שני העצי סטודנטים של הקבוצות: נעשה ע"י יצירת עץ ריק בגודל המתאים O(k1+k2), העברה של כל עץ למערך באמצעות סיור inorder O(k1) + O(k2) (הדבר יוצר 2 מערכים ממיונים לפי המיון של העץ – מתוכונות סיור inorder) ואז merge בין 2 מערכים לתוך העץ O(k1+k2) (מותר משום שאלה 2 מערכים ממוינים, מכניסים את האיברים לעץ הריק באמצעות סיור inorder עליו). את העץ הנ"ל נשים בקבוצת האב של הקבוצה המאוחדת. נשים לב כי המערכים הנוספים שמוקצים כאן אינם משפיעים על סיבוכיות הזיכרון הכללית משום שכל אחד מהם הוא O(k)

איחוד של 2 קבוצות בUnion find: O(1)

חיפוש הסטודט המקסימלי בקבוצה המאוחדת: חיפוש בעץ דרגות בינארי מאוזן (O(log(k1+k2))

סה"כ: O(log*n + k1 + k2) משוערך

<u>TeamFight(Team1,Team2, numOfFighters):</u>

מוצאים כל קבוצה בUnion Find: 2O(log*n) משוערך

חיפוש של האיבר בעל דרגה numOfFighters בכל עץ: O(logk1) + O(logk2), באיבר בדרגה המתאימה שמור גם סכום הכוחות של האנשים מעליו בסדר.

משווים בין סכומי הכוחות של 2 הצוותים: O(1)

מעדכנים את מונה הניצחונות של הקבוצה המנצחת: O(1)

סה"כ O(log*n + logk) משוערך (כאשר k הוא המקסימלי בין k1 לk2).

<u>GetNumOfWins(Team):</u>

מוצאים את הקבוצה בUnion Find: O(log*n) משוערך

מחזירים את המונה נצחונות הפנימי של הקבוצה: O(1)

סה"כ: O(log*n) משוערך

<u>GetStudentTeamLeader(ID):</u>

מוצאים את הסטודנט בhash table: O(1) משוערך בממוצע על הקלט.

משתמשים באינדקס הפינימי של הסטונדט כדי למצוא את הקבוצה שלו בUnion Find: O(log*n) משוערך

מחזירים את הסטונדט ששמור בקבוצה בתור הסטודנט החזק ביותר: O(1)

סה"כ: O(log*n) משוערך בממוצע על הקלט.

Quit():

משחררים את כל הקבוצות בUnion Find: O(n+k) (עוברים על הקבוצות אחת אחת ומשחררים אותן, כאשר בשחרור של כל קבוצה יש לשחרר גם את העץ הפנימי שלה. נשים לב כי כל סטודנט מופיע בעצי הקבוצות בעץ יחיד, לכן סכום גדלי העצים של הקבוצות שווה למספר הסטודנט במערכת ולכן זמן השחרור של כל העצים יחד הוא k)

משחררים את הUnion Find: שחרור של 3 מערכים 3O(1)

משחררים את הhash table: שחרור של מערך דינאמי בגודל k כאשר בכל תא משחררים את הסטונדט שנמצא בו: O(k)

סה"כ O(n+k).

בונוס:

על מנת לבצע את AddStudent וRemoveStudent ב(O(log*n + logk משוערך ולא בממוצע על הקלט, נרצה שההכנסה\הוצאה מהhash table יעשה באופן משוערך ולא בממוצע על הקלט.

לצורך כך נשנה את שיטת האיכסון בכל תא, נשתמש בchain hashing אבל במקום שיהיה באמצעות רשימות מקושרות נעשה זאת ע"י עצים מאוזנים בכל תא.

אם כלל הסטונדטים נופלים באותו תא (המקרה הכי גרוע), גובה העץ יהיה logk ולכן הכנסה\הוצאה מהעץ יעשה בlogk. בנוסף נעשה זאת במערך בגודל קבוע ולכן כלל הפעולות של הhash table ייקחו O(logk).

באופן כללי נקבל פעולה של AddStudent וRemoveStudent ב(O(log*n + logk משוערך.