**מגישים:**

יותם צביאלי – ת"ז 209497023

נופר שלמה – ת"ז 208410084

**תיעוד הפונקציות:**

**מחלקה Fibonacci Heap:** מחלקה זו מייצגת ערמת פיבונאצי התומכת בכול הפעולות שהוצגו בכיתה, במחלקה הגדרנו את השדות הבאים: firs, min , marked, rootsNum, size, Static total links, static total cats.

**מחלקה Heap Node:** מחלקה המייצגת צומת בערמה. במחלקה הגדרנו את השדות הבאים: info, rank, key, mark, child, next, prev, parent, original pointer(optional)

כמו כן הגדרנו לשתי המחלקות לכול השדות מתודות get ומתודות set.

deleteMin :

**הסבר:** כדי למחוק את האיבר המינימלי אנו בודקים באיזה מקרה אנחנו נמצאים ומעדכנים בהתאם את השדות הרלוונטיים. אם זהו השורש היחידי ואין לו ילדים אז נקבל ערמה ריקה ונחזיר. אם זהו השורש היחידי ויש לו ילדים אז נרוץ בלולאה על כל הילדים שלו ונעלה אותם לרמת השורש ונמצא את המינימום החדש. נגדיר לכל הילדים של המינימום כי אין להם הורה. במקרה שהמינימום שמחקנו גם היה האיבר הראשון נמצא את האיבר הראשון החדש בעל הדרגה הקטנה ביותר. לאחר מכן נרוץ על כל הילדים בשנית ונבדוק אם הם מסומנים, אם כן נסיר את הסימון. לבסוף נקרא לפונקציית considulation.

**סיבוכיות:** במקרה הגרוע ובמקרה amortized . במקרה הגרוע נקבל שזמן הריצה הוא מכיוון שזה המקרה בו יש רק צמתים בודדים ומחקנו את האיבר המינימלי. כדי למצוא את המינימום החדש נצטרך לעבור על כל השורשים שנותרו וזאת אנו מבצעים בפונקציית ה considulation. בנוסף בפונקציה זו אנו רק נבצע מעבר על כל השורשים כדי להסיר סימון אם היה. (במקרה זה שיש רק צמתים בודדים המעבר לא יהיה רלוונטי אך מכיוון שהוא מתבצע בכל מקרה נקבל ). בתוך הלולאה יש רק השמה לשדות שזה זמן קבוע של עבודה ולכן בסה''כ זמן הריצה בלולאה זו הוא . כפי שנתאר בהמשך, זמן הריצה של פונקציית ה considulation המקרה הגרוע הוא ולכן בסה''כ במקרה הגרוע זמן הריצה יהיה .

זמן הריצה amortized הוא מכיוון שאופן פעולת האלגוריתם הוא כפי שהסברנו קודם זהה לאלגוריתם המתואר בהרצאה ולכן זמן הריצה הוא .

considulation :

**הסבר:**  בפונקציה זו אנו מבצעים את פעולת ה considulatio. כלומר, משתמשים במערך כך שמקום העץ במערך יהיה לפי הדרגה שלו. עוברים בלולאה על כל השורשים ובודקים אם המערך במקום של הדרגה ריק אז מכניסים את העץ ואם לא אז נכנס ללולאה שקוראת לפונקציית connect שמבצעת חיבור של שני עצים בעלי אותה דרגה. הלולאה תמשיך לקרות עד שהגענו לתא שלא נמצא בו עץ ותשים שם את העץ שקיבלנו מ connect. לאחר שעברנו על כל השורשים נעבור בלולאה על המערך שקיבלנו ונקשר את הצמתים בנוסף נבצע עדכון והשמה לשדות הרלוונטיים בלולאה נוספת נפרדת.

