

קורס: מבני נתונים 1 - 234218

חלק יבש לרטוב 2

מגישות:

1. אמילי דרור [324934082]
2. עדי צח [207844358]

תאריך הגשה: 18.06.2023

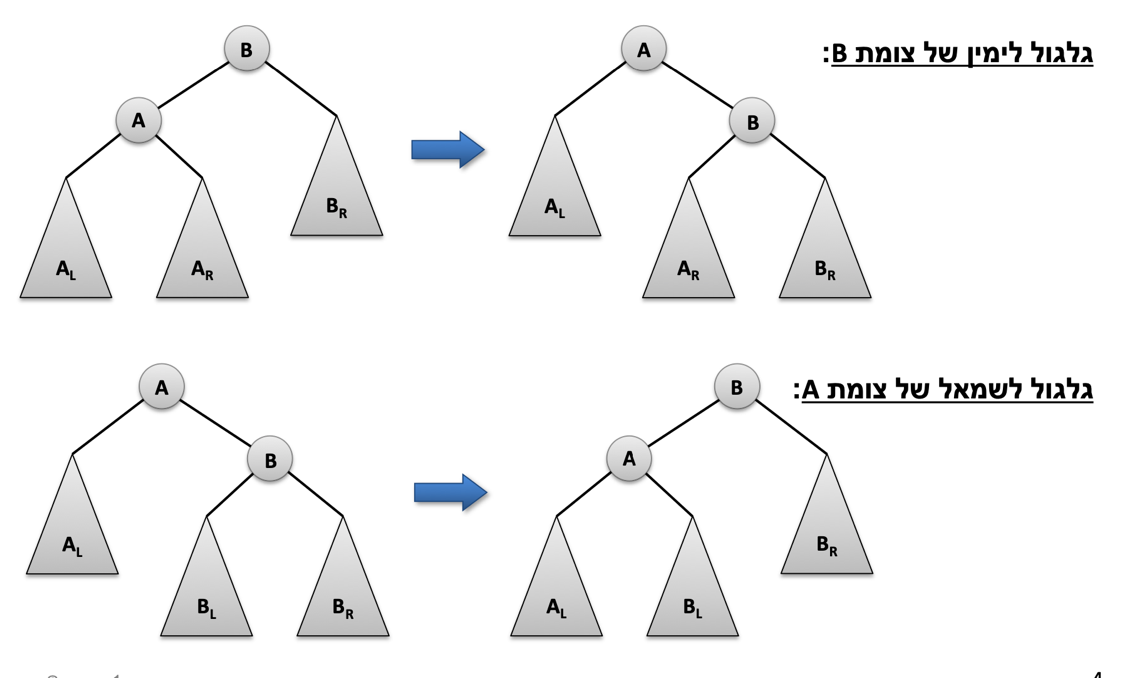
**תיאור מבנה הניתונים**

מחלקת RecordsCompany

1. טבלת ערבול [customers\_id] – טבלת ערבול מסוג Chain Hashing אלא שבמקום רשימות מקושרות תחזיק עצי AVL אשר יכילו מצביעים ללקוחות ע״פ המזהי ID שלהם. כלומר כמות האיברים המוחזקים בטבלה מקיימת O(n). כפי שלמדנו, סיבוכיות ההכנסה לhash היא O(1) בממוצע על הקלט משוערך: הhash עובדת עם פונקציית ערבול: מודלו על גודל הטבלה. הטבלה דינמית וגודל הטבלה פרופורציונלי למספר האיברים בה (כפי שנלמד בתרגול) לכן פקטור העומס יהיה O(1) וכן מאחר שהכנסה של איבר יכולה לגרום להגדלתה של הטבלה (ממש כפי שנלמד בתרגול על מערך דינמי) הסיבוכיות הינה גם משוערכת. נציין כי חיפוש והכנסה בhash הם בסיבוכיות O(log(n) כי במקרה הגרוע ביותר כלל n הלקוחות ימוקמו באותו עץ ה-avl בטבלה.
2. עץ AVL עם שדה אקסטרה [prizes] – עץ avl המחזיק צמתים הממויינים לפי id (של לקוחות שהינם חברי מועדון). כלומר כמות האיברים המוחזקים בעץ מקיימת O(n). כמו כן, נתחזק שדה בכל צומת באופן כזה כך שביצוע חיפוש בעץ הנ״ל עבור id נתון וסכמת שדות האקטרה בכל צומת בדרך יניב את הפרס לו הלקוח בעל ה-id הנתון זכאי. ההסבר אודות אופן תחזוקת השדה אקסטרה הוצג לנו ע״י עידו גליל בפונקציית ה-add בתרגול שלו, בחרנו לממש פונקציה זהה ע״פ אותם עקרונות:

