מבני נתונים- תרגיל מעשי 2

ערימה בינומית - מסמך תיעוד

דמיטרי אורלוב, ת.ז. 321892366, שם משתמש- dmitruyo, מייל- dmitruyo@post.tau.ac.il

נעם מזור, ת.ז. 305373458, שם משתמש- noammazor, מייל- noammaz@gmail.com

תיאור המחלקות:

- **מחלקת ערימה בינומית**

```
public class BinomialHeap{
    private int size=0; //Size of the Heap
    private BinomialHeapTree min; //min node in heap
    private BinomialHeapTree trees=null; //list of trees of the heap
    private BinomialHeapTree lastTree=null; //last tree in trees
```

המחלקה מגדירה יישות חדשה של ערימה בינארית עם התכונות המתאימות.
הישות החדשה מטיפוס ערימה בינארית מוגדרת על ידי 4 פרמטרים המופיעים בתור שדות המחלקה:

- שדה `size` מטיפוס מספר שלם `int` המכיל מידע על גודל הערימה הנוכחית(במונחים של מספר הצמתים).
- שדה `min` מטיפוס עץ של ערימה בינומית `BinomialHeapTree` המצביע על האיבר המינימלי בערימה.
- שדה `trees` מטיפוס עץ של ערימה בינומית `BinomialHeapTree` המצביע על תחילת הרשימה המקושרת של העצים המרכיבים ערימה.
- שדה `lastTree` מטיפוס עץ של ערימה בינומית `BinomialHeapTree` המצביע לעץ אחרון ברשימת העצים.

- **מחלקת עץ של ערימה בינומית**

```
private class BinomialHeapTree{
    public int rank=0;//rank of the node
    public int value;//value of the node
    public BinomialHeapTree child=null;//list of the children of the node
    public BinomialHeapTree nextSibling=null;//next tree in the list
```

המחלקה מגדירה יישות חדשה של עץ של ערימה בינומית של ערימה בינומית עם התכונות המתאימות.
הישות החדשה מטיפוס צומת של עץ אדום שחור מוגדרת על ידי 4 פרמטרים המופיעים בתור שדות המחלקה:

- שדה `rank` מטיפוס מספר שלם `int` המכיל מידע על דרגת הצומת הנוכחי.
- שדה `value` מטיפוס מספר שלם `int` המכיל מידע על הערך המפתח של הצומת הנוכחי.
- שדה `child` מטיפוס עץ של ערימה בינומית `BinomialHeapTree` המצביע על תחילת רשימת הבנים של הצומת.
- שדה `nextSibling` מטיפוס עץ של ערימה בינומית `BinomialHeapTree` המצביע על הצומת של העץ הבא ברשימת העצים.

תיאור וסיבוכיות זמן ריצת המתודות כתלות במספר האיברים בעץ:

פונקציות של מחלקת BinomialHeap:

```
public boolean empty()
```

הפונקציה מחזירה "true" אם הערימה אינה מכילה צמתים ו-"false" אחרת.
מידע זה מתקבל על סמך גודל הערימה השמור בתור שדה size של המחלקה. הפונקציה מבצעת בדיקה בודדת האם גודל הערימה שווה ל-0. לכן סיבוכיות זמן ריצת הפונקציה הינה O(1).

```
public void insert(int value)
```

הפונקציה מקבלת פרמטר value המסמן את ערך המפתח אותו נדרש להכניס לערימה. הפונקציה יוצרת עץ חדש מדרגה 0.
העץ בנוי מצומת בודד המכיל בתור מפתח את הערך אותו מעוניינים להכניס לערימה.
אם האיבר מוכנס לערימה בינומית ריקה, העץ החדש הופך להיות עץ יחיד ברשימת העצים הקיימים. אחרת, העץ החדש מתווסף לסוף רשימת העצים הקיימים.
בנוסף גודל הערימה עולה באחד ומתבצע עדכון הערך המינימלי בערימה: אם ההכנסה מתבצעת אל תוך ערימה ריקה או שהאיבר החדש קטן מהמינימום הנוכחי, האיבר החדש הופך להיות איבר מינימלי.
סיבוכיות זמן ריצת הפונקציה insert הינה O(1) כיוון שבכל הכנסה מתבצע מספר קבוע של פעולות.

```
private int deleteMin()
```