**סיבוכיות:** במקרה הגרוע ובמקרה amortized . במקרה הגרוע נקבל n צמתים בודדים מדרגה 0 ולכן נבצע n-1 חיבורים כדי לקבל את הערמה. ההפרש במספר השורשים במקרה הזה הוא n-1 (כלומר, הגדול ביותר) כי בהתחלה היו n שורשים ובסוף מסיימים עם עץ אחד בעל שורש אחד המכיל את כל שאר הצמתים. לולאת ה while החיצונית עוברת על כל השורשים והלולאה הפנימית קוראת לפונקציית connect שרצה בזמן קבוע (מתואר בהמשך) ותמשיך עד שתגיע לתא ריק כלומר לכל היותר פעם אחת (כי בכל פעם מחברים שני עצים מאותה דרגה ושמים בתא הבא). לכן, סיבוכיות לולאות ה while בסה''כ . לאחר מכן רצים בלולאת for 3 פעמים נפרדות על גודל המערך (גודל המערך חסום ע''י כפי שראינו בהרצאה) כדי לבצע השמות לשדות שונים ולכן זה לא מוסיף לסיבוכיות ובסה''כ במקרה הרוע קיבלנו שזמן הריצה .

זמן הריצה amortized הוא מכיוון שהאלגוריתם שתיארנו לעיל זהה לאלגוריתם המתואר בהרצה ולכן זמן הריצה הוא .

decreaseKey :

**הסבר:** בהתחלה אנו עושים בדיקה שלא מתבצע overflow הגורם להגדלה של המפתח במקום להקטנה. לאחר מכן, אנו מחסירים מהמפתח את הערך שקיבלנו ומעדכנים את השדה המינימלי בהתאם. אחר כך, אנו בודקים אם הצומת הוא לא שורש וגם מפר את כלל הערמה אז קוראים לפונקציית cascadingCut שאחראית על ההמשך (חיתוך הצומת סימון וכך הלאה עד השורש).

**סיבוכיות:** במקרה הגרוע ובמקרה amortized . במקרה הגרוע לאחר ביצוע הפחתה במפתח נקבל הפרה במפתח ונקרא לפונקציית cascadingCut. פונקציה זו רצה בסיבוכיות במקרה הגרוע (כפי שיתואר בהמשך) ולכן בסה''כ נקבל כי זמן ריצה במקרה הגרוע הוא .

במקרה amortized זמן הריצה הוא קבוע מכיוון שבסך הכל פונקציה זו עושה פעולות בזמן קבוע (השמה לשדות) וגם הקריאה לפונקציית cascadingCut באמורטייזד זה . ולכן בסה''כ נקבל כי זמן הריצה באמורטייזד הוא .

cutNode :

**הסבר:** אנו קוראים לפונקציה זו רק מתוך פונקציית cascadingCut כאשר יש הפרה של כלל הערמה. בהתחלה מבטלים את הסימון אם הוא היה. מעדכנים את השדות הרלוונטים ומנתקים את הצומת שקיבלנו מההורה ומחברים אותו לשאר השורשים שהיו בעזרת המצביעים.

**סיבוכיות:**  קבועה. בסך הכול, כל מה שנעשה בפונקציה זו הוא השמה של של מצביעים ולכן הסיבוכיות היא קבועה.

cascadingCut :

**הסבר:** פונקציה זו נקראת כאשר מבצעים decreaseKey ויש הפרה של כלל הערמה. נבדוק אם ההורה של הצומת שקיבלנו מסומן. אם לא, נקרא לפונקציית cutNode עבור הצומת ונעדכן את הסימון. אם ההורה של הצומת שקיבלנו כן מסומן אז נעבור בלולאה על הצומת שקיבלנו עד לשורש כל עוד ההורה מסומן ונחתוך את הצומת ע''י קריאה לפונ' cutNode ונדאג לסימונים של הצמתים המעורבים.

**סיבוכיות:** במקרה הגרוע ובמקרה amortized . במקרה הגרוע כל השרשרת מהצומת ועד לשורש מסומנת ולכן נעבור בלולאת ה while על כל המסלול עד לשורש. בתוך הלולאה מבצעים פעולות בזמן קבוע ולכן נקבל שזמן הריצה הוא .