* בפניה ראשונה ימינה נגדיל את ערך האקטרה בצומת ממנו אנחנו פונים ב-amount.
* בפניה ימינה אחרי פניה ימינה נמשיך ללא שינויי בשדה .
* בפניה שמאלה אחרי פניה ימינה , נקטין את ערך האקסטרה ממנו ירדנו ב-amount.
* בפניה שמאלה אחרי פניה שמאלה, נמשיך בחיפוש ללא שינויי בשדה .
* בפניה ימינה אחרי פניה שמאלה נגדיל את ערכו של הצומת ממנו אנחנו פונים ימינה ב-amount.
* במידה והגענו לצומת העניין דרך פנייה שמאלה נגדיל את שדה האקסטרה של הצומת ב-amount, ובמידה וקיים לצומת בן ימני נקטין את שדה האקטסרה שלו ב-amount ונסיים את החיפוש.
* במידה והגענו לצומת העניין אחרי פנייה ימינה לא נגדיל את ערכו , אך נקטין את ערך שדה האקטסרה של של בנו הימני ב-amount ונסיים את החיפוש.

מדובר בתנועה בעץ avl וביצוע פעולות ב-O(1) ועל כן מדובר בפעולה בסיבוכיות של O(log(n)).

בעת ביצוע גלגולים יש לעדכן את שדה האקסטרה באופן הבא:

גלגול LL

את A בציור נעדכן : A\_after→extra=A\_before→extra+B\_before→extra

את B בציור נעדכן : B\_after→extra=-A\_before→extra

את השורש של Ar נעדכן : A\_r→extra +=A\_before→extra

גלגול RR

את A בציור נעדכן : A\_after→extra=A\_before→extra+B\_before→extra

את B בציור נעדכן : B\_after→extra= -A\_before→extra

את השורש של Al נעדכן : A\_l→extra+=A\_before→extra

1. UnionFind [records] – המבנה יחזיק את כלל התקליטים במערכת. נממש את המבנה בעזרת עצים הפוכים עם איחוד לפי גודל וכיווץ מסלולים (נממש את זה כפי שראינו ברצאה בעזרת מערך המתייחס להורים של כל אובייקט, מערך המתייחס לגדול הקבוצה של כל צומת, כמו כן נחזיק מערך שמתאר את גובה ״ערמות״ התקליטים ומערך שיחזיק את התקליטים עצמם). נדאג לתחזק את פרמטר הגובה של כל תקליט דרך המבנה הזה בהתאם לשאלת הארגזים שהוצגה לנו ע״י עידו גליל בתרגול UnionFind. מבחינת כמות האיברים המוחזקים במבנה מתקיים O(m).

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

מחלקת Costumer

שדות:

* id - מזהה ייחודי [int]
* clubMember - שדה בוליאני שמציין אם המשתמש הינה חבר מועדון. [bool]
* phone – מספר הטלפון של המשתמש [int]
* monthlyPatments – חובות חודשיים למשמתמש במידה והנ״ל הינו חבר מועדון. [double]

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

מחלקת Record

שדות:

* id - מזהה ייחודי [int]
* copies - שדה שמציין את מספר העותקים של התקליט [int]
* buys – כמות הפעמים שהתקליט נקנה [int]
* stackHeight – פרמטר בו אנו נעזרים לחישוב גובה התקליט. [int]
* column – שדה בו נעזר על מנת למצוא את העמודה הרלוונטית לתקליט [int]

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

**­­­**

**תיאור המתודות של RecordsCompany:**

RecordsCompany()

*תיאור:* . נאתחל את כלל המבנים הקיימים במערכת (עץ דרגות, UnionFind, HashTable) שלנו עם ערכי null.

*סיבוכיות: .* O(1)

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

~RecordsCompany()

*תיאור:* נמחק את המידע המוחזק בעץ הפרסים, ב-UnionFind וב-HashTable.

