

חלק יבש מבני נתונים רטוב 2

תיאור מבנה הנתונים

מבני הנתונים מורכב מ:

1. מבנה Union Find על התקליטים בשם records. הקבוצות במבנה הן הערמות, והתקליטים במבנה הם האיברים. התקליטים והקבוצות שמורים במערכים בגודל n, בדומה להסבר בהרצאה על מימוש Union Find באמצעות מערכים. האינדקס של תקליט ושל קבוצה הם המיקום שלהם במערכים. כול תקליט מכיל:

- שדה extra המתכן בעת איחוד וחיפוש, כך שסכימה על ערכי extra של כול התקליטים בחיפוש הקבוצה של תקליט שהמזהה שלו הוא r_id תחזיר את גובה התקליט.
- שדה buys בו מספר הרכישות של התקליט
- שדה parent בו נשמר האינדקס במערך התקליטים של התקליט הבא. אם הוא מספר שלילי, התקליט מצביע לקבוצה שהמזהה שלה הוא r_id.

כול קבוצה מכילה:

- שדה size המכיל את גודל הקבוצה, כמו בUnion Find רגיל.
- שדה column המכיל את עמודת הקבוצה.
- שדה height שהוא מספר העותקים של כול התקליטים בקבוצה ביחד.

2. club_members - עץ AVL שהוא עץ דרגות של חברי המועדון. המפתח הוא המזהה של חבר המועדון, והדרגה היא שדה rank_ המועדון כך שסכימה עליו לאורך מסלול החיפוש של חבר מועדון תיתן את הוצאותיו החודשיות. תקינות השדה תוכנן באופן רקורסיבי – נראה כי ניתן לשמור על תכונות השדה במתודות הרלוונטיות בהנחה שהתקיימו קודם לכן.

3. customers - טבלת האש דינאמית של לקוחות, בה פתרון התנגשויות נעשה באמצעות עצי AVL. המפתחות של העצים הם המזהה של הלקוחות. כול לקוח מכיל:

- phone – מספר הטלפון של הלקוח
- member – האם הלקוח חבר מועדון.

תיאור אלגוריתמים :

לאורך כול התיאור, n הוא מספר הלקוחות במבנה בעוד שm הוא מספר התקליטים, כמו בתיאור התרגיל.

אתחול המבנה – RecordsCompany()

באתחול המבנה, המערכים שבתוך records, customers הם מצביעים שעוד לא נעשה להם הקצאת זיכרון. לכן האתחול נעשה ב $O(1)$, כמו האתחול של עץ AVL של חברי המועדון מפני ש עצים מאותחלים ב $O(1)$. זאת, כפי שראינו בתרגול ובהרצאה. לכן, בסה"כ, הסיבוכיות הכוללת היא $O(1)$.

הפונקציה - `newMonth(int * records_stocks, int number_of_records)`

נאתחל את המערך של התקליטים ושל הקבוצות ב-Union Find למערכים בגודל `number_of_records`. המזהה של התקליט והקבוצות הוא האינדקס שלהם במערך. הגבהים ההתחלתיים של הקבוצות מאותחלים לפי הגבהים ב-`records_stocks`, והגודל שלהם מאותחל ל-0, והעמודה לאינדקס שלהם במערך. ההורה של המשתמשים מאותחל לקבוצה בעלת המזהה שלהם (מספר שלילי) והרכישות שלהם ל-0. ניתן לעשות זאת ב- $O(1)$ לפי אתחול מערכים, אבל עשיתי זאת ב- $O(m)$ בשל הצורך לטפל בהקצאות זיכרון, בגלל פשטות מימוש ובגלל שלא רציתי לסמוך על כך שהערכים במערך `records_stocks` קבועים במהלך ההרצה. נחזיר ערכי שגיאה עם נכשלנו בהקצאת זיכרון או שהקלט לא תקין כמתואר בדרישות התרגיל.

נעבור ב- $O(n)$ על כול האיברים בעץ של הלקוחות כמו באחד הסיורים שהוכח שהם מסיבוכיות גודל העץ. בכול איבר נאפס את הדרגה.

בסה"כ, הסיבוכיות של הפונקציה היא $O(m + n)$.