במקרה amortized זמן הריצה הוא קבוע מכיוון שהאלגוריתם שתואר לעיל זהה לאלגוריתם שראינו בכיתה ולכן זמן הריצה הוא קבוע.

Insert:

**הסבר:** הפונקציה מקבלת מפתח ומוסיפה צומת עם המפתח שהתקבל למבנה ע"י הוספת האיבר כשורש (השמאלי ביותר).

כמו כן מסיבות שיפורטו בהמשך העמסנו פונקציית insert נוספת עם חתימה שונה כדי לתמוך בהכנסה של איבר עם שדה מיוחד.

**סיבוכיות:** כל פעולה לוקחת לכל היותר זמן קבוע ולכן גם במקרה הגרוע וגם ב – Amortized זמן הריצה הוא קבוע O(1).

Is Empty:

**הסבר:** מחזירה האם הערימה ריקה על ידי בדיקה האם קיים איבר מינימלי.

**סיבוכיות:** קבוע – שכן יש לנו שדה מינימום - O(1) גם במקרה הגרוע וגם ב- amortized.

Connect:

**הסבר:** פונקציית עזר המקבלת שני צמתים עם דרגה זהה. הפונקציה מחברת ביניהם ע"י חיבור של הצומת הגדול יותר כבן שמאלי ביותר של צומת הקטן יותר ומחזירה את הצומת הקטן יותר (לאחר החיבור).

**סיבוכיות:** זמן הריצה הוא קבוע שכן אנו משנים מצביעים מקומית – O(1) גם במקרה הגרוע וגם ב- amortized.

Find min:

**הסבר:** הפונקציה מחזירה את הצומת המינימלי במבנה – אנו מחזיקים מצביע למינימום.

**סיבוכיות:** זמן ריצה קבוע.

Meld:

**הסבר:** הפונקציה מקבלת שתי ערמות (this,heap2) ומחברת את heap2 לערמה נוכחית.

הפונקציה כפי שנלמד בהרצאה משרשרת את הערמות אחת לשנייה. כלומר מחברת איבר אחרון של ערמה נוכחית עם איבר ראשון של ערמה השנייה.

**סיבוכיות:** הפונקציה משנה מצביעים מקומית ולכן רצה בזמן קבוע גם WC וגם amortized.

Size:

**הסבר:** הפונקציה מחזירה את גודל המבנה – אנו שומרים שדה גודל המעודכן בכל הכנסה ומחיקה.

**סיבוכיות:** זמן ריצה קבוע – החזרת ערך משדה.

Counter rep:

**הסבר:** הפונקציה מחזירה מערך שגודלו כדרגת העץ המקסימלי בערמה. כאשר בתא ה – i מופיעה מספר השורשים מדרגה i. עוברים על כול השורשים וסופרים דרגות.

**סיבוכיות:** זמן הריצה הוא כאמור כמספר השורשים במבנה ולכן במקרה הגרוע הוא לינארי. כמו כן בזמן amortized נביט בסדרת ב m פעולות counter rep רצופות עבור מצב בוא מספר השורשים הוא n ונקבל כי זמן הריצה של סדרת פעולות גרוע הוא n\*m ולכן גם כאן הזמן הוא לינארי.

Delete:

**הסבר:** מחיקת צומת בהינתן מצביע. עושים decrease key בערך מינימלי של integer ומוחקים צומת מינימלי מהמבנה.

**סיבוכיות:** כפי שתואר מעלה decrease key עולה במקרה הגרוע O(log n) וכן delete min עולה O(n) במקרה הגרוע ולכן בסה"כ העלות במקרה הגרוע היא O(N) ובזמן amortized עלות פעולה זו היא כסכום זמני ה amortized שפורטו מעלה עבור הפעולות הרלוואנטיות ולכן בסה"כ נקבל זמן amortized o(log n).

