**ממן 17 – מבני נתונים ומבוא לאלגוריתמים – ברנדס איתי ת.ז.**

במהלך שאלות 1,3,4 אנו נדרשים להציע מבני נתונים בעזרת מבני נתונים קיימים, שידעו לממש שגרות מסוימות בסיבוכיות מוגדרת. בממ"נים הקודמים טרחתי להפנות אל המקומות המדויקים בספר הלימוד שבהם מוגדר כל מבנה נתונים/שגרה/ניתוח סיבוכיות זמן ריצה, אך הפעם לא אעשה כך, שכן משום שזוהי מהות כל הממ"ן, התשובות יתארכו בהרבה. במהלך הפתרונות אשתמש במבני הנתונים, השגרות, וניתוחי סיבוכיות זמן הריצה שמופיעים בספר הלימוד ובמדריך הלמידה.

**שאלה 1:**

*נציע מבנה נתונים S שיענה על תנאי השאלה.*

*במבנה הנתונים S יהיו:*

* *עץ אדום-שחור T המכיל את המפתחות k הייחודיים במבנה. בנוסף, בכל צומת יהיו השדות הבאים:*
  + *data – שיצביע לרשימה מקושרת דו-כיוונית, שתכיל את כל הרשומות בעלי אותו מפתח k)*
  + *freq – שיצביע אל הצומת המתאימה בתור הקדימויות P של השכיחויות.*
* *תור קדימויות P שישמור שכיחויות של מפתחות.*
  + *נניח שבנוסף לשגרות המוגדרות בעמוד 115, תור הקדימויות תומך גם בשגרה DECREASE-KEY(P, x, k) שמורידה את ערך המפתח של האיבר x ל-k. ניתן לממש שגרה זו באופן סימטרי לHEAP-INCREASE-KEY מעמוד 117 בספר הלימוד, ע"י השוואה לבנים במקום להורה, החלפה בין הצומת לאחת הבנים במקרה הצורך, והמשך טיפול בתת-העץ המושרש בבן שהחלפנו עתה.*
  + *כל מפתח בערימה ישמור את השכיחות של מפתח כלשהו בעץ T.*
  + *בנוסף, בכל צומת נגדיר גם את השדה SourceNode שיצביע אל המפתח המתאים בעץ T.*

*ניגש כעת לאפיון השגרות.*

*INSERT(k, R, S)*

*נשתמש בשגרה TREE-SEARCH(T, k) של עץ חיפוש בינארי.*

*אם המפתח כבר קיים, השגרה תחזיר את הצומת x, ואז ניגש לרשימה המקושרת L בdata[x] ונכניס את הרשומה R אליה בעזרת שגרת הרשימה המקושרת LIST-INSERT(L, R).*

*בנוסף, נקדם את השכיחות שנמצאת בfreq[x] בתור הקדימויות P בעזרת שגרת תור הקדימויות INCREASE-KEY(S, freq[x], i) כאשר i שאנו רוצים לקדם אליו יהיה key[freq[x]]+1.*

*אם המפתח איננו קיים, ניצור צומת חדשה x עם מפתח k ורשימה מקושרת data המאותחלת לרשימה עם איבר אחד, שיכיל את הרשומה R. נשתמש בRB-INSERT(T, x) ע"מ להכניס את הצומת x החדש אל העץ T.*

*ניצור גם צומת חדשה y עם המפתח 1 ועם שדה SourceNode שיפנה אל הצומת x שיצרנו קודם. את הצומת החדשה y נכניס לתור הקדימויות P בעזרת השגרה PRIORITY-QUEUE-INSERT(S, y).*

*נגדיר את freq[x] להפנות את הצומת החדשה שבנינו, y.*

*ננתח את סיבוכיות זמן הריצה של השגרה*

*יהי n מספר צמתי העץ T, וm מספר צמתי תור הקדימויות P.*

*השגרה TREE-SEARCH של עץ אדום-שחור רצה ב, שכן בעץ אדום-שחור גובה העץ לוגריתמי למספר צמתיו.*

