

מטלת מנחה (ממ"ן) 18 - פרויקט 2**❖ תקציר מופשט****✚ מטרת פרויקט ותוצאותיו**

מטרת פרויקט זה היא לכתוב תכנית לניהול קבוצה דינאמית של חשבונות בנק. הניהול כולל את פעולות המילון הבסיסיות – הכנסה, מחיקה וחיפוש, וכן שאילתות נוספות כגון מציאת המקסימום, מציאת כל החשבונות שיתרתם שלילית, ושאילתת עדכון של הגדלת/הקטנת היתרה בחשבון. הדרישה העיקרית בתכנית היא בחירת מבנה נתונים יעיל ככל האפשר (כפונקציה של מספר החשבונות), שיאפשר מענה על השאילתות הללו בסיבוכיות זמן ריצה מינימאלית.

מבנה הנתונים שנבחר לצורך המימוש הינו עץ ערכי מיקום (עץ אדום-שחור מורחב). מבנה זה תומך בכל הפעולות הנדרשות בסיבוכיות זמן לוגריתמית (ביחס למספר החשבונות) במקרה הגרוע ביותר, ובחלקם אף בזמן קבוע.

❖ תיאור הבעיה והגישה הכללית של התוכנית לפתרון

הבעיה המרכזית הייתה לבחור מבנה נתונים מתאים ולממש אותו פיזית בתכנית מחשב הלכה-למעשה.

פיתרון: נחלק את הפיתרון ע"פ 4 השלבים המפורטים בסעיף 14.2:

1. **בחירת מבנה נתונים תשתיתי:** בשלב ראשון נלקחו בחשבון החלופות הבאות עבור בחירת מבנה נתונים התומך בפעולות הבסיסיות ביעילות המקסימאלית –

| שאלת/מבנה נתונים | ערימת מקסימום/מינימום | טבלת גיבוב | עץ-חיפוש בינארי | עץ אדום-שחור | עץ ערכי מיקום |
|-------------------------|---|--|--|---|---|
| הכנסה | $O(\lg n)$ | $O(1)$ תוחלת | $O(h)$ | $O(\lg n)$ | $O(\lg n)$ |
| מחיקה | $O(\lg n)$ | $O(1)$ תוחלת | $O(h)$ | $O(\lg n)$ | $O(\lg n)$ |
| חיפוש | $O(n)$ | $O(1)$ תוחלת | $O(h)$ | $O(\lg n)$ | $O(\lg n)$ |
| מקסימום | $\Theta(1)$ | $O(n)$ | $O(h)$ | $O(1)$ | $O(1)$ |
| הגדלת/הפחתת ערך | תמיכה רק באחת מהפעולות (הגדלה בערימת מקסימום והקטנה בערימת מינימום) $O(\lg n)$ | $O(1)$ ע"י מחיקה של המקסימום והכנסה של המינימום העדכני | $O(h)$ ע"י מחיקה של המקסימום והכנסה של המינימום העדכני | $O(\lg n)$ ע"י מחיקה של המקסימום והכנסה של המינימום העדכני | $O(\lg n)$ ע"י מחיקה של המקסימום והכנסה של המינימום העדכני |
| הערות – יתרונות/חסרונות | בניית הערימה עצמה $O(n)$ | הקבוצה דינאמית ואינה סטטית: המפתחות אינם ידועים מראש ומשתנים לאחר ההכנסה | במקרה הגרוע ביותר גובה העץ ליניארי למספר הצמתים $h = n - 1$ | מובטח שגובה העץ מאוזן ויהיה לוגריתמי לכל היותר | בנוסף לאיזון תומך בשאלות ערך מיקום |