*סיבוכיות:* O(n+m)

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

StatusType newMonth(int \*records\_stocks, int number\_of\_records)

*תיאור: נבדוק מקרי קצה ובמידה והקלט תקין נמחק את מבנה ה-unionfind הקיים וכך נאפס את סידור התקליטים בסיבוכיות O(m). נייצר את מבנה unionfind חדש ונמלא אותו במידע של התקליטים החדשים (אותם ניצור כל אחד בסיבוכיות של O(1) ) סה״כ יבוצע בסיבוכיות O(m). נרוץ על כלל הלקוחות הקיימים במערכת ונאפס את החובות החודשיים שלהם באופן לינארי, בסיבוכיות O(n). נמחק את עץ הפרסים בסיבוכיות O(n).*

*סיבוכיות.*

*סיבוכיות:* O(n+m)

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

StatusType addCostumer(int c\_id, int phone)

*תיאור: נבדוק ובמידה והקלט תקין ננסה להוסיף את הלקוח הנוסף למערכת ע״י צירופו ל-HashTable* O(1) בשיערוך בממוצע על הקלט*, נבחן מקרי בקצה ובמקרה והקלט תקין נייצר אובייקט חדש מסוג costumer* O(1)*.*

*סיבוכיות: O(1) משוערך בממוצע על הקלט*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

Output\_t<int> getPhone(int c\_id)

*תיאור: נבחן מקרה קצת וננסה לגשת ללקוח בעל המזהה המתקבל בעזרת טבלת הערבול שלנו ב-O(1) בממוצע על הקלט, נחזיר את מספר הטלפון של האובייקט.*

*סיבוכיות: O(1) בממוצע על הקלט*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

StatusType makeMember(int c\_id)

*תיאור: נבחן מקרי קצה וננסה לגשת ללקוח בעל המזהה המתקבל, נעשה זאת בעזרת טבלת הערבול שלנו בסיבוכיות O(log(n)) זאת כיוון שטבלת הערבול שלנו ממומשת עם עצי avl ולא רשימות מקושרות (במקרה הגרוע ביותר כלל n הלקוחות ימוקמו באותו עץ ה-avl ועל כל הסבוכיות הינה O(log(n))) במידה והלקוח הינו כבר חבר מועדון נחזיר את הפלט המתאים, אחרת נהפוך את הלקוח לחבר מועדון ע״י שינוי השדה הרלוונטי אצלו O(1) כמו כן, נוסיף לעץ הפרסים צומת המתייחס ל-id של הלקוח הנתון O(log(n). נאפיין את כמות הפרס אליה ה-id של הלקוח הנתון זכאי לפי עץ הפרסים ונחסיר כמות זו משדה החוב החודשי של הלקוח בסיבוכיות O(log(n).*

*סיבוכיות: O(log(n))*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

Output\_t<bool> isMember(int c\_id)

*תיאור: נבדוק מקרה קצת וננסה לגשת ללקוח בעל המזהה המתקבל בעזרת טבלת הערבול שלנו בסיבוכיות O(1) בממוצע על הקלט, נחזיר את שדה חבר-מועדון של הלקוח O(1).*

*סיבוכיות: O(1) בממוצע על הקלט*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

StatusType buyRecord(int c\_id, int r\_id)

*תיאור: נעזר במבנה ה-UnionFind בשביל למצוא את התקליט הרלוונטי סיבוכיות O(log\*(m)) (בגלל* איחוד לפי גודל וכיווץ מסלולים)*. נעזר בhashtable על מנת לגשת ללקוח הרלוונטי בסיבוכיות O(log(n)) זאת כיוון שטבלת הערבול שלנו ממומשת עם עצי avl ולא רשימות מקושרות (במקרה הגרוע ביותר כלל n הלקוחות ימוקמו באותו עץ ה-avl ועל כל הסבוכיות הינה O(log(n))). כעת במידה והלקוח הינו חבר מועדון אזי נבחן את כמות הקניות שבוצעו לתקליט לפני פועלה זו ונוסיף לתשלום החודשי של הלקוח את סכום הקניות הנ״ל + 100, לאחר מכן נוסיף 1 לכמות הקניות שבוצעו לתקליט, סיבוכיותO(1) . אם הלקוח אינו חבר מועדון אזי רק נוסיף 1 לכמות הקניות שבוצעו לתקליט בסיבוכיותO(1) .*

*סיבוכיות: סה״כ O(log(n)*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

StatusType addPrize(int c\_id1, int c\_id2, double amount)

