**תיאור הפרויקט**

קובץ זה מצורף יחד עם תוכנית המכילה את הקוד המתאים.

הבעיה המוצגת בממ"ן 16 הינה מציאת k האיברים הקטנים ביותר במערך בגודל n בסדר עולה.

לצורך פיתרון בעיה זו, השתמשנו במבנה הנתונים עץ אדום-שחר כאשר ישנה הגבלה על גודל העץ (גודל k).

רוב השגרות מבוססות על-פי האלגוריתמים המוכחים בספר, לכן שגרות אלו לא נוכיח ורק נצרף את העמודים המתאימים.

השגרות שאינן מופיעות בספר נוכיח את נכונותן כולל סיבוכיות זמן ריצה וסיבוכיות מקום.

השתמשנו במחלקות גנריות למען פשטות הקוד ובכדי שתהיה אפשרות לעבוד עם כל סוג משתנה.

את התוכנית החלטנו לממש בשפת Java.

**המחלקה RedBlackNode**

המחלקה מייצגת צומת אחד בעץ אדום-שחור.

**משתני המחלקה:**

* BLACK = 0, קבוע המייצג את הצבע שחור.
* RED = 1, קבוע המייצג את הצבע אדום.
* key, המייצג את הערך המספרי של הצומת.
* color, המייצג את הצבע הנוכחי של הצומת.
* parent, המכיל מצביע לאבא של הצומת.
* left, המכיל מצביע לבן השמאלי של הצומת.
* right, המכיל מצביע לבן הימני של הצומת.
* numLeft, המכיל את מספר הצמתים המחוברים לצומת הנוכחי מצידו השמאלי.
* numRight, המכיל את מספר הצמתים המחוברים לצומת הנוכחי מצידו הימני.

**שגרות המחלקה:**

* **public RedBlackNode():**

בנאי המחלקה, שיוצר צומת חדש ומאפס את הערכים (צבע ברירת המחדל של צומת חדש הוא שחור).

* **public RedBlackNode(T key):**

בנאי המחלקה, אשר מקבל ערך למפתח, קורא לבנאי הראשי ולאחר מכן מכניס את המפתח לצומת החדש.

**נכונות השגרות:**

מכיוון שהשגרות היחידות במחלקה הינן הבנאים, והפעולות היחידות שמתבצעות בבנאי אלו פעולות השמה פשוטות למשתני המחלקה, השגרות אכן מבצעות את הדרוש עליהן בסיבוכיות O(1).

**המחלקה RedBlackTree**

המחלקה מייצגת שורש של עץ אדום-שחור.

**משתני המחלקה:**

* nil, משתנה מסוג RedBlackNode שמייצג nil.
* root, משתנה מסוג RedBlackNode שמאותחל ל-nil ומייצג את שורש העץ.
* k, משתנה המייצג את הגודל של העץ. משתנה זה נבחר על-ידי המשתמש.

**שגרות המחלקה:**

* **public RedBlackTree(int k):**

בנאי המחלקה, שיוצר עץ חדש המאותחל ל-nil ומקבל את גודל העץ k מהמשתמש.

* **private Boolean isNil(RedBlackNode node):**

שגרה פרטית המקבלת צומת של עץ אדום-שחור ומחזירה אמת אם הצומת אכן nil ואחרת מחזירה שקר.

* **public int size():**

שגרה ציבורית המחזירה את מספר הצמתים בעץ (גודל העץ). הפונקציה משתמשת בנתונים של המחלקה RedBlackTree כדי לעשות זאת.

* **private void leftRotate(RedBlackNode<T> x):**

שגרה פרטית המקבלת צומת x בעץ אדום-שחור ומבצעת סיבוב שמאלי על x. אנו מניחים כי הבן הימני שלו y אינו nil. "ציר" הסיבוב השמאלי הוא הקשר מ-x ל-y.