- מבנה ערימה נפסל מאחר שאינו תומך בחיפוש ביעילות האופטימאלית, וכן פעולת ה-Increase/Decrease של ערך מפתח פועל באופן חד-כיווני, ואינו מאפשר הגדלת/הקטנת יתרה באופן גמיש.
 - מבנה טבלת גיבוב אמנם מציג ביצועים אופטימאליים, אך הם רק בתוחלת, ובמקרה הגרוע ביותר הסיבוכיות עשויה להיות ליניארית. כמו כן, נדרש תמיכה בקבוצה דינאמית, שאין לנו יודעים בהכרח את טווח המפתחות שלה, וייתכן והוא בלתי מוגבל, מה שאינו מאפשר הקצאת מקום "אינסופי" רציף עבור הטבלה.
 - עץ חיפוש-בינארי אמנם תומך במרבית השאלות הבסיסיות, אך אין זה מובטח שגובה העץ יהיה מאוזן ובמקרה הגרוע הפעולות יבוצעו בזמן ליניארי.
 - עץ אדום-שחור מבטיח את ביצוע הפעולות הבסיסיות בזמנים אופטימאליים במקרה הגרוע ביותר, אך עם קצת השקעה נוספת נוכח להרחיבו לעץ ערכי מיקום (Order-Statistic tree), שיתמוך בכל הפעולות של עץ אדום-שחור בסיסי, ובנוסף עליהן, גם בשאלות מציאת ערך מיקום, קטעים, עוקב, קודם, מקסימום ומינימום בזמן אופטימאלי, שאלות שיוכלו לסייע במתן מענה לשאלות כמעין מציאת כלל הלוקחות שיתרתם שלילית.
- מסקנה:** מבנה הנתונים התשתיתי שנבחר לפרויקט הוא עץ אדום-שחור מורחב - עץ ערכי מיקום.

2. קביעת המידע הנוסף שיש לתחזק במבנה הנתונים התשתיתי: מבנה הנתונים

התשתיתי הבסיסי שנבחר, תומך בארגון הרשומות ע"פ מפתח אחד בלבד (לצורך ביצוע השאילתות ביעילות עבור מפתח זה). בתכנית הנדרשת בפרויקט זה חלק מהשאילתות מצריכות חיפוש לפי מפתח של מספר לקוח/חשבון, כגון הדפסת היתרה של הלקוח שמספרו 123456, ואילו שאילתות אחרות, דורשות חיפוש ע"פ ערך מפתח אחר, של יתרה, כדי לענות על שאלות כגון מי הלקוח שיתרתו מקסימאלית, וכן מיהם כל הלקוחות שיתרתם שלילית. אי לכך – יש להרחיב את מבנה הנתונים כך שיתמוך ביעילות בשאילתות עבור 2 המפתחות – מספר לקוח ויתרה בחשבון.

אופן המימוש שהוחלט עליו בפיתרון זה הוא לנהל 2 עצים אדומים-שחורים: האחד יהיה עץ ערכי-מיקום, שהרשומות בו מאורגנות לפי מפתח של יתרה, ואילו העץ השני יהיה עץ אדום-שחור שהרשומות בו מאורגנות לפי מפתח של מספר לקוח. מטעמי חיסכון במקום (זיכרון), הרשומות בעץ המורחב מכילות את כל הנתונים הנלווים הדרושים לניהול – שם הלקוח, ת"ז, וכו', ואילו העץ הבסיסי "מנוון" ומכיל רק את המפתח של מספר הלקוח ללא נתונים נלווים נוספים, שכן כל מטרתו היא להוות אינדקס – גישה מהירה לרשומות ע"פ מפתח נוסף (מספר הלקוח).

איזה מידע נוסף יש לתחזק?

נדרש לתחזק את הקשר בין 2 העצים: העץ המורחב והעץ הבסיסי. קשר זה נעשה ע"י הוספת מאפיין נוסף בכל רשומה, שתצביע על הרשומה המקבילה לה בעץ המקביל. כך למשל, כאשר יתווסף לקוח חדש, מייד תבוצע הצבעה ממנו אל הרשומה המקבילה לו בעץ המקביל ולהפך, וכן לדוגמה כאשר נרצה לחפש לקוח ע"פ מספרו, תהיה לנו גישה ישירה לנתונים הנלווים המצויים בעץ המאורגן בכלל לפי יתרות בחשבון.

3. אימות היכולת לתחזק את המידע הנוסף: המידע הנוסף ניתן לחישוב/הסקה

מתוך צומת ו-2 בניו: ולכן ע"פ משפט 14.1, ניתן לתחזק מידע זה מבלי להשפיע על זמן הריצה האופטימאלי הקיים במבנה התשתיתי הבסיסי של $O(lgn)$.

