Eden Pollinger 318688991 עבודה מעשית 2 במבני נתונים 318895877 Ron Mondshein
pollinger **מימוש ערימת פיבונאצ'י** **ronmondshein**

<u>תיעוד המתודות במחלקות:</u>

<u>המחלקה HeapNode:</u>

המחלקה מממשת את המנשק HeapNode ומכילה את השדות הבאים:

Key - הערך של הצומת.

Rank - מספר הילדים ברשימת הילדים של הצומת.

Parent - מצביע להורה של הצומת.

Child - מצביע לאחד מילדי הצומת, לא משנה לאיזה.

Next - מצביע לצומת הבא, אם אין כזה אז לעצמו.

Prev - מצביע לצומת הקודם, אם אין כזה אז לעצמו.

Mark - true אם הצומת כבר איבד ילד, false אחרת.

cloneMe - משתנה שמחזיק שכפול של הצומת, עבור פונקצית kMin.

במחלקה יש שתי פונקציות בלבד:

1. getKey() – מחזירה את הערך של הצומת ב(O(1.
2. cloneNode(HeapNode node) - מעדכנת את שדה הcloneMe של הצומת עליו הופעלה הפונקציה להיות הצומת שהתקבל ב(O(1.

<u>המחלקה FibonacciHeap:</u>

המחלקה מממשת את המנשק FibonacciHeap ומכילה את השדות הבאים:

Min - מצביע עבור הצומת המינימלית.

First - מצביע עבור הצומת הראשון שהוכנס, לפי הגדרות השאלה.

totalMarked - מספר הכולל של הצמתים המסומנים.

totalCuts - מספר החיתוכים הכולל מרגע הרצת התוכנית.

totalLinks - מספר הקישורים הכולל מרגע הרצת התוכנית.

totalTrees - מספר העצים הכולל, נספר לפי מספר השורשים.

Size - כמות הצמתים הכוללת בערימה.

המתודות במחלקה:

1. insert(int key) - הפונקציה מקבל ערך מסוג int אשר מייצג את הkey של הצומת שאנחנו רוצים לבנות ומחזירה את הצומת שהוכנס.
   - ראשית, הפונקציה בונה עצם חדש מסוג heapNode עם הערך הנתון.
   - נעזרת בפונקציית העזר insertNode שתתואר בהמשך.
   - הפונקציה מגדילה את העץ להיות הגודל הקודם ועוד האיבר שנוסף.
   - **סיבוכיות** -O(1) רק מוסיפה צומת לרשימת העצים מבלי לעשות שינויים נוספים.
2. isEmpty() - פונקציה שמחזירה Boolean שמציין האם האיבר המינימלי הוא ריק, ז"א כל העץ ריק.
   - **סיבוכיות** - O(1).
3. getFirst() - פונקציה שמחזירה הצומת הראשון בערימה.
   - **סיבוכיות** -O(1) מחזירה את המצביע הרלוונטי.
4. getMin() - פונקציה שמחזירה int שמציין את ערכו של האיבר המינימלי. אם לא קיים מחזירה 0. משתמשת בפונקציה שמוצאת את האיבר המינימלי.
   - **סיבוכיות** -O(1) מחזירה את המצביע הרלוונטי.
5. findMin() - פונקציה שמחזירה את האיבר המינימלי מסוג heapNode. אם העץ ריק מחזירה null.
   - **סיבוכיות** - O(1) מחזירה את המצביע הרלוונטי.
6. deleteMin() - פונקציה שמוחקת את האיבר המינימלי בערמת הפיבונצ'י.
   - אם העץ בגודל אחד מוחקת אותו ומסיימת.

   אם לאיבר המינימלי יש לפחות ילד אחד, אז, ראשית נשנה את המצביע לאיבר הראשון להיות המצביע לבן. לאחר מכן, אם יש יותר מילד אחד, נחבר את כולם במקום של האבא שלהם כפי שהוגדר במטלה. לאחר מכן ננתק את כולם מהאבא ובכך שהפכנו אותם לשורשים נבטל את הסימון שלהם (שורשים לעולם לא מסומנים!).
   - אחרת, אם לשורש אין ילדים, נתקן את ההצבעה לאיבר הראשון להיות למינימום ונחבר בין הקודם לעוקב של הצומת הנמחקת.
   - נקרא. לפונקציית Consolidate שתארגן מחדש את העץ לאחר המחיקה, נקטין את גודל העץ ב1 ונסיים.

