**מבנה נתונים – תרגיל מעשי 2**

מגישים: יהב בן יעקב 305170987, ניר אנדי 205686397

**תיעוד הפונקציות:** (הסיבוכיות המוצגת היא worst-case)

**מחלקת FibonacciHeap:**

שדות:

* totalLinks – מספר ה-links שבוצעו בסך הכל.
* totalCuts – מספר ה-cuts שבוצעו בסך הכל.
* markedNodesCount – מספר הצמתים המסומנים בערימה.
* sizes – מספר הצמתים בערימה.
* treesCount – מספר העצים בערימה.
* first – מצביע לאיבר שרירותי ברשימת העצים של הערימה (יכול להשתנות כשמשתנה מבנה העץ).

מתודות:

* empty – מחזירה true אמ"מ הערימה ריקה. סיבוכיות: O(1)
* treesCount – מחזירה את מספר העצים בערימה. סיבוכיות: O(1)
* last – מחזירה את האיבר האחרון ברשימת העצים של הערימה, ביחס לעץ ששורשו first. סיבוכיות: O(1)
* Insert – מכניס צומת חדש לרשימה, בעל מפתח key. סיבוכיות: O(1)
* insertNode – מקבלת צומת כפרמטר ומכניסה אותו לרשימה. מנתק את הצומת ממשפחתו הקודמת (אחים/אבא) במידת הצורך (במחיקות משתמשים בפונקציה זו כדי להעביר תתי עצים בחזרה לרשימה). סיבוכיות: O(1)
* resetTree – מכניסה ערכי ברירת מחד לשדות של העץ (כאשר הוא מתרוקן). סיבוכיות: O(1)
* deleteMin – מוחקת את האיבר המינימלי מהעץ ובונה אותו מחדש. סיבוכיות: O(n), אמורטייזד:O(logn)
* add – מוסיפה צומת חדש לרשימת העצים בערימה. סיבוכיות: O(1)
* consolidating – בונה את הערימה מחדש בעזרת successive linking, בסיום התהליך הערימה תכיל עד log(n+1) עצים, 1 מכל דרגה. סיבוכיות: O(n), אמורטייזד:O(logn)
* updateMin – מוצא מינימום בערימה לאחר בניית העץ מחדש בעזרת הפונקציה consolidating, מוחק marks ופוינטרים ישנים ל-parent במהלך הסריקה. סיבוכיות: O(logn)
* updateTreesCount – סופרת את העצים בערימה ומעדכנת את השדה. סיבוכיות: O(logn)
* getTreesListIterator – מחזירה איטרטור שרץ על העצים (השורשים) בערימה. סיבוכיות: O(n)
* findMin – מחזיר את הצומת שמחזיק במפתח המינימלי.
* meld – ממזג את הערימה עם ערימה נוספת. סיבוכיות: O(1)
* size – מחזיר את מספר הצמתים בערימה. סיבוכיות: O(1)
* getPossibleMaxRank – מחזיר את הדרגה המקסימלית של איבר בעץ. סיבוכיות: O(1)
* countersRep – מחזיר מערך של מונים, כך שבמקום ה-i נמצא מספר העצים שהם בדרגה i. סיבוכיות: O(n)
* delete – מוחקת איבר מהערימה. סיבוכיות: O(n), אמורטייזד:O(logn)
* decreaseKey – מקבלת איבר בערימה ומקטינה את מפתח שלו בדלתא. סיבוכיות: O(n) (במקרה של שרוך למשל), אמורטייזד:O(1)
* checkAndChangeMin – מקבלת צומת ומשווה אותו למינימום, אם הוא קטן יותר, הופכת אותו למינימום החדש. סיבוכיות: O(1)
* potential – מחזירה את הפוטנציאל של מבנה הנתונים. סיבוכיות: O(1)
* totalLinks – מתודה סטטית, מחזירה את מספר ה-links שבוצעו בסך הכל בכל הערימות יחד. סיבוכיות: O(1)
* totalCuts – מתודה סטטית, מחזירה את מספר ה-cuts שבוצעו בסך הכל בכל הערימות יחד. סיבוכיות: O(1)
* link – מתודה סטטית, מבצעת פעולת link בין שני עצים. מחזירה את שורש העץ שנוצר. סיבוכיות: O(1)

**מחלקת HeapNode:**

שדות:

* rank – דרגת הצומת (מספר הילדים).
* mark – true אם הצומת מסומן.
* parent – האבא של הצומת, null אם הוא שורש.
* firstChild – מצביע לאחד הילדים (לא קבוע), null אם אין כאלו.
* next – מחזיר את הצומת הבא ברשימה שהצומת שייך אליה.
* prev – מחזיר את הצומת הקודם ברשימה שהצומת שייך אליה.
* key – המפתח של הצומת.