4. פיתוח פעולות חדשות: מרבית הפעולות המבוקשות למימוש בפרויקט זה –

חיפוש, הכנסה, מחיקה ומציאת מקסימום כבר נתמכות ע"י מבנה הנתונים התשתיתי. ניתן אף לבצע אופטימיזציה עבור המקסימום מזמן לוגריתמי לזמן קבוע. הפעולות החדשות שיש לממן הן הוספת/הפחתת ערך מפתח היתרה והדפסת כל הרשומות של הלקוחות שיתרתם שלילית.

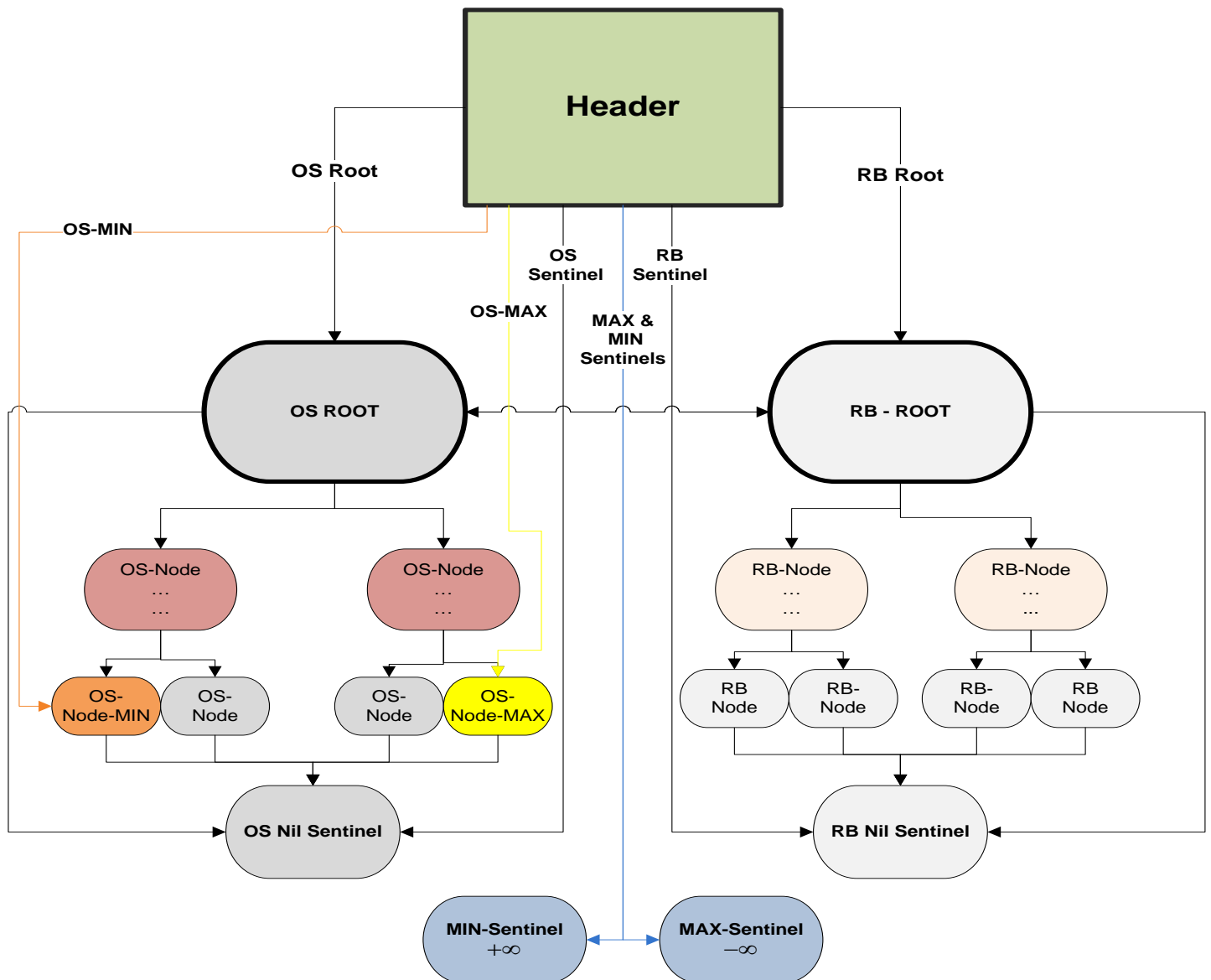
- פעולת הוספת/הפחתת ערך מפתח היתרה (הפקדה/משיכה), עשויה להפר את תכונת עץ החיפוש הבינארי, על כן יש לבצע תיקון בעץ. הטקטיקה שנעשתה למימוש פעולה זו היא בהשראת $HEAP-EXTRACT-MAX$, מבוצעת מחיקה של האיבר המקורי, והכנסה מחדש של האיבר עם ערך המפתח העדכני לאחר השינוי.