- **סיבוכיות-**amortized(logn), WC(n) . ע"פ סיבוכיות פונקציית העזר בהנחה ששאר הפונקציות שפועלות עובדות בסיבוכיות של O(1), כי מדובר בפעולות כמו שינוי מצביעים ותנאים.

7. Delete(HeapNode x)- פונקציה שמוחקת את האיבר הx מהערמה. פועלת בהנחה שאכן יש כזה בערמה. נעזרת ב2 פעולות עזר: ראשית, מפעילה על הצומת decreaseKey אם הערך המינימלי הכי קטן שקיים לטיפוס integer בג'וואה ולאחר מכן בידיעה שמי שאנחנו רוצים למחוק הוא בעל הערך הכי קטן, מוחקת את המינימום בעזרת הפונקציה deleteMin.
   - **סיבוכיות-** זהה ל2 הפונקציות ממנה היא מורכבת, בפרט זהה ל deleteMin שלה יש את הסיבוכיות הגבוהה מבין ה2 ולכן אמורטיז O(logn), WC O(n)
8. nonMarked()- מחזירה את מספר הצמתים שלא מסומנים מסוג integer. מחשבת זאת על ידי חישוב גודל העץ פחות כמה צמתים מסומנים.
   - **סיבוכיות-**O(1) .
9. potential () - מחזירה את פוטנציאל הערמה מסוג integer. מחשבת זאת על ידי חיבור בין מספר העצים הכולל לבין פעמיים מספר הצמתים המסומנות כמוגדר בשאלה.
   - **סיבוכיות-**O(1) .
10. totalLinks () - מחזירה את מספר הקישורים הכוללים בערמה מסוג integer. פונקציה סטטית. קישור היא פעולה שנעשת בזמן ריצת התוכנית ברגע שמחברים בין 2 עצים בעלי אותה דרגה. לאחר שנחבר אותם נקבל עץ מדרגה אחת יותר ממה שהיה לכל אחד מהם על ידי כך שנתלה את העץ ששורשו בעל הערך הגדול יותר על העץ ששורשו בעל הערך הקטן יותר.
    - **סיבוכיות-**O(1) .
11. totalCuts () - מחזירה את מספר החיתוכים הכוללים שנעשו בערמה מסוג integer. פונקציה סטטית. כל החיתוכים שנעשו בזמן ריצת התוכנית. פעולת חיתוך היא פעולה שמנתקת תת עץ מההורה שלו. נעשה במהלך decreaseKey/delete.
    - **סיבוכיות-**O(1) .
12. Size()- מחזירה integer שמייצג את מספר האיברים ברשימה.
    - **סיבוכיות-**O(1) .

13. insertNode(HeapNode node)- פעולה שמטרתה לקבל איבר מטיפוס צומת ולהכניסו לערמה. אם הערמה ריקה, מכניסה אותו ומעדכנת את המינימום ואת הראשון בהתאם. אם לא, מכניסה אותו אחרי האיבר הראשון ובודקת אם הוא האיבר המינימלי, אם כן מעדכנת את המינימום בהתאם.
בנוסף הפונקציה מעדכנת את מספר השורשי עצים הכולל ומחזירה את האיבר.
    - **סיבוכיות-**O(1) .
14. Consolidate()- פעולה שמטרתה לארגן מחדש את הערמה אחרי שאיבר מוסר ממנה. פונקציית עזר לפונקציית המחק מינימום.
תחילה מייצרים מערך בגודל לוג על בסיס 2 של גודל הערמה, ניקרא לו "מערך דליים".
לאחר מכאן מייצרים רשימת שורשים.
כעת עוברים על רשימת השורשים, עבור כל שורש אם קיים איבר במערך הדליים בעל אותה דרגה נראה מי המינימלי ומי המקסימלי מבניהם.
נבצע קישור כך שהמינימלי יהיה האבא של המקסימלי כדי לשמור על חוק הערמה. תמשיך ככה עד שנחבר כל 2 שורשים בעלי אותה דרגה ונייצר עץ בינומי אחד העונה לכלל הערמה (ערך הורה קטן מערך ילד).
    - **סיבוכיות-**amortized(logn), WC(n)
    במקרה הגרוע נצטרך לחבר איבר-איבר לכדי ערמה בינומית כך שסך כל הצמתים הוא n.
    באמורטיז יהיו לנו כבר תתי עצים בינומים מחוברים שנצטרך רק לחבר ביניהם.