*השגרה LIST-INSERT של הרשימה המקושרת רצה בזמן קבוע.*

*השגרה INCREASE-KEY של תור הקדימויות רצה ב.*

*השגרה RB-INSERT של עץ אדום-שחור רצה ב.*

*השגרה PRIORITY-QUEUE-INSERT של תור הקדימויות רצה ב.*

*כל שאר הפעולות רצות בזמן קבוע.*

*נבחין גם שאנו קוראים לRB-INSERT ו- PRIORITY-QUEUE-INSERTיחדיו, ולכן בכל פעם שיתווסף מפתח לעץ T, יתווסף מפתח לתור הקדימויות P, וכן להפך. לכן מתקיים .*

*לסיכום, השגרה רצה ב, כנדרש.*

*INCREASE(k, d, S)*

*נשתמש בשגרה TREE-SEARCH(T, k) של עץ חיפוש בינארי.*

*אם המפתח איננו קיים, נעלה חריגה.*

*אם המפתח כבר קיים, השגרה תחזיר את הצומת x.*

*ניגש לרשימה המקושרת בdata[x], ונוציא ממנה את הרשומה R (נחקה פעולת POP מראש הרשימה ע"י הגדרת R 🡨 ואז נבצע LIST-DELETE(R)).*

*את הרשומה R נכניס בחזרה למבנה S בעזרת השגרה INSERT(k+d, R, S) (רק שהפעם הכנסנו את הרשומה R במפתח k+d).*

*עלינו גם להפחית את השכיחות של המפתח k ב-1. נעשה זאת ע"י קריאה לשגרה*

*DECREASE-KEY(P, freq[x], i) כאשר i שאנו רוצים להפחית אליו יהיה key[freq[x]]-1.*

*ננתח את סיבוכיות זמן הריצה של השגרה*

*יהי n מספר צמתי העץ T, וm מספר צמתי תור הקדימויות P.*

*השגרה TREE-SEARCH של עץ אדום-שחור רצה ב, שכן בעץ אדום-שחור גובה העץ לוגריתמי למספר צמתיו.*

*השגרה LIST-DELETE של הרשימה המקושרת הדו-כיוונית רצה בזמן קבוע.*

*השגרה INSERT רצה, כפי שהוכחנו קודם לכן בשאלה, ב.*

*השגרה DECREASE-KEY של תור הקדימויות רצה בסיבוכיות לינארית לגובהו של מבנה תור הקדימויות, שכן האלגוריתם רץ במקרה הגרוע על כל העץ מהשורש אל העלים, ובכל פעם מבצע מספר פעולות קבוע.*

*תור הקדימויות ממומש באמצעות ערימת-מקסימום, שגובהו לוגריתמי ביחס למספר צמתיו, ולכן השגרה רצה ב.*

*כל שאר הפעולות רצות בזמן קבוע.*

*מאותה הצדקה מניתוח הסיבוכיות של השגרה INSERT, מתקיים .*

*לסיכום, השגרה רצה ב, כנדרש.*

*FIND-NUM(k, S)*

*נשתמש בשגרה TREE-SEARCH(T, k) של עץ חיפוש בינארי.*

*אם המפתח איננו קיים, נעלה חריגה.*

*אם המפתח כבר קיים, השגרה תחזיר את הצומת x.*

*כעת נחזיר את שכיחות הצומת – key[freq[x]].*

*ננתח את סיבוכיות זמן הריצה של השגרה*

*יהי n מספר צמתי העץ T.*

*השגרה TREE-SEARCH של עץ אדום-שחור רצה ב, שכן בעץ אדום-שחור גובה העץ לוגריתמי למספר צמתיו.*

*כל שאר הפעולות רצות בזמן קבוע.*

*לכן, השגרה רצה בסיבוכיות של , כנדרש.*

*MODE(S, k)*

*נקבל את ערך המקסימום x של תור הקדימויות P בעזרת השגרה MAXIMUM.*

*נפנה אל הצומת המקורית ונחזיר את המפתח שלה, בעזרת key[SourceNode[x]].*

*השגרה MAXIMUM רצה ב-. שאר הפעולות מתבצעות גם הן בזמן קבוע.*

*לכן השגרה רצה ב, כנדרש.*

*ובכך מצאנו מבנה נתונים S שעונה על תנאי השאלה.*

**שאלה 2:**

*בפתרון נשתמש בשגרת RB-INSERT מראשית עמוד 236 בספר הלימוד.*

*ניתן לתקן את העץ ע"מ שיהיה עץ אדום-שחור חוקי ע"י השגרה הבאה:*

FIX-RB(T, y, z)

► if y is the root

1 if y==root[T]

2 then root[T] 🡨 z

► if y is a left child of its parent

3 elseif y == left[p[y]]

4 then left[p[y]] 🡨 z

► if y is a right child of its parent

5 else

6 then right[p[y]] 🡨 z

7 p[z] 🡨 p[y]

8 color[z] 🡨 BLACK

9 RB-INSERT(T, y)

נסביר בקצרה את נכונות השגרה:

יהי T העץ האדום-שחור, ו y, x, z הצמתים כמתואר בתנאי השאלה.