* **private void rightRotate(RedBlackNode<T> x):**

שגרה פרטית המקבלת צומת x בעץ אדום-שחור ומבצעת סיבוב ימני על x. השגרה עובדת באופן סימטרי לשגרה של הסיבוב השמאלי, וגם כן אנו מניחים כי הבן השמאלי של x אינו nil.

* **public RedBlackNode<T> TreeMinumum(RedBlackNode<T> x):**

שגרה ציבורית המקבלת צומת בעץ אדום-שחור ובודקת החל מצומת זה מי האיבר בעל המפתח המינימלי, ומחזירה את המצביע לצומת הזה. כאשר השגרה מופעלת על שורש העץ, האיבר שמוחזר הינה האיבר בעל הערך המינימלי בכל העץ.

* **public RedBlackNode<T> TreeMaximum(RedBlackNode<T> x):**

שגרה ציבורית המקבלת צומת בעץ אדום-שחור ובודקת החל מצומת זה מי האיבר בעל המפתח המקסימלי, ומחזירה את המצביע לצומת הזה. כאשר השגרה מופעלת על שורש העץ, האיבר שמוחזר הינה האיבר בעל הערך המקסימלי בכל העץ.

* **public RedBlackNode<T> TreeSuccessor(RedBlackNode<T> x):**

שגרה ציבורית המקבלת צומת בעץ אדום-שחור. השגרה מחזירה את העוקב לצומת x בעץ אם קיים כזה, ו-nil אם x הוא בעל המפתח הגדול ביותר בעץ.

* **public void insert(T key):**

שגרה ציבורית ומשמשת לגישור נוח בין צד המשתמש לשיטות הפרטיות בעזרת העמסה.

השגרה מקבלת ערך להוסיף לעץ. במידה והעץ אינו גדול מידי, היא קוראת לשיטה פרטית אשר מוסיפה צומת בעל ערך זה. אחרת, השגרה בודקת האם האיבר בעל הערך המקסימלי ביותר בעץ גדול מהערך אותו אנו רוצים להוסיף. אם כן – השגרה מוחקת את האיבר המקסימלי בעץ ומוסיפה את צומת חדש לעץ בעל הערך key המבוקשת. אחרת – השגרה יוצאת מבלי לעשות דבר.

* **private void insert(RedBlackNode<T> z):**

שגרה פרטית אשר נקראת מתוך השגרה הציבורית insert. שגרה זו מבצעת הוספה של צומת חדש לעץ אדום-שחור, ומעדכנת בהתאם את הערכים numLeft, numRight. במידה ויש הפרה בחוקיות העץ, מתבצעת קריאה לשיטה אשר אחראית לתיקון הצבעים.

* **private void insertFixup(RedBlackNode<T> z):**

שגרה פרטית אשר נקראת מתוך השגרה הפרטית insert. השגרה מתקנת הפרות בצבעים של העץ בעזרת השגרות leftRotate, rightRotate שיתכן וקרו לאחר הוספת הצומת z לעץ.

* **private void remove(RedBlackNode<T> z):**

שגרה פרטית אשר נקראת מתוך השיטה הציבורית insert. השגרה מקבלת את הצומת בעל הערך המקסימלי בעץ, מוחקת אותו ומעדכנת את הנתונים של numLeft, numRight בעזרת שגרת עזר. במידה ויש הפרה בחוקיות העץ, מתבצעת קריאה לשיטה אשר אחראית לתיקון הצבעים.

* **private void removeFixup(RedBlackNode<T> x):**

שגרה פרטית אשר נקראת מתוך השגרה הפרטית remove. השגרה מתקנת הפרות בצבעים של העץ בעזרת השגרות rightRotate, leftRotate שיתכן וקרו לאחר מחיקת הצומת x מהעץ.

* **private void fixNodeData(RedBlackNode<T> x, RedBlackNode<T> y):**

שגרה פרטית אשר נקראת מתוך השגרה הפרטית remove. השגרה מקבלת את הצומת שנמחק מהעץ y ואת הבן שלו x ומעדכנת את הערכים של numLeft, numRight בהתאם לשינוים שהתבצעו כאשר נמחק הצומת y מהעץ.