הפונקציה מוחקת את הערך המינימלי מהערימה ומחזירה את מספר פעולות ה-linking שהתבצעו במהלך המחיקה. אם גודל הערימה לפני המחיקה היה קטן שווה ל-1, לאחר המחיקה הערימה תהיה ריקה. במקרה זה מתבצע עדכון המצביעים המתאימים ומוחזר 0 כי התהליך לא דרש ביצוע פעולות linking. אחרת, מכניסים את העצים למערך בהתאם לדרגתם (כך שבכל תא עצים בעלי אותה דרגה) בעזרת פונקציית insertToArray, ולאחר מכן מתקנים את העירמה בעזרת fixHeap. גודל הערימה קטן ב-1.
סיבוכיות זמן ריצת הפונקציות fixHeap, insertToArray הוא O(n) במקרה הגרוע. מלבד פונקציות אלו יש מספר קבוע של פעולות ולכן הסיבוכיות הכוללת- O(n)

```
private void insertToArray(BinomialHeapTree node,BinomialHeapTree[] treeArray)
```

הפונקציה מקבלת רשימה של עצים בינומיים מסוג BinomialHeapTree ומכניסה אותם אל תוך המערך. ההכנסה של כל עץ מתבצעת אל תוך תא במערך, כאשר מספר התא אליו מכניסים את העץ שווה לדרגת אותו העץ. בסוף התהליך כל תא במערך יהיה רשימה מקושרת של עצים מאותו דרגה.
הפונקציה עוברת על כל העצים בערימה. במקרה הגרוע דרגת כ עץ היא 0 ולכן סיבוכיות זמן הריצה במקרה הגרוע היא O(n)

```
public int fixHeap(BinomialHeapTree[] treeArray)
```

הפונקציה מתקנת את הערימה לפי הצורך ומחזירה את מספר הפעמים שבוצעה פעולת ה-linking. הפונקציה מקבלת מערך המכיל בכל תא i את כל העצים בדרגה המתאימה i, וממזגת אותם בהתאם לצורך כך שיהיה עץ אחד מכל דרגה. לבסוף מחפשת את המינימום החדש ומחזירה את העצים לרשימה מקושרת אחת.
הפונקציה עוברת על כל העצים ומחברת אותם בהתאם לדרגתם. לכל צומת יש מקסימום הורה אחד, ולכן סיבוכיות זמן הריצה היא לינארית במספר העצים, כלומר, O(n) במקרה הגרוע.

```
public int findMin()
```

הפונקציה מחזירה את הערך המינימלי בערימה אם הערימה אינה ריקה, אם הערימה ריקה מוחזר 1-.
סיבוכיות זמן ריצת הפונקציה findMin הינה O(1) כיוון שבכל הכנסה מתבצע מספר קבוע של פעולות.

```
public void meld (BinomialHeap heap2)
```

הפונקציה מבצעת חיבור של ערימה בינומית נוספת אל הערימה הבינומית הקיימת. אם גודל הערימה הקיימת שווה לאפס או שהמינימום של הערימה החדשה קטן יותר מהמינימום של הערימה הקיימת, המצביע על המינימום הופך להצביע על המינימום של הערימה החדשה. גודל הערימה לאחר הוספת הערימה השניה גדל בגודל הערימה החדשה.
סיבוכיות זמן ריצת הפונקציה meld הינה O(1).

```
public int size()
```

הפונקציה מחזירה את גודל הערימה(כמות הצמתים).
גודל הערימה נשמר בשדה size של מחלקת הערימה. הפונקציה מחזירה את ערך השדה ולכן סיבוכיות זמן ריצת הפונקציה size הינה O(1).

```
public static int sortArray(int[] array)
```

הפונקציה ממיינת מערך בעזרת השימוש בערימה ופעולות בסיסיות המוגדרות על הערימה: delete_min ו-insert. הפונקציה מחזירה את מספר פעולות ה-linking שהתבצעו במהלך המיון המערך. הפונקציה יוצרת ערימה חדשה ומכניסה את n איברי המערך אל תוך ערימה תוך שימוש בפעולת insert. לאחר מכן מתבצעות n פעולות find_min ולאחר כל פעולה כזו האיבר הנמצא מוכנס אל תוך המערך. לאחר כל מציאת המינימום והכנסתו למערך, האיבר נמחק בעזרת delete_min, מספר פעולות ה-linking גדל במספר פעולות ה-linking שהתבצעו במהלך מחיקת האיבר המינימלי הנוכחי.
ראינו בכיתה שסיבוכיות delete_min אמורטייזד היא O(log n), וinsert,find_min נעשים ב O(1), ולכן סיבוכיות זמן הריצה הכוללת היא O(n+nlog n)=O(nlog n)

```
public int[] treesRanks()
```