נבצע חיבור ישיר בין אביו של y לבין בנו z. בדרך זו, נמחק את כל התת-עץ המושרש בבנו השני של y, x, וגם נמחק את y עצמו (מחיקת y איננה רצויה, אך נוסיף אותו מאוחר יותר).

בשורות 1-2 אנו מטפלים במקרה שבו y הוא השורש, ומגדירים את z להיות השורש החדש.

בשורות 3-4 ו-5-6 אנו מטפלים במקרים שבהם y הוא בן שמאלי/ימני של אביו (בהתאמה) ומחברים את אביו ישירות את בנו z.

בשורה 7 אנו מגדירים את אביו של z להיות אביו של y.

ובכך, בשורות 1-7, אנו מבצעים חיבור ישיר בין אביו של y ל-z.

בשורה 8 אנו צובעים את צבע z לשחור.

בשורה 9 אנו מחזירים את y אל העץ. השגרה RB-INSERT דואגת לצבוע מחדש את הצומת, ולהגדיר מחדש את אביו ובניו של הצומת, לכן אין צורך "לנקות" את הצומת y לפני שאנו מעבירים אותה אל RB-INSERT.

בסוף הריצה אכן קיבלנו עץ שבו נמחק תת-העץ המושרש ב-x, אך נשאלת השאלה מדוע העץ שהתקבל הינו עץ אדום-שחור. נראה זאת ע"י קיום כל תכונות העץ האדום-שחור, כמפורטים בעמוד 230 בספר הלימוד תחת הנושא "תכונותיהם של עצים אדומים-שחורים":

1. **כל צומת הוא אדום או שחור** – העץ המקורי הוא עץ אדום-שחור. לא בוצעה צביעה בצבעים אחרים, לכן חוק זה לא נפגע.
2. **השורש הוא שחור** – אם y הוא השורש, הרי שהוא מוחלף ב-z, שאותו אנו צובעים בשחור בשורה 8. אם y איננו השורש, הרי שאיננו משנים את צבע השורש בשגרה, ומשום שהעץ T במקור היה עץ אדום-שחור, השורש היה ונשאר שחור.
3. **כל עלה (NIL) הוא שחור** – לא שינינו צבע של אף עלה NIL, לכן צבעיהם נשארו שחורים.
4. **אם צומת אדום, שני בניו שחורים** – השינוי היחיד שבוצע הוא חיבור בן שונה לאביו של y, ואת בן זה, z, צבענו בשחור.

אם אביו של y שחור, הרי שאין הפרה של חוק זה.

אם אביו של y אדום, אז בנו החדש z הוא שחור (שכן צבענו אותו בשורה 8), ובנו האחר הוא שחור גם הוא (שכן הבן האחר נשאר ללא שינוי מהעץ המקורי, שהוא עץ אדום-שחור, לכן מחויב שצבעו יהיה שחור).

אם אביו של y לא קיים, שכן y הוא צומת, כמובן שאין הפרה של חוק זה.

כל שאר הצמתים בעץ מקיימים את תנאי זה, שכן העץ המקורי הוא עץ אדום-שחור.

1. **כל המסלולים הפשוטים מצומת נתון לצאצאים מכילים אותו מספר של צמתים שחורים** – כל המסלולים הפשוטים שאינם כוללים את y אינם מושפעים כלל מן השינוי, שכן לא מחקנו או צבענו שום צומת במסלולים הפשוטים מהם אל הצאצאים. משום שהעץ T המקורי הוא עץ אדום-שחור, תכונה זו נשמרת גם כאן.

לגבי המסלולים שכללו בהם את y - כל מסלול כזה עדיין מכיל את אותו מספר של צמתים שחורים, כי מחקנו במסלול זה צומת שחור (את y עצמו) ואז צבענו צומת אדום לשחור (את z), ולכן מכל צומת, כל מסלול פשוט לעלה מכיל את אותו מספר של צמתים שחורים (וכמובן שזה גם יהיה אותו מספר צמתים שהיה בעץ המקורי לפני הרצת השגרה).