15. Meld(FibonacciHeap heap2)- הפונקציה מחברת בין העץ הקיים לעץ נוסף שמועבר אליו. אם העץ הקיים ריק אז נגדיר אותו כעץ השני וסיימנו. אם הוא לא ריק נקשר בין העצים בעזרת המצביע לאיבר הראשון ונמצא את המינימום מביניהם.
נעדכן את כמות העצים, החיבורים, החיתוכים, הסימונים והגבוה על ידי חיבור של מה שנצבר אצל הערמה החדשה לבין מה שנצטבר בערמה הקיימת.
    - **סיבוכיות -**amortizedO(1), WC(1) כוללת העברת משתנים ופעולות חיבור בסיסיות.
16. countersRep()- פעולה שמטרתה להחזיר מערך שבכל תא מכיל את מספר העצים מסדר מסויים בערמה.

נייצר מערך בגודל הדרגה המקסימלית. אם העץ ריק, נחזיר מערך ריק. אחרת, נעבור בלולאה על כל הצמתים ונעדכן את המערך לפי הדרגה של אותו העץ. נחזיר את המערך כך שבסופו לא יהיו אפסים.

- **סיבוכיות -**O(logn) עוברים על מערך עד גודל הדרגה המקסימלית שהיא logn.

17. decreaseKey(HeapNode x, int delta)- פונקציה שמטרתה להוריד את ערכו של האיבר שנמסר לה בערך המספר אותו היא מקבלת. הפונקציה משנה את ערך האיבר ולאחר מכן מבצעת חיתוכים בהתאם לאיבר ולאביו בעזרת פעולות ה cut והcascadingcut להורה. אם יש צורך מעדכנת את המינימום בהתאם.

- **סיבוכיות-**amortized(1), WC(n)

באמורטיז לא נצטרך לבצע פעולות שינוי רבות עקב שינוי הערך אך במקרה הגרוע נצטרך לבצע עד n חיתוכים בעקבות פעולה זו.

18. CascadingCut(HeapNode(x))- פעולת עזר שמטרת לנתק צומת מהוריו. אם הוא לא מסומן היא מסמנת אותו ומעלה את מספר הסימונים. אם הוא כן מסומן מבצעת מחיקה ברקורסיה וחתיכה של האיבר עצמו.

- **סיבוכיות-**amortized(1), WC(logn)

במקרה הגרוע נצטרך לבצע חתיכות עד לשורש, באמורטיז נעשה מספר סופי של חיתוכים.

19. Cut(HeapNodex)- פעולת עזר שחותכת איבר מהוריו ומבצעת סימון שהוא נמחק, משנה את ההפניות מהאיבר הקודם אליו לבא אחריו בהתאם (אם יש). מורידה מהדרגה של ההורה את האיבר שנמחק ומעדכנת את מספר החיתוכים הכללי.

- **סיבוכיות-**O(1) .

מבצעת שינויי הפניות ופעולות סכימה.

20. KMin(FibonacciHeap H, int k)- פונקציה סטטית שמטרתה להחזיר מערך שמכיל את k האיברים הכי קטנים בערמה. הפעולה מתבצעת מבלי לשנות את H. אם העץ ריק או קיי שווה לאפס נחזיר מערך ריק.