* **public void printkMin():**

שגרה ציבורית ומשמשת לגישור נוח בין צד המשתמש לשיטות הפרטיות בעזרת העמסה.

השגרה יוצרת מצביע חדש לשורש וקוראת לשגרה הפרטית printkMin.

* **private void printkMin(RedBlackNode<T> x):**

שגרה פרטית הנקראת מתוך השיטה הציבורית printkMin. השגרה מקבלת את צומת העץ ומדפיסה את כל האיברים שבו. העץ מוגבל ל-k איברים לכן מודפסים k איברים של העץ.

**נכונות השגרות:**

* **public RedBlackTree(int k):**

מכיוון שהפעולות היחידות שמתבצעות בבנאי אלו פעולות השמה פשוטות למשתני המחלקה, השגר אכן מבצעת את הדרוש עליה בסיבוכיות O(1).

* **private Boolean isNil(RedBlackNode node):**

מכך ששגרה זו מחזירה ערך של אמת או שקר בהתאם לנכונות התנאי return node == nil, השגרה מבצעת את הדרוש ונכונה, וכמו כן זמן הריצה שלה הוא O(1).

* **public int size():**

השגרה מחזירה את הסכום של numLeft+numRight+1 כאשר המספר 1 מייצג את שורש העץ.

נכונות השגרה תלויה בהגדרה ובתחזוקה של המשתנים numLeft, numRight, אותם נבדוק בהמשך בשגרות המחיקה וההוספה. זמן הריצה של השגרה הוא O(1).

* **private void leftRotate(RedBlackNode<T> x):**

האלגוריתם לשגרה זו מבוסס על הפסאודוקוד של LEFT-ROTATE אשר ממומש ומוכח בספר בעמוד 234, לכן השגרה מבצעת את הדרוש. בנוסף, על פי עמוד 234 בספר זמן הריצה של השגרה הוא O(1).

* **private void rightRotate(RedBlackNode<T> x):**

האלגוריתם לשגרה זו מבוסס על הפסאודוקוד של LEFT-ROTATE אשר ממומש ומוכח במדריך הלמידה בעמוד 187, לכן השגרה מבצעת את הדרוש. בנוסף, על פי עמוד 234 בספר זמן הריצה של השגרה הוא O(1).

* **public RedBlackNode<T> TreeMinumum(RedBlackNode<T> x):**

האלגוריתם לשגרה זו מבוסס על הפסאודוקוד של TREE-MINUMUM אשר ממומש ומוכח בספר בעמוד 217. מכיוון שעץ אדום-שחור הינו עץ חיפוש בינארי מבסיסו,

השגרה הנ"ל שממומשת עבור עץ חיפוש בינארי מבצעת את הדרוש גם בעבור עץ אדום-שחור. בנוסף, על פי עמוד 217 בספר, זמן הריצה של השגרה על עץ בגובה h הוא O(h). מכיוון שהעץ מכיל k איברים לכל היותר, על-פי עמוד 213 פסקה שלישית, גובה עץ אדום-שחור בעל n צמתים הינו logn, לכן זמן הריצה במקרה הגרוע ביותר של השגרה הוא O(logk).

* **public RedBlackNode<T> TreeMaximum(RedBlackNode<T> x):**

האלגוריתם לשגרה זו מבוסס על הפסאודוקוד של TREE-MINUMUM אשר ממומש ומוכח בספר בעמוד 217. מכיוון שעץ אדום-שחור הינו עץ חיפוש בינארי מבסיסו, השגרה הנ"ל שממומשת עבור עץ חיפוש בינארי, מבצעת את הדרוש גם בעבור עץ אדום-שחור. בדומה לשגרה TREE-MINIMUM, זמן הריצה של השגרה הזו גם הוא O(logk), כאשר k גודל המקסימלי של העץ.