ננתח את סיבוכיות השגרה:

שורות 1-8 רצות בזמן קבוע.

שורה 9 רצה בזמן של , שבמקרה שלנו, משום שT הוא עץ שחור-אדום, גובה העץ הוא לוגריתמי למספר צמתיו, ולכן שורה 9 רצה ב, כפי שגם הוכח בעמוד 241 בספר הלימוד תחת הנושא "ניתוח זמן הריצה".

**שאלה 3:**

*נציע מבנה נתונים S שיענה על תנאי השאלה.*

*במבנה הנתונים S יהיו:*

* *עץ אדום-שחור T המכיל את המפתחות k הייחודיים במבנה. בנוסף, בכל צומת יהיו גם השדה sum, שיכיל את סכום כל המפתחות בתת-העץ המושרש בצומת זה. הsum של nil[T] יהיה 0.*
* *שדה min1 שישמור את המפתח הכי קטן במבנה. יכיל NIL בתחילה.*
* *שדה min2 שישמור את המפתח השני הכי קטן במבנה. יכיל NIL בתחילה.*

*ניגש כעת לאפיון השגרות.*

*INSERT(S, k)*

*נשתמש בשגרה TREE-INSERT(T, k) של עץ בינארי וניצור צומת חדשה x.*

*נדאג להגדיר בו את שדה הsum, ע"י כך שנגדיר (\*).*

*כמו כן, משום שהוספת הצומת החדשה תשנה את ערכי הsum באבות הקדמונים שלה, עלינו לעבור בכל האבות הקדמונים של x ולחשב בהם מחדש את הsum (בעזרת הנוסחה (\*)).*

*כעת נטפל בחישוב min2. נבדוק את מספר הצמתים שהיו בעץ טרם ההכנסה. אם לmin2 יש ערך (ז"א הוא לא NIL) אז היו לפחות 2 צמתים. אם לmin2 אין ערך, אבל לmin1 יש ערך, אז היה בדיוק צומת אחד. אחרת, לא היו צמתים במבנה.*

*נטפל אחרת בכל מצב:*

* + *אם לא היו צמתים קודם לכן, כעת יש צומת אחד, ונכניס אותו אל min1. אין היגיון בלהגדיר את min2.*
  + *אם היה קודם צומת אחד, כעת יש 2 צמתים. נתבונן על x ועל min1, נכניס את הקטן מבניהם בmin1 ואת הגדול מבניהם בmin2.*
  + *אם היו קודם 2 צמתים או יותר, עלינו לבדוק האם עלינו לעדכן את min1 ו/או min2. הנ"ל יכול לקרות אם הצומת החדש הוא המינימלי במבנה, או מספר 2 במבנה.*

*נבדוק האם . אם כן, נמיין את שלושת הצמתים x, min1, min2 לפי המפתח שלהם ונכניס את הקטן ביותר אל min1 ואל הקטן אחריו לmin2.*

*ננתח את סיבוכיות זמן הריצה של השגרה*

*יהי n מספר צמתי העץ T.*

*השגרה TREE-INSERT של עץ אדום-שחור רצה ב, שכן בעץ אדום-שחור גובה העץ לוגריתמי למספר צמתיו.*

*חישוב שדה הsum בכל צומת נעשה בזמן קבוע. עלינו לרוץ מהצומת החדש מעלה עד לשורש, ולחשב מחדש את שדה הsum בכל צומת. במקרה הגרוע נאלץ לרוץ על כל גובה העץ, ז"א, על h צמתים, ובכל פעם לבצע עבודה קבועה. לכן חישובי הsum כולם ירוצו ב.*

*חישוב הmin1, min2 נעשה בזמן קבוע.*

*השגרה LIST-DELETE של הרשימה המקושרת הדו-כיוונית רצה בזמן קבוע.*

*לסיכום, השגרה רצה ב, כנדרש.*

*DELETE(S, x)*

*נשתמש בשגרה TREE-DELETE(T, x) של עץ חיפוש בינארי.*

*משום שמחיקת הצומת החדשה תשנה את ערכי הsum באבות הקדמונים שלה, עלינו לעבור בכל האבות הקדמונים של x ולחשב בהם מחדש את הsum (בעזרת הנוסחה (\*)).*