If Root:

**הסבר:** מחזירה האם צומת הוא שורש על ידי בדיקה של ההורה שלו. אם ההורה שלו Null אזי שהוא שורש.

**סיבוכיות:** זמן קבוע.

Non marked:

**הסבר:** הפונקציה מחזירה את מספר הצמתים הלא מסומנים בעזרת השדה marked המכיל את מספר הצמתים שכן מסומנים. עדכון השדה מתבצע בכל המקומות בהם נעשה שינוי במספר המסומנים כמו כן עדכון השדה אינו משפיעה על זמני הריצה.

**סיבוכיות:** מחזירים שדה – זמן קבוע.

Potential:

**הסבר:** הפונקציה מחזירה את מספר השורשים + פעמיים מספר המסומנים.

**סיבוכיות:** זמן ריצה קבוע שכן יש שדה למספר השורשים וכן שדה למס המסומנים.

עדכון השדות נעשה ביעילות שכן אנו מבצעים את העדכון או עבור שינוי בצומת בודד ואז השינוי עולה זמן קבוע או כאשר אנו בכול מקרה עושים פעולת cascading ואז משלמים עוד פעם את מספר השורשים שזה שקול למציאת המינימום ולכן עדכון השדה אינו משפיעה על זמני ה – amortized, ובפרט אינו משפיע על זמני הריצה במקרה הגרוע.

Total links:

**הסבר:** הפונקציה מחזירה את מספר פעולות ה – Link שבוצעו בכל הערמות שנוצרו מתחילת הרצת התוכנית.

מתודה סטטית זו עושה שימוש בשדה סטטי הסופר את מספר הפעולות הנ"ל.

**סיבוכיות:** קבוע

Total cuts:

**הסבר:** הפונקציה מחזירה את מספר פעולות ה – cut שבוצעו בכל הערמות שנוצרו מתחילת הרצת התוכנית.

מתודה סטטית זו עושה שימוש בשדה סטטי הסופר את מספר הפעולות הנ"ל.

**סיבוכיות:** קבוע

K-min:

**הסבר:** הפונקציה מחזירה את K האיברים הכי קטנים בערמה H המתקבלת כקלט.

אופן פעולת הפונקציה: תחילה ניצור ערמת בא לכל צומת יש מצביע ל – node.

נאתחל את ערמת העזר עם המינימום (השורש של H) ומצביע לאותו צומת ב H לאחר מכן בכול שלב (k שלבים בסה"כ) נוציא מערמת העזר את המינימום ע"י find Min + delete min ונכניס את ילדיו (ב- H) לערמת העזר בעזרת המצביע המיוחד וכמו כן נכניס את ילדיו עם מצביע לאותם האיברים ב H כך שבכל פעם שנוציא איבר נקבל קישור ל H ונוכל לגשת לצמתים הרלוואנטיים. בכול שלב אנו מקבלים את המינימלי הבא ומכניסים ומערך - לאחר k שלבים אנו מחזירים את המערך ומסיימים. נבחין כי לא שינינו כלל את H.

**סיבוכיות:** נסמן ב n את מספר האיברים ב H. נטען כי בערמת העזר יש לכל היותר k\*log n צמתים שכן בכל שלב אנו מכניסים לערמת העזר לכל היותר מספר צמתים כדרגת הצומת המקסימלי במבנה. שכן הראינו בכיתה כי דרגת צומת מקסימלי היא לוגריתמית בגודל המבנה ולכן זהה עד כדי קבוע לגובהה העץ.

בנוסף בכול שלב אין יותר מ deg(H) שורשים שכן בכול שלב אנו מכניסים לכל היותר deg(H) צמתים חדשים ולכן עלות כל פעולת delete min היא כהפרש מספר השורשים ולכן העלות היא

שכן בין כל שני שלבים אנו מקבלים ערמה בינומית תקינה.

בנוסף כפי שראינו פעולות ההכנסה עולות זמן קבוע ולכן בסה"כ סיבוכיות זמן הריצה של פונקציה זו נתונה על ידי: כנדרש.