* **public RedBlackNode<T> TreeSuccessor(RedBlackNode<T> x):**

האלגוריתם לשגרה זו מבוסס על הפסאודוקוד של TREE-SUCCESSOR אשר ממומש ומוכח בספר בעמוד 218. מכיוון שעץ אדום-שחור הינו עץ חיפוש בינארי מבסיסיו, השגרה הנ"ל שממומשת עבור עץ חיפוש בינארי, מבצעת את הדרוש גם בעבור עץ אדום-שחור. על-פי עמוד 218 זמן הריצה של השגרה הינו O(h) כאשר h מייצג את גובה העץ. לפי עמוד 213 פסקה שלישית, גובה עץ אדום-שחור הינו logn כאשר n הוא מספר האיברים בעץ. מכך נובע כי עבור עץ אדום-שחור בעל k איברים, זמן הריצה של השגרה במקרה הגרוע ביותר הוא O(logk).

* **public void insert(T key):**

שגרה ציבורית ומשמשת לגישור נוח בין צד המשתמש לשיטות הפרטיות בעזרת העמסה.

השגרה מקבלת ערך להוסיף לעץ. במידה והעץ אינו גדול מידי, היא קוראת לשיטה פרטית אשר מוסיפה צומת בעל ערך זה. אחרת, השגרה בודקת האם האיבר בעל הערך המקסימלי ביותר בעץ גדול מהערך אותו אנו רוצים להוסיף. אם כן – השגרה מוחקת את האיבר המקסימלי בעץ ומוסיפה את צומת חדש לעץ בעל הערך key המבוקשת. אחרת – השגרה יוצאת מבלי לעשות דבר.

* **private void insert(RedBlackNode<T> z):**

שגרה פרטית אשר נקראת מתוך השגרה הציבורית insert. שגרה זו מבצעת הוספה של צומת חדש לעץ אדום-שחור, ומעדכנת בהתאם את הערכים numLeft, numRight. במידה ויש הפרה בחוקיות העץ, מתבצעת קריאה לשיטה אשר אחראית לתיקון הצבעים.

* **private void insertFixup(RedBlackNode<T> z):**

שגרה פרטית אשר נקראת מתוך השגרה הפרטית insert. השגרה מתקנת הפרות בצבעים של העץ בעזרת השגרות leftRotate, rightRotate שיתכן וקרו לאחר הוספת הצומת z לעץ.

* **private void remove(RedBlackNode<T> z):**

שגרה פרטית אשר נקראת מתוך השיטה הציבורית insert. השגרה מקבלת את הצומת בעל הערך המקסימלי בעץ, מוחקת אותו ומעדכנת את הנתונים של numLeft, numRight בעזרת שגרת עזר. במידה ויש הפרה בחוקיות העץ, מתבצעת קריאה לשיטה אשר אחראית לתיקון הצבעים.

* **private void removeFixup(RedBlackNode<T> x):**

שגרה פרטית אשר נקראת מתוך השגרה הפרטית remove. השגרה מתקנת הפרות בצבעים של העץ בעזרת השגרות rightRotate, leftRotate שיתכן וקרו לאחר מחיקת הצומת x מהעץ.

* **private void fixNodeData(RedBlackNode<T> x, RedBlackNode<T> y):**

שגרה פרטית אשר נקראת מתוך השגרה הפרטית remove. השגרה מקבלת את הצומת שנמחק מהעץ y ואת הבן שלו x ומעדכנת את הערכים של numLeft, numRight בהתאם לשינוים שהתבצעו כאשר נמחק הצומת y מהעץ.

* **public void printkMin():**

שגרה ציבורית ומשמשת לגישור נוח בין צד המשתמש לשיטות הפרטיות בעזרת העמסה.

השגרה יוצרת מצביע חדש לשורש וקוראת לשגרה הפרטית printkMin.

* **private void printkMin(RedBlackNode<T> x):**

שגרה פרטית הנקראת מתוך השיטה הציבורית printkMin. השגרה מקבלת את צומת העץ ומדפיסה את כל האיברים שבו. העץ מוגבל ל-k איברים לכן מודפסים k איברים של העץ.