*בנוסף, עלינו לבדוק האם עלינו לעדכן את min1, min2.*

*אם הצומת שנמחק הוא אחד מהם, עלינו למצוא ערך חדש מתאים.*

*נמצא את הצומת העוקב y לmin2 בעזרת השגרה TREE-SUCCESSOR של עץ בינארי.*

*כעת נשווה בין y לבין הצומת מ{min1, min2} שאיננו x שמחקנו. את הקטן מבניהם נשים בmin1 ואת הגדול מבניהם נשים בmin2.*

*ננתח את סיבוכיות זמן הריצה של השגרה*

*יהי n מספר צמתי העץ T.*

*השגרה TREE-DELETE של עץ אדום-שחור רצה ב, שכן בעץ אדום-שחור גובה העץ לוגריתמי למספר צמתיו.*

*חישוב שדה הsum בכל צומת נעשה בזמן קבוע. עלינו לרוץ מהצומת שמחקנו מעלה עד לשורש, ולחשב מחדש את שדה הsum בכל צומת. במקרה הגרוע נאלץ לרוץ על כל גובה העץ, ז"א, על h צמתים, ובכל פעם לבצע עבודה קבועה. לכן חישובי הsum כולם ירוצו ב.*

*השגרה TREE-SUCCESSOR של עץ בינארי רצה ב.*

*כל שאר הפעולות רצות בזמן קבוע.*

*לסיכום, השגרה רצה ב, כנדרש.*

*PAIR-DIFF(S, d)*

*יהי n מספר האיברים במבנה. יהי A מערך בגודל n.*

*נבצע סריקה תוכית בעץ T החל מהאיבר המינימלי עד לאיבר המקסימלי.*

*נעשה זאת ע"י שימוש בmin1 להשגת המינימום, ואז בTREE-SUCCESSOR ע"מ להתקדם מכל איבר לאיבר הבא אחריו בתור בסריקה תוכית.*

*בכל פעם נוסיף את המפתח לA.*

*כעת A מכיל את מפתחות העץ T כאשר הם ממוינים.*

*כעת ניתן למצוא 2 מפתחות שהפרש המפתחות שלהם הוא בדיוק d, אם יש כאלה, בדרך הבאה:*

*נגדיר את i להצביע על האיבר הראשון, ואת j להצביע על האיבר האחרון. j מכיל את האיבר הגדול ביותר, וi מכיל את האיבר הקטן ביותר. כמובן ש .*

*נרוץ בלולאה עד שi,j נפגשים, ז"א, עד שמתקיים .*

*בכל איטרציה, נחשב את ההפרש בין A[j] ל A[i]. אם ההפרש הוא d, מצאנו את אשר רצינו. נחזיר את d והשגרה תסתיים.*

*אחרת, נבדוק האם ההפרש הוא גדול מd:*

* + *אם הוא גדול, נפחית 1 מ-j.*
  + *אם הוא קטן, נפחית 1 מ-i.*

*ונמשיך לאיטרציה הבאה בלולאה.*

*אם הגענו לסוף הלולאה, הרי שלא נמצא הפרש איברים שהוא בדיוק d, ונחזיר NIL.*

*ננתח את סיבוכיות זמן הריצה של השגרה*

*יהי n מספר צמתי העץ T.*

*אנו רצים מהאיבר המינימלי עד האיבר המקסימלי בעזרת סריקה תוכית בעץ. אנו עושים זאת ע"י n-1 קריאות לשגרה TREE-SUCCESSOR.*

*כפי שהוכח בשאלה י-8 בעמוד 177 במדריך הלמידה, זמן הריצה של סריקה תוכית הוא .*

אנו מאתחלים את i,j להצביע על תחילת וסוף המערך, והלולאה רצה עד ש-. בכל איטרציה של הלולאה אנו מקדמים את i ב1, או מפחיתים את j ב1, או יוצאים מן הלולאה. במקרה הגרוע אנו רצים עד ש, ואז פירוש הדבר הוא שהלולאה רצה n-1 פעמים. כל איטרציה רצה בסיבוכיות קבועה. לפיכך, הלולאה רצה בזמן של .