- פעולת הדפסת כל הלקוחות שיתרתם שלילית עשתה מחזור (reuse) בפעולה הקיימת של $RB-ENUMERATE$, שמוצאת את כל הרשומות שערך המפתח שלהן בטווח נתון. ניתן להשתמש בשגרה זו עם טווח ממניוס אינסוף ועד 0 (לא כולל), ולקבל את הפלט המבוקש.

❖ תיאור מבנה הנתונים העיקריים שבהם התכנית משתמשת:

1. עץ אדום-שחור מורחב – עץ ערכי מיקום המארגן רשומות מלאות של לקוחות ע"פ מפתח יתרת חשבון, עם מלוא הנתונים הנלווים והצבעה על רשומה מקבילה בעץ אדום-שחור בסיסי מקביל.
2. עץ אדום-שחור בסיסי המארגן רשומות לפי מספר לקוח שאינו מכילות נתונים נלווים נוספים למעט נתון זה ומצביע לרשומה המקבילה בעץ ערכי מיקום.
3. כותרת מבנה נתונים עץ, המכיל מצביעים לשורשי 2 העצים ומצביעים לצמתי הזקיף שלהם (sentinel), לצומת המקסימום הנוכחי, ומינימום נוכחי.

מצ"ב סקיצה כללית למבנה העיקרי – רשומת כותרת המכילה מצביעים לשורשים של שני העצים ולזקיפים שלהם, למקסימום ולמינימום. כל רשומה בעץ ערכי מיקום מכילה את כלל הנתונים הנלווים – ת"ז, מס' לקוח, שם, יתרה ומצביע לרשומה המקבילה בעץ א"ש, ובעץ א"ש כל רשומה מכילה רק את מס' הלקוח ומצביע לרשומה המקבילה בעץ ערכי מיקום:



עבור המקסימום והמינימום הוצבו 2 זקיפים נוספים בעלי ערכי מינימום ומקסימום אפשריים בהתאמה. באופן זה, באתחול ראשוני, המפתח הראשון שיוכנס בוודאי יהיה קטן מערך המקסימאלי האפשרי ולכן ייקבע למינימום, ובאופן דומה, כל ערך יהיה גדול או שווה לערך המינימאלי האפשרי ולכן יאותחל תחילה למקסימום. בעת הכנסה מצביעים אלו יעודכנו בהתאם לצורך.

❖ תיאור כללי של הפונקציות המרכיבות את התכונות והקשרים ביניהן:

בוצעה חלוקה של הפונקציות ל-4 מודולים עיקריים כדלהלן:

א. מודול עזרי קלט/פלט - "IO_UTILS": המודול אחראי על כלל הקלט/פלט ובכלל כך –

- קליטת קלט בדיאלוג מהמשתמש / רשימת קבצים.
- בדיקות תקינות על הקלט.
- הגדרת מבני נתונים המשותפים לכלל המודולים, והמרה ומיפוי של פרמטרי קלט לשדות מבני נתונים אלו.
- חיזוי על הודעות הצלחה/שגיאה.

ב. מודול עץ ערכי מיקום - "OS_TREE": המודול אחראי על מימוש שגרות עץ ערכי מיקום מורחב ובכלל כך –

- שגרות בסיסיות של עץ א"ש – הכנסה, חיפוש, מחיקה, מינימום, מקסימום, עוקב, קודם, סריקה תוכית, סיבובים, תיקונים לאיזון גובה העץ.
- שגרות בסיסיות של עץ ערכי מיקום – מציאת מיקומו היחסי של איבר שערך המפתח שלו נתון, ומציאת ערך איבר שמיקומו היחסי נתון.
- שגרות נוספות – מציאת כל הרשומות שהמפתח שלהם נמצא בקטע נתון (טווח מספרים כלשהו).

