**פרוייקט מעשי – ערמת פיבונצ'י**

# שמות המגישים

● עומר יצחק 205656986

● סער ג'רסי 313221210

# חלק א' - תיעוד

## מבנה כללי

* **מחלקה FibonacciHeap -** מחלקה המגדירה אובייקט מסוג ערימת פיבונאצ'י עם צמתים שהם

אובייקטים מסוג .HeapNode היא מאתחלת עץ באופן ריק עם גודל 0 ומינימום שמוגדר כ.NULL-

* מחלקה HeapNode - מחלקת עזר שמייצרת אובייקטים מסוג node ומבצעת את הקישור בין הצמתים

באופן שלמדנו בכיתה.

## תיאור המחלקה FibonacciHeap

**שדות**

1. Size - מספר האיברים בכל ערימה בערימה
2. numberOfMarked - מספר הצמתים ה"מסומנים" בערימה עד כה בערימה.
3. numberOfCUTS - מספר החיתוכים שהתבצעו עד כה מהערימה.
4. numberOfLINKS - מספר חיבורי הערימות שהתבצעו.
5. numberOfTrees - מספר העצים הנמצאים כעת כחלק
6. Min - מצביע לצומת המינימלי בעץ, לפי הלוגיקה זהו אחד השורשים
7. head - מצביע עבור השורש של העץ השמאלי ביותר .

isEmpty

## מתודות

.1

תיאור - מתודה בוליאנית המחזירה אם הערימה ריקה, כלומר אם השורש של העץ הראשון

בערימה מאותחללהיות null.

סיבוכיות - שליפת ערך של המצביע והשוואתו לNULL- מתבצעת בסיבוכיות קבועה .O(1)

תיאור - מתודה המתחילה את תהליך ההכנסה של עץ בינומי מסדר 1 לערימה. נשענים על העובדה שהמפתח לא נמצא בערימה מלכתחילה. המתודה מגדילה את גודל העץ ב,1- יוצרת

אובייקט צומת חדש ומגדילה את מספר העצים הכללי. המתודה בודקת אם הערימה ריקה,

ואם כן - מכניסה איבר חדש ומגדירה אותו כמינימום וראשון. אם הערימה אינה ריקה, מתבצעת הכנסה של הצומת אל רשימת השורשים משמאל לאיבר הראשון ועדכון המינימום

החדש במידת הצורך.

○

○

## 2. insert

○

סיבסיבוכיות - תחילה אם הצומת אינו קיים ניצור את אובייקט הצומת החדש בזמן קבוע. כך גם עדכוני הגדלים והמצביעים יתבצעו בזמן קבוע. סה"כ כל המתודה מבצעת קריאות לערכי המצביעים ומשנה את ההצבעות ולכן מתבצעת בזמן קבוע .O(1)

תיאור – מתודה זאת נשענת על המתודה decreaseKey בכך שהיא מקטינה את ערך המפתח של הצומת למינימום האפשרי, ולאחר מכן קוראת למתודה deleteMin

שמבצעת את כל השאר.

סיבוכיות - מכיוון שהמתודה הנ"ל לא מבצעת דבר למעט קריאה לשתי המתודות האחרות

שציינתי, הסיבוכיות זהה לסכום ערכי הסיבוכיות של אותן המתודות. כפי שאפרט בהמשך, decreaseKey מתבצעת בזמן קבוע אמורטייזד, ובזמן לוגריתמי במקרה הגרועdeleteMin. מתבצעת בסיבוכיות לוגריתמית אמורטייזד ולינארית במקרה הגרוע. לכן סה"כ הסיבוכיות

היא O(logn) אמורטייזד וO(n)- במקרה הגרוע.

○

delete .3

○

○

deleteMin .4

○ תיאור - מתודה זו מוחקת מהערימה את השורש של העץ הבינומי הקטן ביותר, עליו יש כבר

מצביע. אם הערימה ריקה או שיש בה צומת יחיד, אזי זהו המינימום ונישאר עם ערימה

ריקה. אחרת, נתחיל את המחיקה במקביל לקונדולידציה של העצים כפי שלמדנו בכיתה. המתודה תיצור מערך עזר של "דליים" ובה היא תאחסן את שורשי העצים החל מהצד השמאלי ועד הימני. הקונסולידציה תתבצע במהלך ההעברה לדליים, ואם נגיע לשורש העץ המינימלי, נוכל להעביר את כל ילדיו אל תוך הדליים( גם כן משמאל לימין), בכדי לבצע עליהם קונסולידציה נוספת. בכל פעם שמצטבר בדלי כלשהו יותר מעץ יחיד, נקרא למתודת העזר .linkHeaps לבסוף תהיה קריאה למתודה creatLegalHeap אשר ממירה את הדליים לערימה בינומית תקנית.