*כל שאר הפעולות רצות בזמן קבוע.*

*לסיכום, השגרה רצה ב, כנדרש.*

*SUM(S, k)*

*נשתמש בשדה sum שתחזקנו במיוחד בעץ.*

*נטפל זאת בדרך השלילה: נגדיר משתנה sum עם ערך ראשוני sum[root[T]], ואז נחסיר ממנו את כל סכומי המפתחות שגדולים מk.*

*נגדיר את x להיות שורש העץ, ונרוץ בלולאה כל עוד x הוא לא NIL.*

*בגוף הלולאה, נשווה בין k ל-key[x].*

* + *אם , נחסיר מsum את key[x] (שכן מפתחו גדול מ-k), וגם נחסיר את sum[right[x]] (שכן כל התת-עץ הימני של x בוודאי מכיל מפתחות גדולים מk). נגדיר x 🡨 left[x], שכן יתכן שבתת-עץ המושרש בבן השמאלי יהיו עוד צמתים עם סכומי מפתחות גדולים מk שנצטרך להתחשב בהם, ונמשיך לאיטרציה הבאה.*
  + *אחרת, מתקיים ולכן המפתח של x וכל התת-עץ המושרש בבן השמאלי של x מקיימים את התנאי שכל המפתחות קטנים או שווים ל-k, ולא נרצה להחסיר אותם מsum. נגדיר x 🡨 right[x], שכן יתכן שבתת-עץ המושרש בבן הימני יהיו עוד צמתים עם סכומי מפתחות גדולים מk שנצטרך להתחשב בהם, ונמשיך לאיטרציה הבאה.*
  + *אם הרי שהגענו לצומת הראשון (מלמעלה) שמפתחו k. מפתח x עצמו לא עולה על k, וכך גם התת-עץ המושרש בבן השמאלי. עם זאת, התת-עץ המושרש בבן הימני כולו מכיל מפתחות עם ערכים גדולים מk, ולכן עלינו להחסיר את סכום כולו מsum, ע"י .*

*בסוף הריצה נחזיר את sum.*

*ננתח את סיבוכיות זמן הריצה של השגרה*

*יהי n מספר צמתי העץ T.*

*במהלך הלולאה אנו מתחילים משורש העץ, ובכל פעם ממשיכים לבן השמאלי או הימני, או שאם , אנו מפסיקים את הריצה.*

*במקרה הגרוע, נרוץ על כל גובה העץ.*

*בכל איטרציה אנו מבצעים פעולות בזמן קבוע.*

*לכן השגרה רצה ב.*

*כמובן ש , שכן בעץ אדום-שחור גובה העץ לוגריתמי למספר צמתיו.*

*לסיכום, השגרה רצה ב, כנדרש.*

*MIN2(S)*

*נחזיר את key[min2]. כמובן שהשגרה רצה ב, כנדרש.*

*ובכך מצאנו מבנה נתונים S שעונה על תנאי השאלה.*

**שאלה 4:**

*נציע מבנה נתונים S שיענה על תנאי השאלה.*

*במבנה הנתונים S יהיו:*

* *רשימה מקושרת דו כיוונית ST שתכיל בתוכה מצביעים למפתחות שהוכנסו. נשתמש בה כמחסנית ע"מ לגשת אל הרשומות האחרונות שהוכנסו למבנה.*
* *עץ אדום-שחור T המכיל את המפתחות k הייחודיים במבנה. בנוסף, בכל צומת יהיו השדות הבאים:*
  + *data – שיצביע לרשימה מקושרת דו-כיוונית בעלת מצביעי ראש וזנב, שתכיל את כל הרשומות בעלי אותו מפתח k*
    - *כל איבר ברשימה יכיל גם שדה ST\_ptr שיפנה אל התא המתאים שנוצר ברשימה ST.*
  + *freq – שיצביע אל הצומת המתאימה בתור הקדימויות P של השכיחויות.*
* *תור קדימויות P שישמור שכיחויות של מפתחות.* ***מוגדר באופן זהה לתור הקדימויות P שבשאלה 1.***

*ניגש כעת לאפיון השגרות.*

*INSERT(S, k)*

*נשתמש בשגרה TREE-SEARCH(T, k) של עץ חיפוש בינארי.*

1. ***אם המפתח כבר קיים****, השגרה תחזיר את הצומת x, ואז ניגש לרשימה המקושרת L בdata[x] ונכניס את המפתח k אליה בעזרת שגרת הרשימה המקושרת LIST-INSERT(L, k).*