אחרת, נשכפל את איבר המינימום, כדי לא להרוס את המקור כפי שנדרש בשאלה, ונכניס אותו לערמה חדשה. נבנה מערך בגודל k ונעבור עליו, ראשית נכניס את האיבר המינימלי בערמה החדשה (תחילה יש שם רק איבר אחד) ונמחוק אותו מהערמה. נכניס את כל ילדיו לערמה במקומו ונמצא את המינימום

החדש מביניהם. נחזור בלולאה, נמצא את המינימום מהערמת עזר, נכניס למערך, נמחק אותו וחוזר חלילה עד שנגיד למערך מלא עם k איברים.

- **סיבוכיות** ((O(k*deg(H בשל השימוש בערמת עזר נוכל למצוא מינימום בסיבוכיות של ערמת פיבונאצ'י.

21. Link(HeapNode x, HeapNode y)- פונקציית עזר שמקשרת בין 2 צמתים כך שאחד הופך להיות הבן של השני.

- ראשית נקטין את מספר העצים. אם לy יש מצביע לאיבר הראשון נעביר אותו לבה אחריו.
- אם לx אין ילדים נחבר את y להיות בנו היחיד.
- אחרת, נחבר אותו להיות הchild שלו ונחבר אותו לאחיו.
- נוסיף אחד למספר הלינקים הכולל.
- **סיבוכיות-**O(1) . מדובר בשינוי מצביעים.

Eden Pollinger 318688991 עבודה מעשית 2 במבני נתונים 318895877 Ron Mondshein
pollinger **מימוש ערימת פיבונאצ'י** **ronmondshein**

<u>החלק התאורטי:</u>

<u>שאלה 1</u>

סעיף א'

ננתח שורה-שורה את זמן הריצה האסימפטומטי של סדרת הפעולות כפונקציה של m.

שורה 1-

אנחנו מבצעים m פעולות *insert(k)*. עבור ערמת פיבונאצ'י אנו יודעים כי הסיבוכיות של פעולת *insert* היא *O(1)* . זאת משום שכאשר אנו מכניסים איבר חדש לערמת פיבונאצ'י אנחנו יוצרים תת עץ חדש שהוא בעצם שורש חדש שנוסף לערמה. מספר השורשים שניתן להוסיף אינו מוגבל. לכן יש לנו סיבוכיות *O(1)* קבועה. אנו מבצעים את הפעולה הזאת *m* פעמים ולכן סה"כ הסיבוכיות של השורה הזאת הינה ***O(m)*** = *m*O(1)*.

שורה 2-

אנו מבצעים פעולה אחת של *deleteMin()* . לכאורה פעולה זו אמורה לעלות *O(logn)* במקרה הממוצע. אך ניתן להבחין כי אנו מסתכלים בקטע זה על המקרה הגרוע. במקרה זה הכנסנו *m +1* איברים, אחד אחרי השני, מבלי לעשות אף פעולה בין ההכנסות. לכן קיבלנו סה"כ *m+1* איברים שמהווים *m+1* שורשים מדרגה 0 שמרכיבים את ערמת הפיבונאצ'י. לאחר המחיקה ישארו *m* שורשים כאלו. במקרה זה פעולות ה *consolidate* שהיא מהווה פונקציית עזר משמעותית בתוך ה *deleteMin* צריכה לאחד *m* שורשים לכדי ערמה מאוחדת בגובה $\log_2 m$ מתוך *m* ערמות מדרגה 0 שהיו לפני כן. לכן הסיבוכיות שלה הינה *O(m)* ולא *O(logm)* . שאר הפעולות שמבוצעות בתוך ה *deleteMin* מתבצעות בסיבוכיות הקטנה משמעותית מזו של *consolidate* ולכן סיבוכיות שורה זו הינה ***O(m)***.