**חלק עיוני:**

שאלה 1:

1. זמן הריצה האסימפטוטי של סדרת הפעולות כפונקציה של היא לינארית . בהתחלה אנו מבצעים הכנסה לערמה של 1+m איברים, זמן הריצה של הכנסה הוא מבצעים זאת בלולאה m פעמים ולכן יעלה . לאחר מכן, אנו מבצעים deleteMin פעם אחת, זמן הריצה הוא - ראינו בהרצאה. לאחר ה deleteMin קיבלנו עץ בינומי מלא מכיוון שנתון כי M הוא חזקה של 2. ולבסוף מבצעים פעמים את הפעולה decreaseKey. זמן הריצה של decreasKey הוא . נשים לב שאת הפעולה decreaseKey אנו מבצעים רק לעלים בעלי הורה שונה וזה נובע מהגדרת העץ הבינומי שלכל צומת שיש לו שני בנים הם לא מאותה דרגה ולכן בעלי עומקים שונים. כתוצאה מכך, לכל שני עלים ההורה אינו זהה. לכן צומת לא מאבדת יותר מבן אחד ולא נבצע פעולות חיתוך הגורמות לזמן ריצה שגדול מקבוע עבור פעולת decreaseKey. בעקבות זאת, בסה''כ נקבל שזמן הריצה הוא .

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| m | Run-Time (ms) | totalLinks | totalCuts | Potential |
|  | 1 | 31 | 5 | 14 |
|  | 3 | 1023 | 10 | 29 |
|  | 14 | 32,767 | 15 | 44 |
|  | 170 | 1,048,575 | 20 | 59 |

ב.

1. מספר הפעולות שמתבצעות במהלך סדרת הפעולות של link הוא m-1 מכיוון שבהכנסה נכנסים m איברים כאשר כל איבר צומת בודד. לאחר מכן מבצעים deleteMin מה שגורם לאיחוד של הצמתים. אוכיח באינדוקציה כי מספר פעולות הלינק הוא m-1. עבור m=1 שזה צמתים נקבל 0 לינקים – טריוויאלי. עבור m'<m נניח נכונות ונראה עבור m. נבצע את פעולת הלינק עבור M/2 האיברים הראשונים ונקבל מהנחת האינדוקציה כי ביצענו m/2-1 פעולות לינק (M חזקה של 2 ובפרט m מתחלק ב 2). כעת נבצע באופן זהה על m/2 האיברים הנוספים וכל שנשאר לעשות זה לחבר בין שני העצים בעלי אותה הדרגה שזה עוד לינק נוסף. בסה''כ נקבל כי מספר פעולות הלינקים הן

m/2-1+ m/2-1+ 1 כלומר m-1 פעולות.

מספר פעולות של cut הוא מכיוון שמספר פעולות decreaseKey הוא . וכן כפי שהוסבר בסעיף א, אנו מבצעים פעולות cut רק עבור עלים עם הורים שונים ולכן כמות פעולות החיתוך שוות לכמות פעולות decreaseKey.

הפוטנציאל של המבנה בסוף הסדרה הוא 3 מכיוון שכמו שהזכרנו קדום מספר החיתוכים שווה ל השווה למספר השורשים שהתווספו ובנוסף שווה לכמות הצמתים המסומנים שהתווספו פחות אחד (מפני שלא מסמנים את השורש) ולכן נקבל 3 לפוטנציאל (כי כפלנו ב 2 את מסמפר הצמתים המסומנים). לבסוף נוסיף אחד מכיוון שנשאר לנו השורש המקורי שממנו התחלנו ולכן בסה''כ הפוטנציאל שווה ל 3.