*בנוסף, נקדם את השכיחות שנמצאת בfreq[x] בתור הקדימויות P בעזרת שגרת תור הקדימויות INCREASE-KEY(S, freq[x], i) כאשר i שאנו רוצים לקדם אליו יהיה key[freq[x]]+1.*

1. ***אחרת****, אם המפתח איננו קיים, ניצור צומת חדשה x עם מפתח k ורשימה מקושרת data המאותחלת לרשימה עם איבר אחד, שיכיל את המפתח k. נשתמש בRB-INSERT(T, x) ע"מ להכניס את הצומת x החדש אל העץ T.*

*ניצור גם צומת חדשה y עם המפתח 1 ועם שדה SourceNode שיפנה אל הצומת x שיצרנו קודם. את הצומת החדשה y נכניס לתור הקדימויות P בעזרת השגרה PRIORITY-QUEUE-INSERT(S, y).*

*נגדיר את freq[x] להפנות את הצומת החדשה שבנינו, y.*

*נכניס את x אל ראש הרשימה המקושרת ST בעזרת LIST-INSERT(ST, x).*

*בתא שהוכנס אל הרשימה L נגדיר את השדה ST\_ptr שיפנה אל התא שהוכנס לרשימה ST. זאת על-מנת שנוכל למחוק את התא המתאים מST בעת מחיקת התא החדש בזמן קבוע.*

*ננתח את סיבוכיות זמן הריצה של השגרה*

*יהי n מספר צמתי העץ T, וm מספר צמתי תור הקדימויות P.*

*השגרה TREE-SEARCH של עץ אדום-שחור רצה ב, שכן בעץ אדום-שחור גובה העץ לוגריתמי למספר צמתיו.*

*השגרה LIST-INSERT של הרשימה המקושרת רצה בזמן קבוע.*

*השגרה INCREASE-KEY של תור הקדימויות רצה ב.*

*השגרה RB-INSERT של עץ אדום-שחור רצה ב.*

*השגרה PRIORITY-QUEUE-INSERT של תור הקדימויות רצה ב.*

*כל שאר הפעולות רצות בזמן קבוע.*

*נבחין גם שאנו קוראים לRB-INSERT ו- PRIORITY-QUEUE-INSERTיחדיו, ולכן בכל פעם שיתווסף מפתח לעץ T, יתווסף מפתח לתור הקדימויות P, וכן להפך. לכן מתקיים .*

*לסיכום, השגרה רצה ב, כנדרש.*

*DELETE(S, k)*

*נשתמש בשגרה TREE-SEARCH(T, k) של עץ חיפוש בינארי.*

*אם המפתח איננו קיים, נעלה חריגה.*

*אם המפתח כבר קיים, השגרה תחזיר את הצומת x.*

*נבדוק האם קיימים ערכים ברשימה המקושרת, ע"י בדיקת השכיחות בfreq[x]. אם השכיחות היא 0, נעלה חריגה.*

*אחרת, ניגש לרשימה המקושרת L בdata[x], ניגש אל האיבר הראשון y בעזרת head[L] ונמחק אותו בעזרת LIST-DELETE(L, y).*

*נמחק גם את המצביע הרלוונטי מהרשימה ST, ע"י LIST-DELETE(ST, ST\_ptr[y]).*

*בנוסף, נפחית את השכיחות שנמצאת בfreq[x] בתור הקדימויות P באחד בעזרת שגרת תור הקדימויות DECREASE-KEY(P, freq[x], i) כאשר i שאנו רוצים להפחית אליו יהיה key[freq[x]]-1.*

*ננתח את סיבוכיות זמן הריצה של השגרה*

*יהי n מספר צמתי העץ T, וm מספר צמתי תור הקדימויות P.*

*השגרה TREE-SEARCH של עץ אדום-שחור רצה ב, שכן בעץ אדום-שחור גובה העץ לוגריתמי למספר צמתיו.*

*השגרה LIST-DELETE של הרשימה המקושרת הדו-כיוונית רצה בזמן קבוע.*

*השגרה DECREASE-KEY של תור הקדימויות רצה בסיבוכיות לינארית לגובהו של מבנה תור הקדימויות, שכן האלגוריתם רץ במקרה הגרוע על כל העץ מהשורש אל העלים, ובכל פעם מבצע מספר פעולות קבוע.*

*תור הקדימויות ממומש באמצעות ערימת-מקסימום, שגובהו לוגריתמי ביחס למספר צמתיו, ולכן השגרה רצה ב.*

