**מסמך תיעוד FibonacciHeap**

**סטודנט 1:** אליאור סגל, 326128006, eliorsegal.

**סטודנט 2:** רועי קישון, 205836810, roeekishon.

המחלקה FibonacciHeap מייצגת מופעים של ערימות פיבונאצ'י. יש בה את תת המחלקה HeapNode. כל מופע של HeapNode מהווה צומת או שורש בערימה FibonacciHeap כלשהי.

תיעוד הפונקציות במימוש:

isEmpty – בודקת אם העץ ריק, O(1).

consolidating – מאחדת את כל העצים בערימה כדי שתהיה ערימה בינומית לא עצלה (עד כדי צמתים עם בן אחד חסר) ומסדרת אותם לפי הדרגות שלהם על ידי קריאה לפונקציה consolidating(arr,node) שמטפלת בהכנסה ואיחוד של עץ ספציפי.O(log n) בממוצע.

cascading\_cut– חותכת צומת מהעץ (בעזרת קריאה לפונקציה cut שחותכת את המצביעים המתאימים לצומת מהצמתים הצמודים לה) ואת ההורה שלה במקרה שהוא מסומן באופן רקורסיבי. O(1) בממוצע ו O(log n) בWC

Insert – מוסיפה מפתח מסוים לערימה ומחזירה אותו. בO(1) זמן כי היא מוסיפה אותו בצורה עצלה לרשימת השורשים.

DeleteMin – מוחקת את הצומת עם המפתח המינימלי. על ידי הוצאותו מהרשימה והכנסת ילדיו וכן קריאה ל consolidating שמעדכנת גם את המינימום סה"כ ב O(log(n) בממוצע.

findMin – מחזירה את הצומת עם המפתח הקטן ביותר (כל עוד קיים אחרת (null.O(1).

meld – מוסיפה לרשימת המפתחות את המפתחות של הערימה השנייה.O(1).

Size – מחזירה את גודל העירמה, O(1)

countersRep – מחזירה מערך מונים כך שהאיבר הi מייצג את מספר העצים בערימה מדרגה i. על ידי מעבר על שורשי העצים וספירת דרגתם. O(log n) בממוצע.

delete – מוחקת את צומת מהערימה על ידי הפיכתו למינימלי על ידי decreaseKey ואז הוצאת המינימלי. O(log n) בממוצע.

decreaseKey – מקטינה את ערכו של המפתח במספר חיובי ובמידת הצורך חותכת אותו בעזרת cascading\_cut O(1) בממוצע ו O(log n) בWC

potential – מחזירה את ערך של פונקציה הפוטנציאל על המצב הנוכחי של העץ בO(1)

totalLinks – פעולה סטטית מחזירה את מספר פעולות החיבור במחלקה מאז תחילת ריצת התוכנית בO(1)

totalCuts – פעולה סטטית מחזירה את מספר פעולות החיתוך במחלקה מאז תחילת ריצת התוכנית בO(1)

kMin – פעולה סטטית המחזירה מערך ממוין של k המפתחות הקטנים בעץ H מבלי לשנות בשום שלב את H. O(k\*deg(H))

\*כלל הפונקציות הקטנות במחלקה HeapNode שיורשת מ- לוקחות O(1), והן פועלות שמעדכנות מצביעים ומחזירות אותם (וכן את המפתחות ומשתנים אחרים).

**תשובות לשאלות**

שאלה 1:

1. O(m) שכן שתי השורות הראשונות לוקחת O(m) זמן. (הfor עושה O(m) פעמים פעולה של O(1) והשורה השנייה עושה O(m) פעולות כי כרגע כל האיברים הם עצים מדרגה 0). והשורה השלישית עושה רק O(log m) פעולות של O(1).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Potential | totalCuts | totalLinks | Run-Time(ms) | m |
| 29 | 10 | 1023 | 0.776 |  |
| 44 | 15 | 32767 | 4.792 |  |
| 59 | 20 | 1048575 | 88.844 |  |
| 74 | 25 | 33554431 | 2964.818 |  |

1. כמות פעולות הlink היא m-1 מכיוון שכאשר עושים deleteMin חצי מהאיברים מתחברים ביחד ואז חצי מהם מתחברים ביחד וכך הלאה ומכיוון שאחרי המחיקה יש בדיוק m איברים (שהוא חוזקה של 2) יש עץ אחד, כלומר:

כמות פעולות הcut היא log m מכיוון שבכל פעם בה אנחנו עושים decreaseKey אנחנו עושים לא עושים זאת על צומת שהיא marked אבל אנחנו כן מקטינים מספיק את המפתח כך שיצטרך להיחתך ולכן אנחנו עושים cut אחת. ומכיוון שאנחנו עושים log m פעמים decreaseKey אנחנו עושים גם log m חתיכות.