הפונקציה - `getExpenses(int c_id)`

- נלך לאורך מסלול החיפוש של `c_id` בעץ `club_members` ונסכום את השדה `rank` בכול הצמתים לאורך מסלול החיפוש.
 - נחזיר ערכי שגיאה עם `c_id` שלילי או לא קיים בעץ כמפורט בדרישות התרגיל.
- מסלול החיפוש בעץ `avl` הוא $O(\log(\text{size}))$ כש-`size` גודל העץ. מפני שאנחנו הולכים לאורך המסלול ומבצעים בכול צומת מספר סופי של פעולות בסיבוכיות $O(1)$, נקבל כי הסיבוכיות הכוללת היא $O(\log(n))$ מפני שגודל העץ `club_members` קטן מ- n .

הפונקציה - `addPrize(int c_id1, int c_id2, double amount)`

לעץ יש פונקציה עזר `add_rank(int c_id, double amount)`. בפונקציה זו, נוסיף `amount` עבור ההוצאות של כול חברי המועדון שהמזהה שלהם קטן מ-`c_id`. נעשה זאת באופן הבא:

- נלך לאורך מסלול החיפוש של `c_id`.
- בכול פעם שנתחיל רצף פניות ימינה, נוסיף `amount` לדרגת הצומת הראשון ממנו פנינו ימינה.
- בסיום רצף פניות ימינה, נוסיף לדרגת הצומת הראשון ממנו פנינו שמאלה `-amount`.
- אם מצאנו את `c_id` ואנחנו בתוך רצף פניות ימינה, נפחית מדרגת הצומת `amount`.
- נוסיף `amount` לבן השמאלי של `c_id` עם קיים.

נכונות:

נראה נכונות עבור צמתים שלא בתת העץ של `c_id`. עם ניסכום את `rank` לאורך מסלול החיפוש של צומת מחוץ לתת העץ של `c_id`, אם המזהה של הצומת קטן מזה של `c_id`, צומת זו תתאחד עם מסלול החיפוש של `c_id` עם צומת שממנה פנינו ימינה – כלומר באמצע רצף של פניות ימינה. לכן, ההוצאות – הסכום של `_rank` לאורך מסלול החיפוש יגדלו ב-`amount`. לעומת זאת, אם המזהה של הצומת גדול מזה של `c_id`, צומת זו תתאחד עם מסלול החיפוש של `c_id` עם צומת שממנה פנינו שמאלה – לאחר שנסגרו הרצפים של הפניות ימינה. לכן, ההוצאות – כלומר הסכום של `_rank` לאורך מסלול החיפוש לא השתנו.

עבור תת העץ של c_id , הוספנו amount לדרגת בן השמאלי ודאגנו לאיפוס התוספות לדרגה לאורך מסלול החיפוש בשלב השלישי של האלגוריתם. לכן, הסכום של $_rank$ לאורך מסלול החיפוש יגדלו amount רק עבור איברים בתת העץ השמאלי של c_id כנדרש.

סיבוכיות:

ביצענו פעולות בסיבוכיות $O(1)$ לאורך מסלול החיפוש של c_id . לכן, לפי תכונות עץ avl מההרצאות הסיבוכיות היא $O(\log(n))$ כמו שהראינו במתודה הקודמת.

כעת עבור המתודה `addPrize` נבצע פשוט:

1. `add_rank(c_id2, -amount)`

2. `add_rank(c_id1, amount)`

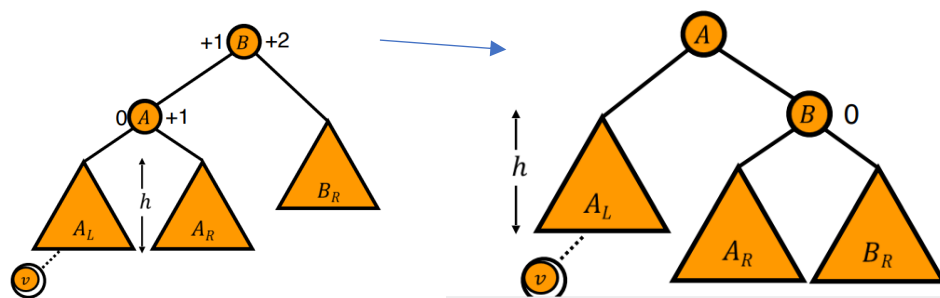
אם לא החזרנו סטטוס שגיאה על קלט לא תקין.

הפונקציה - `getPhone(int c_id)`

נחפש את הלקוח בטבלת ההאש ונחזיר את מספר הטלפון שלו. לפי משפט מההרצאה, הסיבוכיות היא $O(1)$ בממוצע על הקלט. נחזיר ערכי שגיאה כמצופה בתיאור הפונקציה.