ג. מודול עץ ערכי מיקום - "RB_TREE": המודול אחראי על מימוש שגרות בסיסיות של עץ א"ש ובכלל כך –

- שגרות בסיסיות של עץ א"ש – הכנסה, חיפוש, מחיקה, מינימום, מקסימום, עוקב, קודם, סריקה תוכית, סיבובים, תיקונים לאיזון גובה העץ.

בוצע מימוש נפרד בין מבנה עץ ערכי מיקום לעץ א"ש מ-2 סיבות:

1. הצורך לחסוך ביעילות מקום הביא לכך שעבור עץ א"ש ינוהל מבנה שונה, דל יותר, וכמו כן ערכי המפתחות בשני המבנים אינם מאותו טיפוס (האחד שלם/מחרוזת והשני ממשי). להתמודדות עם המבנים השונים הוחלט לפצל את המימוש עבור כל סוג עץ בנפרד.
2. בשל הקשר המקביל בין צומת לצומת. נדרש היה לקבוע מבנה שיהיה "חזק יותר" וינהל שגרת הכנסה שונה, שמנהלת את הקשר בין 2 הצמתים החדשים שהוכנסו (צומת אחד לכל סוג עץ).

ד. מודול ראשי - "MAIN": אחראי על אתחול התכנית כולה, הקצאת ושחרור הזיכרון, שליטה ובקרה על רצף התכנית ומעבר נתונים בין המודולים השונים, וייצוא דוח פלט סופי עם תוצאות ההשוואה ו/או חיזוי שגיאות, בסיוע מודול הקלט/פלט.

❖ ניתוח סדר גודל זמן הריצה של השגרות השונות (פעולות שינוי ושאלות)

א. פעולת הפקדה או משיכה

אופן מימוש: חיפוש רשומת הלקוח בעץ האדום-שחור שבו הצמתים מסודרים ע"פ מפתח של מספר לקוח, וע"י כך הגעה אל הרשומה המקבילה בעץ ערכי מיקום שמכיל את מלוא הנתונים הנלווים לרבות היתרה הנוכחית, שהיא המפתח שעל פיו מסודר עץ ערכי המיקום. מאחר שעדכון ערך היתרה עלול להפר תכונת עץ החיפוש הבינארי, אופן המימוש שנקט מאומץ בהשראה ממימוש ערימה בינארית: הוספת צומת חדש עם ערך היתרה העדכני, כך שצומת זה כבר ימוקם במקומו המתאים, ומחיקת הצמתים הישנים. אמנם די היה בהוספה ומחיקת הצומת מעץ ערכי המיקום בלבד, ובעץ אדום-שחור רק לעדכן מצביעים, אך מאחר שהמימוש של פעולת ההכנסה שנקט בפרויקט כולל הכנסה של הצומת ל-2 העצים באותה השגרה, ממילא מוכנס הצומת הנוסף, לכן יש צורך למחוק את הישן. פעולת מחיקה זו ממילא אינה משפיעה זמן הריצה מבחינה אסימפטוטית.

פסידוקוד:

DEPOSIT-WITHDRAW (RB-Tree, customer-num, amount)

1. RB-node \leftarrow ITERATIVE-TREE-SEARCH (root[RB-Tree], customer-num)
2. if RB-node = nil[T] \triangleright not found
3. return error: "customer does not exist"
4. OS-node \leftarrow parallel-node[RB-node]
5. key[new-node] \leftarrow key[OS-node]
 \triangleright Copy key & satellite data from current node into a new one
6. amount[new-node] \leftarrow amount[OS-node] + amount
7. OS-INSERT(T, new-node)
8. OS-DELETE(T, OS-node)
9. RB-DELETE(T, RB-node)

ניתוח סדר גודל: לוגריתמי ביחס לכמות הלקוחות $O(\lg n)$. פירוט:

בשורה 1 מבוצעת שגרת החיפוש שזמן הריצה שלה ליניארי לגובה העץ h . בעץ אדום שחור הגובה הוא לכל היותר $h = O(\lg n)$, לכן שורה 1 מבוצעת ב- $O(\lg n)$. שורות 2-6 מבוצעות בזמן קבוע $O(1)$.