ד. מספר הלינקים נשאר זהה מכיוון שלינק מתבצע רק במחיקת המינימום – בשורה השנייה בקוד ומכיוון שלא נגענו בשורות אלו מספר הלינקים נשאר זהה. מספר החיתוכים הוא אפס מכיוון שמבצעים decreaseKey במספר קבוע לשורש ואז לבן הכי גדול שלו ואז לבן הכי גדול שלו ולכן באף שלב לא מתקיימת הפרה של חוק הערמה ולכן אין צורך בחיתוכים. הפוטנציאל הוא אחד מכיוון שכלל הערמה לא נפגע כפי שאמרנו ולא לא התבצע אף חיתוך ולא סימנו אף צומת כלומר, נשאר שורש אחד שממנו הפוטנציאל מורכב.

ה. מספר הלינקים הוא אפס מכיוון שמחקנו את שורה 2 של מחיקת המינים האחראית על חיבור הצמתים ולכן לא התבצע אף חיבור. מכיוון שלאחר ההכנסה הצמתים נשארו בודדים ולא חוברו אין אופציה לעשות חיתוכים (כי לא נוצרה ערימה אלא רק צמתים בודדים) ולכן מספר החיתוכים הוא אפס. הפוטנציאל שווה למספר השורשים M+1 , כי כמו שאמנו לא סימנו אף צומת.

ו. מספר הלינקים לא השתנה מכיוון שלא נגענו בשורות הקוד האחריות על זה (שאלו שורות 1 2 כפי שהזכרנו קודם). מספר החיתוכים גדל מכיוון שבהוספת שורה 4 אנו מבצעים decreaseKey להורה שכבר היה מסומן כי חתכנו לו בן בשורה שלוש.

כפי שהסברנו קודם, בשורה שלוש מבצעים חיתוך לעלים שאין להם הורה משותף וכך גורמים לצמתים בעומקים שהולכים וגדלים להיות מסומנים. ולכן כאשר הוספנו את שורה 4 גרמנו לשרשרת של חיתוכים מההורה ועד לשורש כיוון שכל הצמתים במסלול כבר היו מסומנים ולכן עלינו מעלה וחתכנו כל צומת שהיה במסלול עד השורש. בעקבות כך, מספר החיתוכים גדל במספר השווה לגובה הערמה פחות אחד (כיוון שלא חותכים את השורש) - כלומר, סה"כ קיבלנו 2 חיתוכים. וכן בהתאם הפוטנציאל קטן במספר השווה לגובה הערמה פחות אחד, כיוון שהצמתים שהיו מסומנים (שהיו בדרך לשורש וחתכנו אותם) הפכו לשורשים.

כמו כן הפעולה decrease key היקרה ביותר עולה שכן הראינו הסברנו מעלה למה מתקבל פעולה בעלות זו וכן יתר הפעולות כפי שפורט בסעיפים קודמים מבצעים חיתוך יחיד ולכן עולות מספר קבוע של פעולות.

**טבלת סיכום תוצאות:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| case | totalLinks | totalCuts | Potential | decreaseKey max cost |
| (c) original | m-1 |  | 3 | (skip) |
| (d) decKey(m-2^i) | m-1 | *0* | 1 | (skip) |
| (e) remove line #2 | 0 | 0 | *m+1* | (skip) |
| (f) added line #4 | m-1 | 2 | 2 |  |

שאלה 2:

א.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| m | Run-Time (ms) | totalLinks | totalCuts | Potential |
| 728 | 2 | 723 | 0 | 6 |
| 6560 | 6 | 6555 | 0 | 6 |
| 59,048 | 38 | 59040 | 0 | 9 |
| 531,440 | 286 | 531431 | 0 | 10 |
| 4,782,968 | 2673 | 4782955 | 0 | 14 |

1. זמן הריצה האסימפטוטי של סדרת הפעולות כפונקציה של הוא . בהתחלה אנו מבצעים הכנסה לערמה של 1+m איברים, זמן הריצה של הכנסה הוא מבצעים זאת בלולאה m פעמים ולכן יעלה . אנחנו מקבלים ערמה עם m+1 עצים מדרגה 0. לאחר מכן, אנו מבצעים deleteMin פעמים. לאחר המחיקה הראשונה נותרו עצים עליהם נצטרך לעבור ולבצע לינקים כנדרש. כלומר, פעולת המחיקה הראשונה תעלה . לאחר מכן, נקבל ערמה עם עצים ולכן כל פעולות המחיקה הבאות מעכשיו יעלו כמספר העצים ועוד הדרגה המקסימלית של האיבר שנמחק. כלומר, פעולות אלו חסומות ע''י . גודל הערמה קטן עם מספר המחיקות. אחשב את סיבוכיות הזמן באופן הבא:

סה"כ קיבלנו כי זמן הריצה הוא .