*תיאור: ננסה להכניס לעץ הדרגות (אשר מתייחס לפרסים אשר כל id זכאי להם) 4 צמתים המתייחסים לגבולות ה-id שהתקלבו כפרמטרים:(c\_id1-1,c\_id1,c\_id2-1,c\_id2) כמובן שבמידה וקיים כבר צומת עם 1 מערכי ה-id הנ״ל אזי לא נוסיף את הצומת הרלוונטי, מדובר ב-4 פעולות בסיבוכיות O(log(n) (עץ הפרסים מחזיק בתוכו באופן קבוע רק צמתים המיוחסים ל-id-ים השייכים ללקוחות שהפכו לחברי מועדון, את 4 הצמתים שהכנסנו כעת אנו עומדים להוציא בסוף הפונקציה הזו). כעת נפעיל את פונקציית add, מדובר בפונקצייה אשר עקרון הפעולה והנכונות שלה הוסברו ע״י עידו גליל בזמן האקסטרה של התרגול בנושא עצי avl, היא פועלת בסיבוכיות O(log(n). נפעיל את הפונקציה הנ״ל פעמיים add(c\_id2-1,amount) ו- add(c\_id1-1,amount)וכך נדאג שתוספת הפרס שהעץ קיבל הינה רלוונטית באמת אך ורק לתחום 𝑖𝑑2\_𝑐 < 𝑑𝑖\_𝑐 ≤ 𝑖𝑑1\_𝑐 כפי שנדרש.*

*ולבסוף נמחק את ארבעת הצמתים שהכנסנו בתחילת הפונקציה O(log(n)).*

*סיבוכיות: סה״כ O(log(n))).*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

Output\_t<double> getExpenses(int c\_id)

*תיאור: נבחן מקרה קצת וננסה בעזרת ה -hashtable לגשת ללקוח הרלוונטי בסיבוכיות O(log(n)) זאת כיוון שטבלת הערבול שלנו ממומשת עם עצי avl ולא רשימות מקושרות (במקרה הגרוע ביותר כלל n הלקוחות ימוקמו באותו עץ ה-avl ועל כל הסבוכיות הינה O(log(n))). לאחר מציאת הלקוח נבחן את שדה התשלום-החודשי שלו ונחסיר ממנו את הפרס לו הלקוח זכאי לפי עץ הפרסים. נחשב את כמות הפרס לו זכאי הלקוח ע״י חיפוש id של הלקוח הנתון בעץ וסכמת שדות האקסטרה בכל צומת במסלול בסיבוכיות O(log(n)).*

*סיבוכיות: סה״כ O(log(n))*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

StatusType putOnTop(int r\_id1, int r\_id2)

*תיאור: נבחן את הקלט ובמידה ותקין נחפש בעזרת מבנה ה-unionfind את השורשים של שני המזהים שהתקבלו בסיבוכיות . כעת במידה ומדובר באותו האובייקט אזי נחזיר failure. אחרת מדובר בשני עצים הפוכים שונים ויש לאחד ביניהם, נשווה בין גדלי העצים הנ״ל, במידה והעץ של התקליט r\_id1 הינו גדול יותר או שווה לעץ השני אזי נעדכן את גובה השורש של העץ של r\_id2 בפלוס הגודל המלא של עץ r\_id1 ונחסיר את פרמטר הגובה המוחזק בשורש של עץ r\_id1 בסיבוכיות O(1). אחרת מתקיים שהעץ של התקליט r\_id2 הינו גדול יותר מ r\_id1, במצב כזה נגדיר את ה-column של עץ 2 להיות ה-column של עץ 1 בסיבוכיות O(1). נוסיף לגובה שורש העץ 2 את הגובה המלא שמחוזק בעץ 1 בסיבוכיות O(1). נחסיר מגובה שורש עץ 1 את הפרמטר גובה של שורש עץ 2 בסיבוכיות O(1). כמו כן, בכל מקרה נדאג לשינויים הבאים: שדה כמות האיברים בעץ הגדול יותר תהיה כעת סכום כמות האיברים בשני העצים, האבא של העץ הקטן יהפוך להיות שורש העץ הגדול וגודל הערמה המקסימאלית יהיה כעת סכום הגבהים של שתי הקבוצות (הכל בסיבוכיות O(1)).*

*סיבוכיות:*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

StatusType getPlace(int r\_id, int \*column, int \*hight)

*תיאור: נבחן את הקלט ובמידה ותקין נבחן בעזרת מבנה ה-UnionFind את התקליט הרלוונטי, נסכום את גובה התקליט הזה ע״י ריצה בעץ ההפוך במבנה ה-unionfind וסכימת פרמטר הגובה המוחזק אצל כל האובייקטים בדרך סיבוכיות , נחסיר מהסכום המתקבל את כמות הcopies של התקליט ונשים את הערך המתקבל ב* hight *בסיבוכיות O(1). נעזר בפונקציית ה-find של ה-unionfind בשביל למצוא את ראש הקבוצה של התקליט המתקבל בסיבוכיות , ונשים ב-* columnאת שדה ה- column של ראש הקבוצה *בסיבוכיות O(1)*.

*סיבוכיות:*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*