הפונקציה מחזירה את המערך המכיל את כמות העצים מכל דרגה שנמצאים בערימה בינומית. המערך ממוין בסדר עולה של דרגות. הספירה נעשית ע"י מערך מונים (יוצרים מערך בגודל log n, ועבור כל עץ מגדילים ב-1 את התא המתאים לדרגתו).
הסיבוכיות של פונקציה זו הינה O(n).

```
public BinomialHeapTree getFirstTree()
```

הפונקציה מחזירה את המצביע לעץ הראשון במתוך שימת העצים המוכלים בערימה הבינומית.
המצביע לעץ הראשון שמור בתור שדה במחלקת הערימה הבינומית. הפונקציה מחזירה מצביע ה בזמן הפעלתה ולכן הסיבוכיות שלה הינה O(1).

```
public BinomialHeapTree getLastTree()
```

הפונקציה מחזירה את המצביע לעץ האחרון מתוך רשימת העצים המוכלים בערימה הבינומית.
המצביע לעץ האחרון שמור בתור שדה במחלקת הערימה הבינומית. הפונקציה מחזירה מצביע זה בזמן הפעלתה ולכן הסיבוכיות שלה הינה O(1).

```
public BinomialHeapTree getMin()
```

הפונקציה מחזירה מצביע לעץ עם הערך מינימלי הנמצא בערימה הבינומית.
המצביע לערך זה שמור בתור שדה במחלקת הערימה הבינומית. הפונקציה מחזירה מצביע זה בזמן הפעלתה ולכן הסיבוכיות שלה הינה O(1).

מדידות

מספר איברים	מספר linking ממוצע למערך לא ממוין	מספר linking ממוצע למערך ממוין	מספר linking ממוצע למערך ממוין הפוך
10,000	107663	17555	25160
20,000	235342	35294	55041
30,000	368685	56646	75590
40,000	510753	75990	110201
50,000	653831	89452	129391
60,000	797049	108310	172096
70,000	946029	135198	181250
80,000	1101238	155977	229580
90,000	1255091	169374	226392
100,000	1407525	188859	274652

אפשר לראות שעבור מערך לא ממויין מספר פעולות ה linking הוא O(nLog n). דבר זה הגיוני, מכיוון שסיבוכיות זמן הריצה שלdelete_min אמורטייזד היא O(log n) (מבוצעות n מחיקות במהלך המיון), ופעולת הlinking (שזמנה קבוע בכל פעם) היא הפעולה המשמעותית בפונקציה. כלומר, מדידה זו מאשרת שסיבוכיות זמן הריצה אמורטייזד של delete_min היא O(log n).

בנוסף, מהמדידות נראה שמיון מערך ממויין או ממויין בסדר הפוך שניהם עולים זמן לינארי, כאשר הקבוע של הממויין נמוך יותר. כלומר, אסימפטוטית הסיבוכיות של שניהם זהה ועדיפה על מערך אקראי, אבל למערך ממויין יש יתרון של קבוע.

העלות הדומה של מערך ממיון וממויין בסדר הפוך נובעת מכך שהמבנה הפנימי של העצים הבינומים הנוצרים דומה. כלומר, בגלל שבעת איחוד שני עצים העץ עם המפתח המינמלי בשורש יהיה השורש החדש, היפוך הסדר לא משנה יותר מדי (כל עוד בחלוקה לזוגות אותם עצים יהיו ביחד). ניתן לראות ע"י מדידות נוספות שאם ממיינים מערך (ללא חזרות) שגודלו חזקה של 2 פלוס 1, מספר הlinking במערך ממויין וממויין בסדר הפוך זהה. זאת מכיוון שלאחר מחיקת המינימום הראשון, נבנה עץ בינומי יחיד (מכיוון שמספר האיברים הוא חזקה של 2), ועץ זה זהה בשני המצבים. כאשר מספר האיברים הוא לא מהצורה הזו, הערימות שיווצרו לא יהיו זהות (בגלל שחלוקת האברים בין העצים תהיה שונה בכל ערימה) אבל ההגיון הכללי שבמבנה לא ישתנה יותר מדי- ומהמדידות עולה שלא ישפיע יותר מדי על זמן הריצה.