1. מספר הפעולות שמתבצעות במהלך סדרת הפעולות של link הוא m-k. k שווה למספר העצים לאחר מחיקת האיבר המינימלי הראשון וגם שווה למספר האחדות שיש בייצוג הבינארי של m. אוכיח זאת בעזרת הטענות הבאות: לאורך כל סדרת הפעולות, הפעם היחידה בה נעשות פעולות link היא לאחר המחיקה הראשונה. טענה נוספת היא שלאחר המחיקה הראשונה תתקבל ערימה בינומית תקינה כאשר העץ הקטן ביותר הוא בעל האיברים הקטנים ביותר והעץ הגדול ביותר בעל האיברים הגדולים ביותר זה נובע עקב האופן בו נבנתה הערימה – הכנסת האיברים לערמה בסדר עולה של מפתחות. בעקבות כך, בכל מחיקה של האיבר המינימלי נמחק את השורש של העץ הקטן ביותר. לאחר המחיקה הראשונה אחלק את פעולת מחיקת המינימום לשני מקרים. מקרה ראשון לשורש אין ילדים ולכן נצטרך לעבור על השורשים כדי למצוא את המינימום החדש (לא יתבצעו פעולות link). מקרה שני הוא שלאיבר המינימלי יש x ילדים ולכן יתווספו x שורשים לערמה מדרגות 0 עד x-1. כל התאים של דרגות אלו במערך ריקים כי שאר השורשים בדרגות גבוהות יותר מ x, לכן לא יתבצעו פעולות link. טענה נוספת היא שמספר הלינקים שהתבצעו שווה למספר העצים שהיו בהתחלה – m עצים פחות מספר העצים לאחר המחיקה שהוא שווה למספר האחדות בייצוג הבינארי של m. הטענה נובעת מכיוון שבהרצאה הראנו שמספר העצים ודרגתם בעץ בינומי תקין בגודל m נקבעת לפי הייצוג הבינארי של m באופן יחיד. לכן קיבלנו כי מספר העצים לאחר המחיקה הראשונה שווה למספר האחדות בייצוג הבינארי של m. מעבר לכך, כל פעולת לינק מקטינה את מספר השורשים באחד ולכן מספר פעולות הלינק שווה להפרש בין מספר העצים לפני המחיקה הראשונה לבין מספר העצים לאחר המחיקה הראשונה. מהטענות שטענו לעיל נקבל כי מספר הלינקים שווה ל m-k.

מספר פעולות של cut בסוף כל סדרת פעולות הוא 0 מכיוון שמוחקים בכל איטרציה את האיבר המינימלי – לא נוצרת הפרה של כלל הערמה ולכן אין צרך בחיתוך ובפרט בסימון של צומת.

בסוף סדרת הפעולות הפוטנציאל שווה למספר האחדות בייצוג הבינארי של המספר הסיבה לכך היא מכיוון שהפוטנציאל מוגדר להיות מספר השורשים + מספר המסומנים \* 2 , ומספר המסומנים הוא אפס מהסיבה שהסברנו קודם אז הפוטנציאל שווה למספר השורשים בסוף סדרת הפעולות. בהתחלה היו m+1 שורשים בוצעה מחיקה של צמתים ולכן בסוף נותרו צמתים. לכן מהטענה מההרצאה שהזכרנו לעיל, מתקיים כי מספר השורשים שווה למספר האחדות בייצוג הבינארי של מספר הצמתים .