**תרגיל בית 2 מבני נתונים**

**id1: 322520255**

**name1: Itamar Ben Nun**

**username1: itamarbennun**

**id2: 316061787**

**name2: Tal Malka**

**username2: talmalka2**

**חלק מעשי**

**מחלקת HeapNode**

מחלקה זו מייצגת איבר בערימת פיבונאצ'י. מחלקה מקוננת בתוך FibonacciHeap.

שדות מחלקה

* key: המפתח של האיבר.
* info: המידע שמחזיק האיבר.
* child: מצביע לאחד הבנים של האיבר.
* next: מצביע לאח הבא של האיבר.
* prev: מצביע לאח הקודם של האיבר.
* parent: מצביע לאב האיבר.
* rank: מספר הבנים של האיבר.
* mark: האם האיבר מסומן ככזה שאחד הבנים שלו נחתך ממנו.

**מחלקת FibonacciHeap**

מחלקה זו מיישמת ערימת פיבונאצ'י.

שדות מחלקה

* start: מצביע לתחילת רשימת העצים.
* min: מצביע לאיבר המינימלי.
* sizeTrees: כמות העצים בערימה.
* size: כמות האיברים בערימה.
* cuts: מספר הפעמים שאיבר נחתך מאביו.
* links: מספר החיבורים שהתבצעו בין שני איברים.

פעולות מחלקה

FibonacciHeap()

בנאי, יוצר ערימה ריקה על ידי אתחול של כל השדות לערכים ההתחלתיים שלהם.

סיבוכיות: WC ו-amortized, הפעולות כולן הן השמות פשוטות.

insert(int key, String info)

מוסיפה צומת חדש לערימה עם מפתח וערך. בפונקציה זו יוצרים צומת חדש עם הפרמטרים שניתנו, ומשתמשים בפונקציה insert\_node כדי להכניס את הצומת המושלם לערימה. מחזירה מצביע אליו.

סיבוכיות: WC ו-amortized, הפעולות כולן הן השמות פשוטות, יחד עם קריאה ל- insert\_ node שפועלת בזמן קבוע כפי שיתואר בהמשך.

insert\_after(HeapNode existing, HeapNode new\_node)

מכניסה צומת חדש אחרי צומת קיים באמצעות שינוי מצביעים.

סיבוכיות: WC ו-amortized, הפעולות כולן הן השמות פשוטות.

findMin()

מחזירה את הצומת עם המפתח המינימלי בערימה.

סיבוכיות: WC ו-amortized, שמור מצביע למינימום.

deleteMin()

מוחקת את הצומת עם המפתח המינימלי על ידי קריאה לפונקציה delete עם הצומת המינימלי. לאחר מכן מבצעת consolidation.

סיבוכיות: WC ו-amortized , במקרה הגרוע כל אחד מן הצמתים בערימה הוא עץ בפני עצמו, אבל מספר העצים לאחר כל consolidation לוגריתמי במספר הצמתים בערימה.

successiveLinking()

מבצעת תהליך של איחוד עצים עם דרגות שוות בערימה (consolidation). כל עץ מאוחסן במערך בהתאם לדרגה שלו, ולאחר מכן מאחדים עצים עם דרגות שוות עד שמתקבלת ערימה חדשה באמצעות הפונקציה buckets\_to\_heap.

סיבוכיות: WC ו-amortized , זהה ל-deleteMin. יחד עם זאת קוראת ל- buckets\_to\_heap שפעולת באותו זמן.

buckets\_to\_heap(HeapNode[] buckets)

יוצרת ערימה חדשה ממערך של עצים (buckets). עוברת על כל מערך ה-buckets ומכניסה כל עץ לערימה החדשה.

סיבוכיות: WC ו-amortized , ניתוח זהה עבור successiveLinking.

insert\_node(HeapNode node)

מכניסה את הצומת המושלם שהתקבל כפרמטר לערימה בצורה "עצלה", מכסה מקרי קצה של ערימה ריקה ומבצעת את ההכנסה באמצעות insert\_after.

סיבוכיות: WC ו-amortized, הפעולות כולן הן השמות פשוטות, יחד עם קריאה ל- insert\_ after שפועלת בזמן קבוע.

decreaseKey(HeapNode x, int diff)

מפחיתה ממפתח הצומת שהתקבל כפרמטר את diff. אם כתוצאה מכך הופר כלל הערימה, מתבצע תהליך של cascading cut כפי שמתואר בהמשך.

סיבוכיות: WC ו-amortized , מלבד השמות, קוראת ל-cascading\_cut שפעולת בסיבוכיות הנ"ל כפי שיתואר בהמשך.

cut(HeapNode node, HeapNode parent)

מנתקת צומת מהאב שלו ומוסיפה אותו לשורשי הערימה באמצעות insert\_after.