הפוטנציאל הוא 3\*log m-1 מכיוון שבסוף התהליך מספר העצים שיש לנו הוא מספר החתיכות שעשינו ועוד 1 מכיוון שבכל חתיכה שביצענו הוספנו עוד עץ על גבי העץ היחיד שהיה לנו מלכתחילה, בנוסף על כך סימנו (marked) צומת אחת בכל חתך חוץ מבחתך שבו השורש של העץ היה האב כלומר marked= (cuts-1) בסך הכל הפוטנציאל הוא:

1. *במקרה זה לא נקטין אף מפתח מספיק כדי שיהיה צורך לחתוך אותו (מכיוון שנתחיל מהשורש וכל פעם נשנה את הבן המתאים אך לא בהפרש גדול מספיק) ולכן לא יהיה אף שינוי על ידי פעולות ה* decreaseKey*כלומר מספר הlinks ישאר אותו הדבר ומספר ה cuts יהיה 0 והפוטנציאל יהיה 1 מכיוון שאף אחד לא סומן כי אף אחד לא נחתך ויש רק עץ אחד.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| decreaseKey max cost | Potential | totalCuts | totalLinks | m |
| (skip) | 1 | 0 | 1023 |  |
| (skip) | 1 | 0 | 32767 |  |
| (skip) | 1 | 0 | 1048575 |  |
| (skip) | 1 | 0 | 33554431 |  |

1. *במקרה זה לא נעשה consolidating בכלל ולכן נשארו m+1 עצים מדרגה 0 כלומר כל האיברים הם שורשים ולכן לא נעשה אף link וכן כל* decreaseKey *רק שינה מפתח של שורש ולכן לא נעשו חתיכות בכלל ולא סומנו צמתים בכלל. לכן הפוטנציאל יהיה*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| decreaseKey max cost | Potential | totalCuts | totalLinks | m |
| (skip) | 1025 | 0 | 0 |  |
| (skip) | 32769 | 0 | 0 |  |
| (skip) | 1048577 | 0 | 0 |  |
| (skip) | 33554433 | 0 | 0 |  |

1. *המצב של הערימה עד לשורה יהיה זהה למצב שבסעיף 3 אך לאחר הפעלת השורה על כל הצמתים שסומנו יופעל cut (מכיוון שהם בן אחד של השני) ולכן מספר הcuts יגדל ב ואילו מספר הצמתים המסומנים יהיה0 כלומר מספר החתכים יהיה והפוטנציאל יהיה מספר העצים שהוא מספר החתיכות ועוד 1 (מכיוון שכל חתך יוצר עץ נוסף ויש את העץ ההתחלתי)*

*בנוסף ה*decreaseKey *היקר ביותר היה האחרון בו כל הצמתים המסומנים נחתכו ולכן עלה כלומר .*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| decreaseKey max cost | Potential | totalCuts | totalLinks | m |
| 9 | 20 | 19 | 1023 |  |
| 14 | 30 | 29 | 32767 |  |
| 19 | 40 | 39 | 1048575 |  |
| 24 | 50 | 49 | 33554431 |  |

*שאלה 2:*



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Potential | totalCuts | totalLinks | Run-Time(ms) | m |
| 4 | 0 | 723 | 1.599 |  |
| 6 | 0 | 6555 | 3.240 |  |
| 8 | 0 | 59040 | 23.124 |  |
| 10 | 0 | 531431 | 136.755 |  |
| 10 | 0 | 4782955 | 1404.868 | 4782968 |

1. *פעולה ממוצעת של deleteMin לוקחת O(log n) כאשר n זהו מספר האיברים בערימה ומכיוון שבfor השני בכל איטרציה כמות האיברים היא O(m) ועושים את הfor O(m) פעמים זמן הריצה שלו הוא O(m\*log m). ואילו הfor הראשון רץ m פעמים ועושה פעולה רק של O(1) לכן בסך הכל זמן הריצה הוא O(m\*log m)*
2. *כמות הcuts היא 0 מכיוון שאנחנו מוחקים רק את המינימליים ולכן רק שורשי עצים ולא צמתים. לכן גם אין צמתים מסומנים ולכן הפוטציאל הוא רק כמות העצים בסוף התהליך.*

*כמות העצים בסוף התהליך תלויה רק בכמות האיברים מכיוון שאנחנו מוחקים רק את המינמלי ולכן נשארים רק עם עצים בינומיים וכמות עצים זו היא לפי חוזקות ה2 שנכנסות כדי לבנות את מספר האיברים כלומר מספר ה'1' ים בייצוג הבינארי של*

*ומספר הלינקים הוא (או כמעט בדיוק) m פחות הפוטנציאל מכיוון שכל עץ נוסף שקיים ולא מחובר לעץ הראשי אומר שהשורש שלא לא התחבר לשום דבר.*