הפונקציה - `makeMember(int c_id)`

נשתמש בפונקציה `isMember` כדי לבדוק האם הלקוח במערכת והאם הוא חבר מועדון. נחזיר סטטוס שגיאה אם צריך כנדרש. אם לא – נהפוך אותו בטבלת ההאש לחבר מועדון. סיבוכיות המקרה הגרוע של `isMember` היא $\log(n)$. אם הלקוח במערכת, נוסיף אותו לעץ השגיאה כמו בהוספה בעץ `avl`. נראה כיצד נשמור על הדרגה ללא לפגוע בסיבוכיות. בעת ההוספה, נסכום את הדרגות במסלול ההוספה. דרגת הצומת החדשה היא מינוס הסכום. כך, שנסכום את הדרגה במסלול החיפוש של הצומת החדש נקבל 0. בגלגולים במעלה העץ נעדכן את הדרגות כך שהתכונה של השדה `rank` בתת של השורש המוחזר מהגילגול תישמר. לדוגמא בגילגול LL:



נחשב את השדה `_rank` באופן הבא:

$A_R \rightarrow _rank += A \rightarrow _rank$ -
 $A \rightarrow _rank += B \rightarrow _rank$ -
 $B \rightarrow _rank -= A \rightarrow _rank$ -

ניתן לראות שלאחר ביצוע פעולות אלו סכום ערכי הדרגה במסלולים המובילים מ B והבנים שלהם עד לשורש תת העץ נשארים אותו גבר. לכן, תכונות השדה `_rank` נשמרות בגלגולים.

מפני שעדיין ביצענו פעולות שהן $O(1)$ סה"כ בכול צומת בה עבדנו ובגילגולים, הסיבוכיות היא $O(\log(n))$ כסיבוכיות להכנסה לעץ avl שנלמדה בהרצאה.

הפונקציה - `isMember(int c_id)`

נחפש את הלקוח בטבלת ההאש ונחזיר האם הוא חבר מועדון. נחזיר סטטוס שגיאה עם נדרש כמפורט בדרישות התרגיל. לפי משפט מההרצאה, הסיבוכיות היא $O(1)$ בממוצע על הקלט. נחזיר ערכי שגיאה כמצופה בתיאור הפונקציה. מפני שבטבלת ההאש פתרנו התנגשויות על ידי עץ avl, נקבל כי סיבוכיות המקרה הגרוע של פונקציה זו היא $O(\log(n))$ כי ראינו בהרצאה שסיבוכיות חיפוש בעץ avl היא מסדר גודל של גובה העץ.

הפונקציה - `buyRecord(int c_id, int r_id)`

נחזיר הודעת שגיאה עם הקלט לא תקין, או שהמזהים לא קיימים כנדרש. נבדוק האם המשתמש קיים ואם הוא חבר מועדון לפי הפונקציה `isMember` שסיבוכיות המקרה הגרוע שלה היא $\log(n)$. ניקח ב- $O(1)$ את מספר הרכישות של התקליט `r_id` ממערך התקליטים – נסמנו ב-`t`.

נממש באמצאות הפונקציה `rank_add` שהוצגה בתיאור `add_prize` באופן הבא:

1. `add_rank(c_id + 1, t + 100)`

2. `add_rank(c_id, - t - 100)`

הוספנו את הסכום הדרוש לדרגה.

מכיוון שסיבוכיות `add_rank()` היא $\log(n)$, כפי שראינו סה"כ עמדנו בסיבוכיות $\log(n)$.

הפונקציה - `addCostumer(int c_id, int phone)`

נבצע האש ל-`c_id`. פתרנו התנגשויות באמצעות עץ avl – כלומר, בכול תא במערך, קיים עץ avl אליו נוסיף את הלקוח. השתמשנו בפונקציית ערבול רגילה – `%size`, כש-`size` גודל הטבלה. אם נגמר המקום, כלומר – מספר הלקוחות לפני ההוספה כגודל הטבלה, נגדיל פי שניים את הטבלה כמו שלמדנו בשיעור על מערך דינאמי. נשים בה את כול האיברים תוך ביצוע hash מחודש. נחזיר סטטוס שגיאה עם צריך לפי דרישות התרגיל. לפי ההרצאה, הסיבוכיות המשוערכת של טבלת hash דינאמית היא $O(1)$ בממוצע על הקלט עם השתמשנו בפונקציית ערבול רגילה. לכן, עמדנו בסיבוכיות הנדרשת.