בשורה 7 מבוצעת שגרת הכנסה שהיא מימוש מורחב של RB-INSERT שזמן ריצתה הוא כידוע $O(\lg n)$ ¹. ההרחבה כוללת תחזוקה בזמן קבוע $O(1)$ עבור ערכי מיקום, מקסימום ומינימום, קריאה להכנסה נוספת עבור צומת חדש בעץ אדום שחור המקביל בזמן $O(\lg n)$, ותחזוקת המצביעים ביניהם בזמן קבוע $O(1)$.


סה"כ שורה 7 מבוצעת בזמן לוגריתמי: $O(\lg n) + O(\lg n) + O(1) \neq O(\lg n)$

השורה 8 מבוצעת שגרת מחיקה שהיא מימוש מורחב של RB-DELETE שזמן ריצתה הוא כידוע $O(\lg n)$ ². ההרחבה כוללת מציאה ועדכון של המקסימום ו/או מינימום במידה והצומת שנמחק היה בעל ערך המפתח המקסימאלי (או המינימאלי). פעולות אלו של מקסימום ומינימום אורכות $O(\lg n)$ בעץ אדום-שחור, לכן הזמן כולו של שגרת ההכנסה המורחבת נשאר לוגריתמי.

שורה 9 היא מימוש ישיר של RB-DELETE ומבוצעת כאמור ב- $O(\lg n)$.

¹ עמוד 241 בספר הלימוד – פרק 13.3
 עמוד 246 בספר הלימוד – פרק 13.4

ב. פעולת הצטרפות לקוח חדש

 **אופן מימוש:** וידוא שאין לקוח בעל מפתח זהה ע"י חיפוש צומת בעץ האדום-שחור שבו הצמתים מסודרים ע"פ מפתח של מספר לקוח. אם אכן החיפוש נכשל, מבוצעת קריאה לשגרת הכנסה מורחבת, הכוללת הכנסת 2 צמתים בהתאמה ל-2 העצים. בעץ אדום-שחור מבוצעת הכנסה לפי מפתח מס' לקוח, ובעץ ערכי מיקום מבוצע הכנסה לפי מפתח יתרה. מבוצע קישור מצביעים בין 2 הרשומות, ועדכון צמתי מקסימום ומינימום (לפי מפתח יתרה) במידה וערכי הצמתים החדשים דורשים זאת.

 **פסידוקוד:**

NEW-CUSTOMER (T, z)


1. RB-node \leftarrow ITERATIVE-TREE-SEARCH (root[RB-Tree], customer-num[z])
2. if RB-node \neq nil[T] \triangleright found
3. return error: "customer with same key already exists"
4. OS-INSERT(T, z)


 **ניתוח סדר גודל:** לוגריתמי ביחס לכמות הלקוחות $O(\lg n)$. פירוט:

בשורה 1 מבוצעת שגרת החיפוש בעץ אדום-שחור, לכן שורה זו מבוצעת ב- $O(\lg n)$. שורות 2,3 מבוצעות בזמן קבוע $O(1)$.

בשורה 4 מבוצעת שגרת הכנסה שממששת בתוכה 2 הכנסות – האחת עבור צומת לעץ ערכי מיקום המסודר לפי מפתח יתרה והשנייה עבור העץ אדום-שחור. מפתח מספר לקוח. ההכנסות הן מימוש של השגרה RB-INSERT שזמנה $O(\lg n)$ בתוספת זמן קבוע לעדכון שדות גודל עבור תחזוקת ערכי מיקום, זמן קבוע לבדיקת ערכי מינימום ומקסימום ועדכון מצביעים אליהם במידת הצורך, והזמן הקבוע הנדרש לעדכון מצביעים בין ה-2 הצמתים החדשים כך שיצביעו אחד על השני. סה"כ $O(\lg n) + O(\lg n) + O(1) = O(\lg n)$.