*כל שאר הפעולות רצות בזמן קבוע.*

*מאותה הצדקה מניתוח הסיבוכיות של השגרה INSERT, מתקיים .*

*לסיכום, השגרה רצה ב, כנדרש.*

*INCREASE(S, k, )*

*ראשית, נמצא את תא המקור (source) ותא היעד (destination).*

*את המקור נמצא בעזרת השגרה TREE-SEARCH(T, k) של עץ חיפוש בינארי.*

*אם המפתח איננו קיים, נעלה חריגה.*

*אם המפתח כבר קיים, השגרה תחזיר את הצומת x – תא המקור.*

*כעת נחפש את הצומת יעד בעזרת השגרה TREE-SEARCH(T, k+).*

***אם המפתח קיים****:*

* + *השגרה תחזיר את הצומת y – תא היעד.*
  + *נוסיף את איברי הרשימה L1=data[x] אל הרשימה L2=data[y] (ע"י כך שניגש לאיבר האחרון בL1 בעזרת TAIL[L1], נגדיר את שדה הnext שלו אל HEAD[L2], נגדיר את שדה הprev של HEAD[L2] שיפנה אל TAIL[L1], ולבסוף נגדיר HEAD[L2] = HEAD[L1]).*
  + *נקדם את השכיחות שנמצאת בfreq[y] בתור הקדימויות P בעזרת שגרת תור הקדימויות INCREASE-KEY(S, freq[y], i) כאשר i שאנו רוצים לקדם אליו יהיה מספר איברי L2 המעודכן: key[freq[x]]+ key[freq[y]].*
  + *נמחק את הצומת x מהעץ T בעזרת השגרה RB-DELETE(T, x).*
  + *בנוסף, נאפס את השכיחות שנמצאת בfreq[x] בתור הקדימויות P באחד בעזרת שגרת תור הקדימויות DECREASE-KEY(P, freq[x], 0).*

**אם המפתח איננו קיים:**

* + *נמחק את הצומת x מהעץ T בעזרת השגרה RB-DELETE(T, x).*
  + *נעדכן כעת את key[x] להיות k+, ואז נבצע RB-INSERT(T, x).*
  + *בכך הצומת x יחזור אל העץ עם המפתח המעודכן, והעץ ישאר אדום-שחור.*

*(אין צורך לעדכן את שאר המצביעים (מתור הקדימויות P או מרשימת המפתחות האחרונים ST), שכן לא מחקנו את הצומת x, אלא רק הסרנו אותה מן העץ והכנסנו אותה לאחר מכן עם מפתח חדש).*

*ננתח את סיבוכיות זמן הריצה של השגרה*

*יהי n מספר צמתי העץ T, וm מספר צמתי תור הקדימויות P.*

*השגרה TREE-SEARCH של עץ אדום-שחור רצה ב, שכן בעץ אדום-שחור גובה העץ לוגריתמי למספר צמתיו.*

*השגרה INCREASE-KEY של תור הקדימויות רצה ב.*

*השגרה DECREASE-KEY של תור הקדימויות רצה גם היא, באופן סימטרי, ב(הסבר מפורט יותר בניתוח השגרה DELETE).*

*כל שאר הפעולות רצות בזמן קבוע.*

*השגרות RB-INSERT ו-RB-DELETE רצות ב.*

*מאותה הצדקה מניתוח הסיבוכיות של השגרה INSERT, מתקיים .*

*לסיכום, השגרה רצה ב, כנדרש.*

*LAST-FREQ(S)*

*נציץ אל הרשומה R שנמצאת בראש הרשימה המקושרת ST בעזרת HEAD[ST].*

*נחזיר את השכיחות ע"י גישה לkey[freq[R]].*

*כל הפעולות רצות בזמן קבוע, ולכן השגרה רצה ב, כנדרש.*

*MODE(S, k)*

*נקבל את ערך המקסימום x של תור הקדימויות P בעזרת השגרה MAXIMUM.*

*נפנה אל הצומת המקורית ונחזיר את המפתח שלה, בעזרת key[SourceNode[x]].*

*השגרה MAXIMUM רצה ב-. שאר הפעולות מתבצעות גם הן בזמן קבוע.*

*לכן השגרה רצה ב, כנדרש.*

*ובכך מצאנו מבנה נתונים S שעונה על תנאי השאלה.*