שורה 3-

אנחנו רצים בלולאה על *logm* איברים ומבצעים את הפעולה $decreaseKey(m-2^i+1, m+1)$ בכל איטרציה. פעולה זו מפחיתה את הערך במקום המתאים ב *m+1* . בכל איטרציה אנחנו מבצעים את פעולות העזר *cascadingCut* ופעולות נוספות שפועלות ב *O(1)* . *cascadingCut* , הפונקציה המשמעותית, מבצעת חיתוכים וסימונים בעץ לאחר השינוי בערך המפתח. הלולאה רצה כך שנבצע בכל אטרציה, פרט לראשונה, הפחתה מצומת כך שההורה שלו כבר עבר הפחתה באותו מספר. מה שאומר שהיחס סדר ביניהם יישאר זהה ומבנה העץ יישאר תקין ולא יתבצעו חיתוכים כלל. אם כך, עלות הפעולה היא *O(1)*, כעלות כמות החיתוכים, שמתבצעת *logm* פעמים. לכן בסה"כ סיבוכיות חלק זה הינו ***O(logm)*** = *logm*O(1)* .

סה"כ סיבוכיות לכל הקטע הינו: ***O(m)*** =*O(m)+O(m)+O(logm)*

Eden Pollinger 318688991 עבודה מעשית 2 במבני נתונים 318895877 Ron Mondshein
pollinger **מימוש ערימת פיבונאצ'י** **ronmondshein**

סעיף ב'

m	Run-Time (ms)	totalLinks	totalCuts	Potential
2^5	0.135383	26	0	5
2^{10}	0.78567	1013	0	10
2^{15}	8.8676	32752	0	15
2^{20}	127.081973	1048555	0	20

סעיף ג'

מספר פעולות link-

אנחנו מבצעים מעבר בין רשימה שורשים בעלי דרגה 0 לערמה הכוללת את כל אותם שורשים. לשם כך אם יש לנו בהתחלה m+1 הכנסות נבצע תחילה חיבור בין כל זוג בעל דרגה 0 וניצור ממנו עץ מדרגה 1, כעת יש לנו m/2 שורשים. לאחר מכן נחבר את אותם שורשים מדרגה 1 לזוגות וניצור עץ מדרגה 3, כעת יש לנו m/4 שורשים. נמשיך הלאה עד שנחבר יחדיו את כל שורשיו הערמה שהוכנסו בהתחלה לכדי ערמה אחת. בסה"כ יש לנו m שורשים כי מחקנו 1, את המינימום.

$$\sum_{i=1}^{logm} \frac{m}{2^i} = m \sum_{i=0}^{logm} \frac{1}{2^i} = m * \frac{\frac{1}{2}\left(\left(\frac{1}{2}\right)^{logm} - 1\right)}{-\frac{1}{2}} = -m\left(\left(\frac{1}{2}\right)^{logm} - 1\right) = m\left(1 - 2^{-logm}\right)$$

$$= m\left(1 - 2^{\log\left(\frac{1}{m}\right)}\right) = m\left(1 - \frac{1}{m}\right) = \boldsymbol{m - 1}$$

מספר פעולות cut-

אנחנו מבצעים חתיכה כל פעם שאנחנו מוחקים איבר מצומת מסומן או מוחקים בן של שורש. כפי שהוסבר קודם אנחנו מבצעים בדיוק **אפס חתיכות**. לא תתבצע הפרה בערמה כי ערכו של כל צומת שנפחית ממנו m+1 יופחת רק לאחר שערכו של אביו יופחת, יחס הסדר יישאר ולא נבצע חתיכות נוספות.

מספר פעולות potential-

חישוב הפוטנציאל מתבצע בצורה הבאה potential = #totalTrees + 2 * #totalMarked .

לאחר המחיקה נותרנו עם עץ אחד- m צמתים שמהווים חזקה של 2 שסודרו כערמה אחת. בנוסף, סומנו 0 צמתים כי לא בוצעו חתיכות ולכן בסה"כ ערך הפוטנציאל יהיה 1.

(*) ניתן לראות כי במימוש שלנו הגענו ל$\log(m)$ צמתים, מה שהוביל לפוטנציאל של$\log(m)$ במקום 1.

case	totalLinks	totalCuts	Potential	decreaseKey Max Cost
(c)original	m-1	0	1	<skip>
(d)	m-1	logm-1	3logm-2	<skip>
(e)	0	0	m+1	<skip>
(f)	m-1	0	1	0

סעיף ד'

בסעיף זה השינוי נעשה לאחר ביצוע מחיקת האיבר המינימלי ולכן מספר הלינקים לא השתנה. אנחנו מפעילים את decreaseKey() במצב שהערמה מכילה עץ בינומי יחיד ותקין בגודל m. ההפחתה הפעם נעשית על העלים ברמה אחת לפני האחרונה.