ג. פעולת עזיבת לקוח

 **אופן מימוש:** חיפוש הלקוח ע"פ מספרו בצומת אדום-שחור, ותוך כך הגעה אל הצומת המקביל בעץ ערכי-מיקום ומחיקת 2 הצמתים.

 **פסידוקוד:**

CUSTOMER-LEAVE (T, customer-num)

1. RB-node \leftarrow ITERATIVE-TREE-SEARCH (root[RB-Tree], customer-num)
2. if RB-node = nil[T] \triangleright not found
3. return error: "customer does not exist"
4. OS-node \leftarrow parallel-node[RB-node]
5. RB-DELETE(T, RB-node)
6. OS-DELETE(T, OS-node)

 **ניתוח סדר גודל:** לוגריתמי ביחס לכמות הלקוחות $O(\lg n)$. פירוט:

בשורה 1 מבוצעת שגרת החיפוש בעץ אדום-שחור, לכן שורה זו מבוצעת ב- $O(\lg n)$. שורות 2-4 מבוצעות בזמן קבוע $O(1)$.

שורה 5 היא מימוש ישיר של RB-DELETE וזמנה הוא כאמור $O(\lg n)$. בשורה 6 מבוצעת שגרת המחיקה המורחבת שכוללת חיפוש מינימום ומקסימום עדכניים ועדכון המצביעים אליהם במקרה שהאיבר שמחק היה מינימום/מקסימום. חיפוש מינימום/מקסימום אורכים כל אחד זמן ריצה לינארי ביחס לגובה העץ, לכן בעץ ערכי המיקום שהוא עץ אדום שחור, שגובהו לכל היותר לוגריתמי ביחס לכמות

הצמתים הרי שפעולה זו אורכת זמן לוגריתמי $O(\lg n)$. עדכון המצביעים מבוצע בזמן קבוע $O(1)$. סה"כ במקרה הגרוע ביותר בו יש צורך לעדכן מינימום ומקסימום (למשל כאשר קיים רק צומת אחד בעץ שהוא גם המינימום וגם המקסימום בו זמנית, ויש צורך לעדכן את המצביעים לזקיפי מינימום ומקסימום בהתאמה על מנת שיתעדכנו בהכנסות הבאות), זמן הריצה של פעולת העדכון של עזיבת לקוח הוא זמן הריצה של חיפוש, 2 מחיקות ומציאת מינימום ומקסימום:

$$O(\lg n) + O(\lg n) + O(\lg n) + O(\lg n) + O(\lg n) + O(1) = O(\lg n).$$

ד. שאלת יתרה

אופן מימוש: חיפוש צומת בעץ אדום-שחור ע"פ מספר לקוח, ואכן החיפוש צלח גישה באמצעות המצביע שבצומת זה אל הצומת המקביל בעץ ערכי-מיקום המכיל את הנתונים הנלווים והדפסת היתרה.

פסידוקוד:

QUERY_BALANCE (T, customer-num)

1. RB-node \leftarrow ITERATIVE-TREE-SEARCH (root[RB-Tree], customer-num)
2. if RB-node = nil[T] \triangleright not found
3. return error: "customer does not exist"
4. OS-node \leftarrow parallel-node[RB-node]
5. print balance[OS-node]

ניתוח סדר גודל: לוגריתמי ביחס לכמות הלקוחות $O(\lg n)$. פירוט:

בשורה 1 מבוצעת שגרת החיפוש בעץ אדום-שחור, לכן שורה זו מבוצעת ב- $O(\lg n)$. שורות 2-5 מבוצעות בזמן קבוע $O(1)$. סה"כ $O(\lg n)$.

ה. שאלת מקסימום

אופן מימוש: הרחבת מבנה הנתונים ב-2 מצביעים שיכילו בכל נקודת זמן הצבעה לצומת בעל ערך היתרה המקסימאלי ולצומת בעל ערך היתרה המינימאלי בעץ ערכי-מיקום, בו היתרה מהווה את שדה המפתח לארגון הרשומות. התחזוקה השוטפת מבוצעת בשגרות ההכנסה והמחיקה.

- **אתחול:** באתחול הראשוני של מבנה הנתונים, מצביע המקסימום מצביע על צומת ייעודי שתפקידו זקוף מקסימום בעל הערך $-\infty$ (מינוס אינסוף), ומצביע המינימום יצביע על צומת ייעודי שתפקידו זקוף מינימום בעל הערך $+\infty$ (אינסוף).


- **תחזוקה:**

○ **הכנסה:** בכל הכנסה של צומת חדש ערך המפתח שלו ישווה לערך המקסימום וערך המינימום הנוכחיים, ובמידת הצורך יעודכנו המצביעים לצומת החדש.