הפונקציה - `putOnTop(int r_id1, int r_id2)`

נחפש את הקבוצות של `r_id1` ו-`r_id2` כמו בפעולת `find` ב-`UnionFind` הנלמדה בכיתה. פעולת `find` תפורט בתיאור `getPlace`. נסמן מזהים של קבוצות `A` ו-`B` בהתאמה. הקבוצות `A` ו-`B` והתקליטים המצביעים אליהם במקומות אם אינדקסים אלו במערכים של הקבוצות והתקליטים. נאחד קבוצות אלו ונעדכן את הגבהים והגודל בהתאם. נעדכן את ה-`culomnn` של הקבוצה המאוחדת להיות `culomnn` של `B`. נחלק למקרים:

אם הגודל של `A` גדול מהגודל של `B`:

- נוסיף ל-`extra` של התקליט `A` את הגובה של הקבוצה `B`.
- נחסיר מה-`extra` של התקליט `B` את ה-`extra` המעודכן של התקליט `A`.

במקרה זה נקבל שמפני שהאיחוד מתבצע על ידי כך שההורה של התקליט B הוא התקליט A, נקבל כי הסכימה על השדה extra לאורך מסלול חיפוש הקבוצה תגדל בגובה של B עבור תקליטים בקבוצה A, ולא תשתנה עבור תקליטים בקבוצה B.

אם הגודל של B גדול מהגודל של A:

- נוסיף extra של התקליט A את הגובה של הקבוצה B פחות extra של התקליט B.

במקרה זה נקבל שמפני שהאיחוד מתבצע על ידי כך שההורה של התקליט A הוא התקליט B, הסכימה על השדה extra לאורך מסלול חיפוש הקבוצה תגדל בגובה של B עבור תקליטים בקבוצה A. עבור תקליטים בקבוצה B הסכימה לא תשתנה.

מכיוון שביצענו פעולות Find Union, בשינויים שלוקחים $O(1)$ בסי"כ, לפי מה שהוכח בכיתה נקבל כי הסיבוכיות המשוערכת היא $O(\log^* m)$ כנדרש.

ראינו שתקינות השדה extra נשמרת בפעולה.

נחזיר סטטוסים של שגיאה עבור קלט לא תקין כפי שנתבקשנו בתרגיל.

הפונקציה - `getPlace(int r_id, int *column, int *hight)`

נבצע פעולת Find של Union Find כפי שנלמד בכיתה. השינויים בפעולה נעשה הם הבאים:

- בשלב מציאת הקבוצה של התקליט בעל `r_id` – ניסכום את השדה extra במסלול חיפוש הקבוצה. לא נבצע כיווץ.
- נשמור במשתנה נוסף את הסכום הקודם פחות האסטרך של הקבוצה (מיוצגת במקרה זה על ידי התקליט שמצביע אליה). בעת הכיוון של כול תקליט במסלול החיפוש, נוסיף את המשתנה הנ"ל extra של התקליט. לאחר מכן, ונפחית ממנו את extra הישן התקליט. הכיווץ נעשה על ידי כך שהparent הוא התקליט שמצביע לקבוצה.

הסיבוכיות של Find לא נפגעת, כי בכול שלב במסלול החיפוש של הקבוצה ובכיווץ השינויים שביצענו הם $O(1)$.

התקינות של השדה extra נשמרה במהלך הכיווץ, מפני שהוספנו לכול תקליט שכיווץנו את את סכום extra של התקליטים לפניו במסלול חיפוש הקבוצה – מפני שהם כבר לא נמצאים במסלול החיפוש שלו.

כעת עבור המתודה, נחזיר את העמודה של הקבוצה ובתור הגובה את הסכימה של שדה אקסטרה. מפני שהפונקציות `getPlace` ו-`putOnTop` ביצענו פעולות Find Union, נקבל כי הסיבוכיות המשוערכת שלהן היא $\log^* m$.

נחזיר סטטוס שגיאה עם צריך כמפורט בדרישות התרגיל.

ההורס - `~RecordsCompany()`

במהלך ההורס, נשחרר את המערכים שהקצינו עבור Union Find וטבלת ההאש – עם שיחרור כול העצים בטבלה. לאחר מכן, נשחרר את העץ של חברי המועדון. נקבל כי הסיבוכיות הכוללת של פעולות אלו הינה גודל עץ חברי המועדון וסכום גדלי העצים בטבלת ההאש, כלומר $O(n)$ שזה גם $O(n+m)$. ניתן להרוס עץ בינארי על ידי חיפוש postorder – הורסים את הבנים ואז את השורש. הסיבוכיות של חיפוש זה לפי ההרצאה היא $O(\text{size})$ כשsize גודל העץ.