נסכום את כמות העלים האלו- הם כל הצמתים ברמה זו פחות כל העלים ברמה האחרונה (כי אם יש עלה ברמה האחרונה אז יש לו הורה ברמה שלפניו והורה הוא אינו עלה). בערמה כזו כל עלה יהיה שייך להורה אחר ולכן כל עלה שנחתך ייצג חתיכה אחת וסימון צומת אחת של ההורה שלו, כי לא מדובר בשורש.

הוכח במטלה 4 כי מספר הצמתים בעומק d הוא $\binom{k}{d}$. kמספר דרגה, d עומק. לכן נחשב:

$$\#cut = \binom{logm}{logm-1} - \binom{logm}{logm} = \frac{logm!}{(logm-1)!} - 1 = logm - 1$$

חישוב הפוטנציאל מתבצע בצורה הבאה potential = #totalTrees + 2 * #totalMarked .

ולכן, בהנחה שמספר ה*cut* הוא 1 פחות מספר העצים (מוסיפים את העץ הראשוני) ומספר ה*cut* מייצג גם את מספר העצים המסומנים:

Potential = 1+logm-1+2(logm-1) =**3logm-2***

סעיף ה'

מחקנו את השורה שמוחקת את המינימום. לכן לא נבצע כלל חיבור של עצים. הכנסנו m+1 עצים מדרגה 0, לאחר מכן שינינו להם את הערך. לכן בסה"כ נשאר עם 1+m עצים, לא נחתוך, לא נחבר ולא נסמן אף צומת.

כמובן ש potential = #totalTrees + 2 * #totalMarked=m+1+2*0=m+1.

Eden Pollinger 318688991 עבודה מעשית 2 במבני נתונים 318895877 Ron Mondshein
pollinger **מימוש ערימת פיבונאצ'י** **ronmondshein**

סעיף ו'

ראשית, השורה הנוספת היא הרביעית ולכן מספר החיבורים לא השתנה (שורה 2 לא השתנתה).

נפעיל את הפעולה החדשה שנוספה על עץ בינומי בגודל m . לא יתבצעו חיתוכים כמו במקרה המקורי כי לא שינינו את שלושת השורות הראשונות.

כאשר נרצה להוריד את ערך האיבר ה-m2 בדומה למקרה המקורי, אביו כבר עבר הפחתה ולכן יחס הסדר בעץ בינומי לא יופר ולא נצטרך לחתוך או לסמן בחיתוך או סימון נוסף.

עלות decreaseKey שקולה לעלות של מספר חיתוכים. הוא יגיע לכמות המקסימלית האפשרית שלו אם נפר לאורך כל הערמה את יחס הסדר, זאת נבצע את הפעולה בצורה כזו שנתחיל מהעלה, נוריד את ערכו, נבצע חיתוך, נתקדם חיתוך חיתוך עד לשורש. תרחיש שכזה שקול לחיתוך כל קשת מהעלה עד לשורש בעץ בגובה של logm ולכן שווה ל logm-1 . חיתוך מתבצע לצומת מסומנת לכן המעבר צריך לקראת פעמיים- פעם אחת נעבור ונסמן צמתים ופעם שניה נחתוך.

אם נשנה את שורה 4 להיות m-1 במקום m-2 ואת שורה 3 להיות 2+ במקום +1 נגיע לכזה תרחיש.

בשורה 3 נסמן מסלול של צמתים (נסמן צומת בכל רמה) ובשורה 4 נמחק אותם.