נכונות: הצומת הראשון שיוכנס, ערכו כמובן יהיה גדול מ- $-\infty$, ולכן הוא יהיה גדול מה- "מקסימום" הנוכחי, הזקיף, ובכך המקסימום יעודכן לצומת החדש בעל מפתח "אמיתי". באופן סימטרי, לצומת הראשון שיוכנס בוודאי יהיה ערך קטן מאינסוף, ולכן הוא יהיה קטן מערך המינימום של הזקיף, כך שהמינימום יעודכן להצביע על הצומת


החדש. בכל הוספת עוקבת, ההשוואה והעדכון במידה והתנאי מתקיים נכונים באופן טריוויאלי.

○ מחיקה: בעת מחיקה מבוצעת בדיקה האם הצומת שנמחק היה המקסימום או המינימום (או שניהם גם יחד, למשל כאשר היה צומת בודד בעץ). אם כן, מבוצע חיפוש לקביעה מחדש של הצומת הדורש (מינימום, מקסימום או שניהם יחד). במידה ונמצא כזה הצביע מעודכן להצביע עליו. אחרת, המצביע מעודכן להצביע על הזקיף התואם.


 פסידוקוד: האתחול מבוצע עם יצירת מבנה הנתונים, והתחזוקה כאמור מבוצעת בנפרד בשגרות ההכנסה והמחיקה. לפיכך כל מה שנדרש בשביל שאילתת המקסימום הוא לגשת אל המצביע:


QUERY_MAX (T)

1. $x \leftarrow \max[T]$
2. if $x = \text{nil}[T]$ \triangleright no nodes
3. return error: "Structure is empty"
4. return x

 ניתוח סדר גודל: קבוע $O(1)$. פירוט: האתחול שמבוצע בזמן קבוע ביצירת מבנה נתונים, והתחזוקה השוטפת בשגרות ההכנסה ומחיקה (תחזוקה שאינה משפיעה על זמן הריצה האסימפטוטי של שגרות אלה) מאפשרות אופטימיזציה של זמן הריצה של השאילתה עבור מקסימום ומינימום, ע"י גישה ישירה בזמן קבוע למצביע שתמיד מצביע על הצומת בעל הערך המקסימאלי/מינימאלי ובדיקה אם הוא הזקיף או שאכן זה צומת ממשי.


1. שאילתת הלקוחות שיתרתם שלילית (QUERY_NEGATIVE_BALANCE)

 אופן מימוש: שימוש בשגרה $^3\text{RB_ENUMERATE}$, המוצאת את כל הרשומות שערך המפתח שלהם נמצא בטווח נתון, ע"י הפעלתה על הקטע החצי פתוח $[-\infty, 0)$. באופן זה הפלט יהיה מורכב מכל המפתחות שערכם שלילי ממש.

 פסידוקוד:

OS_ENUMERATE (T, x)

1. if $x \neq \text{nil}[T]$ and $\text{key}[x] \geq -\infty$
2. OS_ENUMERATE (T, left[x])
3. if $x \neq \text{nil}[T]$ and $\text{key}[x] \geq -\infty$ and $\text{key}[x] < 0$
4. print x
5. if $x \neq \text{nil}[T]$ and $\text{key}[x] < 0$
6. OS_ENUMERATE (T, right[x])

 ניתוח סדר גודל: ליניארי $\Theta(n)$. פירוט: נסמן ב- n כמות את סך הצמתים הפנימיים בעץ, וב- m את סך כמות הצמתים הרלוונטיים לפלט, כלומר הצמתים שערכי המפתח שלהם שליליים. ע"פ פירוט הניתוח בתשובה לשאלה י"ב-10 בחוברת הלימוד (עמוד 199), זמן הריצה של מימוש זה הוא $\Theta(m + \lg n)$. במקרה הגרוע ביותר, בו לכל הלקוחות קיימת יתרה שלילית, נקבל כי $m = n$ ואז זמן הריצה הוא $\Theta(m + \lg n) = \Theta(n + \lg n) = \Theta(n)$

³ מבוסס על פיתרון לשאלה 14.2-5 שהוצע במדריך הלמידה בעמוד 199.