סיבוכיות: WC ו-amortized, הפעולות כולן הן השמות פשוטות, יחד עם קריאה ל- insert\_ after שפועלת בזמן קבוע.

cascading\_cut(HeapNode node, HeapNode parent)

מבצעת ניתוקים חוזרים של צמתים מהאבות שלהם באופן רקורסיבי באמצעות cut במידה והם מסומנים, עד שמגיעים לצומת שאינו מסומן או לשורש.

סיבוכיות: WC ו-amortized, כפי שהוכחנו בכיתה באמצעות פונקציית פוטנציאל שהיא מספר הצמתים המסומנים.

delete(HeapNode x)

מוחקת צומת מסוים מהערימה באמצעות מצביע אליו. אם הצומת אינו המינימום, מקטינים את המפתח שלו למינימום האפשרי בעזרת decreaseKey ואז מוחקים אותו. אם לצומת יש ילדים, הם נוספים לשורשי הערימה. מבצעת consolidation רק במידה והתבקשנו למחוק את המינימום מלכתחילה דרך deleteMin.

סיבוכיות: WC ו-amortized , זהה ל-deleteMin.

totalLinks()

מחזירה את מספר החיבורים שהתבצעו בין שני איברים בערימה.

סיבוכיות: WC ו-amortized, הערך שמור בערימה.

totalCuts()

מחזירה את מספר החיתוכים שהתבצעו בין איבר לאביו בערימה.

סיבוכיות: WC ו-amortized, הערך שמור בערימה.

meld(FibonacciHeap heap2)

ממזגת שתי ערימות בצורה "עצלה" על ידי שרשור רשימות שורשי העצים שלהם, הופכת את הערימה השנייה ללא שמישה.

סיבוכיות: WC ו-amortized, הפעולות כולן הן השמות פשוטות.

size()

מחזירה את מספר האיברים בערימה.

סיבוכיות: WC ו-amortized, הערך שמור בערימה.

numTrees()

מחזירה את מספר העצים בערימה.

סיבוכיות: WC ו-amortized, הערך שמור בערימה.

**חלק ניסויי**

ניסוי ראשון

1.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| מס"ד | זמן ריצה (מילי-שניות) | גודל הערימה בסיום | מספר חיבורים | מספר חיתוכים | מספר עצים בסיום |
| 1 | 1.15 | 6559 | 6550 | 0 | 9 |
| 2 | 2.3 | 19681 | 19674 | 0 | 7 |
| 3 | 3.1 | 59047 | 59037 | 0 | 10 |
| 4 | 11 | 177145 | 177133 | 0 | 12 |
| 5 | 72.3 | 531439 | 531427 | 0 | 12 |

2. כפי שניתן להסיק מהטבלה, זמן הריצה התיאורטי של הניסוי הוא .

3. עבור ניסוי זה, אף מדידה לא עשויה להשתנות כתוצאה מסדר ההכנסה, שכן לאחר כלל ההכנסות, ישנה רשימה של כל הצמתים ללא ילדים וטרם היו תיקונים, והמינימום הוא אותו המינימום, אליו יש מצביע, ללא תלות בסדר ההכנסה.

ניסוי שני

1.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| מס"ד | זמן ריצה (מילי-שניות) | גודל הערימה בסיום | מספר חיבורים | מספר חיתוכים | מספר עצים בסיום |
| 1 | 2.35 | 3280 | 39827.2 | 36552.2 | 5 |
| 2 | 8.75 | 9841 | 135207.7 | 125373.7 | 7 |
| 3 | 18.4 | 29524 | 450841.8 | 421325.8 | 8 |
| 4 | 64.1 | 88573 | 1498953 | 1410392 | 12 |
| 5 | 307.05 | 265720 | 4887903 | 4622192 | 9 |

2. כפי שניתן להסיק מהטבלה, זמן הריצה התיאורטי של הניסוי הוא .

3. המדידות שעשויות להשתנות הן מספר החיתוכים, ומספר החיבורים. הסיבה לכך היא שפעולת מחיקת המינימום הוא שגורמת לתיקון הערימה, וסידור הערימה לאחר התיקון יושפע

ניסוי שלישי

1.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| מס"ד | זמן ריצה (מילי-שניות) | גודל הערימה בסיום | מספר חיבורים | מספר חיתוכים | מספר עצים בסיום |
| 1 | 2.1 | 31 | 6550 | 6549.85 | 30.85 |
| 2 | 2.55 | 31 | 19673.95 | 19673.95 | 30.95 |
| 3 | 6.55 | 31 | 59037 | 59037 | 31 |
| 4 | 26.65 | 31 | 177133 | 177133 | 31 |
| 5 | 137.25 | 31 | 531427 | 531427 | 31 |

2. כפי שניתן להסיק מהטבלה, זמן הריצה התיאורטי של הניסוי הוא .

3. המדידות שעשויות להשתנות הן מספר החיתוכים ומספר העצים.

4.