<u>שאלה 2</u>

סעיף א'

case	Run-Time(ms)	totalLinks	totalCuts	Potential
728	1.603511	1642	0	6
6,560	7.902098	12093	0	6
59,048	40.744535	125825	0	9
531,440	323.228378	1280108	0	10
4,782,968	3185.80664	14004299	0	14

סעיף ב'

שורה ראשונה-

Insert(k)- הכנסת איבר לערמת פיבונצ'י עולה O(1). זאת מכיוון שאנחנו רק מוסיפים אותו ומעדכנים את רשימת השורשים ולא מבצעים איזשהו link. לכן עבור m הכנסות ברצף ללא כל פעולה ביניהם הסיבוכיות תהיה O(m) .

שורה שניה-

Delete-Min()- לפני הפעולה הראשונה יש מבנה דמוי רשימה של m שורשים, לאחר הפעולה הראשונה השורשים האלו מתאחדים לכדי ערמה בינומית תקינה. עלות של כל פעולה של מחיקת מינימום בעץ בגודל n הוא O(logn) ואנחנו מוחקים בתחילה מעץ בגודל m עד לכדי עץ בגודל m-3/4m=1/4m. נחשב:

$$\sum_{i=m}^{\frac{1}{4}m} logn = \sum_{i=m}^{1} logn - \sum_{i=\frac{1}{4}m}^{1} logn \leq \sum_{i=m}^{1} logn = mlogm = O(mlogm)$$

זמן ריצה אסימפטוטי כולל:

$$O(m) + O(mlogm) = \boldsymbol{O(mlogm)}$$

סעיף ג'

מספר פעולות חתיכה

גם בעת פעולת ההוספה וגם בעת מחיקת המינימומים במקרה זה לא ימחקו או יסומנו צמתים, לכן מספר החתיכות הכולל יהיה **0**.

מספר פעולות link

יתבצעו פעולות link רק במחיקת המינימום הראשונה. זאת משום שכפי שציינו, תחילה נכניס m איברים אחד אחרי השני ומתכונות ערמת פיבונצ'י לא יקרה איתם כלום הם פשוט יכנסו כמעין רשימת שורשים. לאחר מחיקת המינימום הראשונה יתחילו להתבצע links ויהיה ניסיון לחבר את השורשים לכדי ערמות בינומיות כמה שיותר מלאות לפי כלל הערמה, כל הורה קטן מבניו.

נחשב את מספר החיבורים: (פירוט בשאלה 1 סעיף ג שם נעשה הפירוט והחישוב המלא)

$$links = \sum_{i=1}^{logm} \frac{m}{2^i} = \boldsymbol{m-1}$$

כעת האיברים מסודרים בסדר יורד ולכן נקבל שככל שדרגת העץ גבוהה יותר, כך האיברים בו גדולים יותר. לכן המינימום יכול להימצא כשורש בעץ הקטן ביותר באותו הרגע. מכיוון שנמחק כל פעם שורש בעץ הקטן ביותר לא יהיה

עץ יותר קטן אליו נוכל לעשות לשורש החדש אחרי המחיקה קישור ולכן מספר הקישורים הכולל יהיה כמספר הקישורים אחרי מחיקת המינימום הראשונה.

(*) ניתן לראות כי במימוש שלנו יש טעות כלשהי שגורמת לכך שלכל כמות לינק גדול בבערך פאקטור 2.5 ממה שהוא צריך, לא הצלחנו להתאים את המימוש לתאוריה.

הפוטנציאל

הגדרת הפוטנציאל כפי שצויין כבר קודם הוא: potential = #totalTrees + 2 * #totalMarked.

כפי שציינו, לא יהיו מחיקות וגם לא סימונים לכן ערך זה יהיה 0. לכן במקרה זה potential = #totalTrees.

אם הכנסנו בתחילה m צמתים ומחקנו 3/4m נשאר בסה"כ עם 1/4m צמתים. בערמה בינומית, בכל תת עץ לאחר פעולת מחיקת המינימום יש 2^i צמתים ולכן נוכל להבין כי מספר השורשים הוא חיבור בין חזקות שלמות של 2. ניתן לייצג זאת ויזואלית על ידי הייצוג הבינארי של 1/4m כך שה MSB מייצג את השורש של העץ הגדול ביותר. כל פעם שיש 1 יש תת עץ בגודל זה. לכן הפוטנציאל שווה לכל המקומות בייצוג הבינארי בהם יש 1.