מתודות:

* HeapNode – בנאי, מייצר צומת חדש בעל מפתח key.
* increaseRank – מעלה את הדרגה של הצומת ב-1. סיבוכיות: O(1)
* decreaseRank – מקטין את הדרגה של הצומת ב-1. סיבוכיות: O(1)
* isOnlyChild – מחזירה true אמ"מ לצומת אין אחים. סיבוכיות: O(1)
* isFirstChild – מחזירה true אמ"מ הצומת היא ה-firstChild של אבא שלה. סיבוכיות: O(1)
* isMarked – מחזירה true אמ"מ הצומת מסומן. סיבוכיות: O(1)
* markNode – מסמנת את הצומת (מעדכנת את השדה mark).
* unMarkNode – מורידה סימון מצומת. סיבוכיות: O(1)
* deacreaseKeyBy – מקטין את המפתח של הצומת ב-delta. סיבוכיות: O(1)
* isLeaf – מחזירה true אמ"מ לצומת אין ילדים. סיבוכיות: O(1)
* isRoot – מחזירה true אמ"מ הצומת הוא שורש. סיבוכיות: O(1)
* disconnect – מנתקת את הצומת מהאחים ומאבא שלה. נקראת כאשר צומת פנימי מוכנס מחדש לערימה (במהלך בנייה מחדש של הערימה). סיבוכיות: O(1)
* getKey – מחזירה את המפתח של הצומת. סיבוכיות: O(1)
* removeSiblingRelations – מוציאה צומת מהרשימה שבא הוא נמצא. סיבוכיות: O(1)
* setFirstChild – מקבלת צומת והופכת אותו ל-firstChild של הצומת הנוכחי. סיבוכיות: O(1)
* setNext – מקבלת צומת והופכת אותו לאח ימני של הצומת הנוכחי. סיבוכיות: O(1)
* getSiblingsIterator – מחזירה איטרטור שרץ על הצומת הנוכחי ועל כל האחים שלו. סיבוכיות: O(n)
* getChildIterator – מחזירה אירטור שרץ על הילדים של הצומת הנוכחי. סיבוכיות: O(n)

**מדידות:**

סדרת M פעולות insert:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Potential** | **totalCuts** | **totalLinks** | **Run-Time (ms)** | **m** |
| 1000 | 0 | 0 | 0.673854 | 1000 |
| 2000 | 0 | 0 | 0.762897 | 2000 |
| 3000 | 0 | 0 | 2.604518 | 3000 |

זמן הריצה האסימפטוטי של סדרת פעולות זו הוא O(m) ב-w.c. ובפרט זהו זמן הריצה האמורטייזד - O(1) בממוצע.

מספר הפעולות ה-link וה-cuts הוא 0 (בכל סדרה של פעולות insert בלבד). הפוטנציאל של המבנה הוא O(m), כמספר פעולות ה-insert, כי בסוף התהליך יש m עצים (בגודל 1).

הפוטנציאל שווה למספר העצים, ששווה למספר ה-insert - מכיוון שלא בוצעה פעולת מחיקה, לא בוצעה פעולת ה-consolidating ולכן כל העצים נותרו בדרגה 0.

סדרת M פעולות insert ואחריה פעולות delete-min:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Potential** | **totalCuts** | **totalLinks** | **Run-Time (ms)** | **m** |
| 6 | 0 | 991 | 0.614956 | 1000 |
| 6 | 0 | 1990 | 1.715475 | 2000 |
| 7 | 0 | 2990 | 2.645793 | 3000 |

זמן הריצה האסימפטוטי של סדרת פעולות זו הוא O(mlogm) – פעולות ה-insert הן O(m). פעולת delete min היא O(logm) לפעולה באמורטייזד – יש פעולות delete-min, וכל אחת עולה O(logm) (באמורטייזד), ולכן בסה"כ .

מספר פעולות ה-cut הוא 0 כי לא מתבצעות פעולות decrease key. מספר פעולות ה-links הוא O(m) באמורטייזד ו-O(1) בממוצע לפעולה – וזה מתיישב עם מה שלמדנו בהרצאה.

הפוטנציאל של המבנה הוא O(logm) – הפעולה האחרונה שמתבצעת היא delete-min, לאחר פעולה זו העץ נבנה מחדש כך שיש לכל היותר עץ אחד מכל דרגה, והדרגה המקסימלית היא O(logm).