○ סיבוכיות - מכיוון שבכל מקרה במקרה הגרוע ביותר יתכן שיהיו n עצים ונצטרך לעבור על

כולם כדי לבצע קונסולידציה, הסיבוכיות תהיה .O(n) המתודה linkHeaps מתבצעת בזמן קבוע, משום שהיא רק משנה מצביעים, והקריאה לcreatLegalHeap שרצה בזמן לוגריתמי אינה מגדילה את זמן הריצה הלינארי, סיבוכיות אמורטייז של פעולה זו תהיה O(logn).

findMin .5

תיאור - מתודה המחזירה את ערך המצביע לצומת המינימלי.

סיבוכיות - המתודה מחזירה ערך של מצביע ולכן הסיבוכיות קבועה .O(1)

תי

אותיאור- מתודה אשר מבצעת חיבור בין שני עצים בינומים בכך שהיא בודקת שהם לא ריקים, או שאם אחד ריק היא מחזירה אחד מהם. אם שניהם לא ריקים, אזי היא מבצעת שינויי מצביעים בכדי לאחד בין העצים ובכך מדגילה את ה,size מספר העצים, מעדכנת מינימום וכו.'

סיבוכיות - מתבצעים שינויי מצביעים בלבד ולכן הסיבוכיות קבועה .O(1)

○

○

meld .6

○

○

תיאור - המתודה מופעלת על הערימה ומחזירה את כמות הצמתים בה.

סיבוכיות - מדובר במתודה המחזיקה ערך של מצביע ולכן הסיבוכיות קבועה .O(1)

size .7

○

○

counterRep .8

○ תיאור - הפונקציה יוצרת מערך שבו ערך האינדקס מסמל את מספר העצים בערימה בדרגה

זו. המתודה מייצרת מערך בגודל מספר הצמתים ועוברת על שורשי העצים בערימה ובבדיקת

הדרגה שלהם, מעדכנת את האינדקסים.

○ סיבוכיות - במקרה הגרוע ביותר יהיו n שורשים בערימה ונצטרך לעבור על כולם, ולכן

הסיבוכיות תהיה .O(n)

creatLegalHeap .9

○ תיאור - המתודה הינה מתודת עזר אשר מקבלת מערך של דליים ומתחילה מחיפוש האיבר

בעל הדרגה הקטנה ביותר בכדי שיהיה האיבר הראשון. לאחר מכן מתבצעת בנייה של הערימה מתוך איחוד שרשרת השורשים באמצעות שינויי המצביעים. לבסוף מתבצע עדכון של

השורש המינימלי וסגירת המעגל של רשימת השורשים כפי שנדרש.

○ סיבוכיות - נצטרך לעבוד על כל מערך הדליים ולכן .O(logn)

decreaseKey .10

○ תיאור - המתודה מקטינה את הערך של המפתח של הצומת שהיא מקבלת לערך שנקבע על ידי

הדלתא. היא מעדכנת את המינימום להיות העדכני. החלק השני של הריצה הינו בדיקה

שהסדר של הערימה נשמר, ואם לא, תבוצע קריאה למתודת העזר .cascadingCuts

○ סיבוכיות - בחלק הראשון של הריצה, מתבצעים שינויי מצביעים ועדכון של ערכים בזמן

קבוע. מכיוון שמתבצעת קריאה למתודה cascadingCuts בחלק השני, הסיבוכיות תלויה בה

)אפרט בהמשך( ולכן נקבל סה"כ O(n) בזמן הגרוע וזמן קבוע אמורטייזד.

cascadingCut .11

תיאור - הפונקציה מתחילה מקריאה למתודת העזר cutnode אשר "חותכת" מהעץ את הצומת שיצר את הפרת הסדר במערך. המתודה הינה רקורסיבית ונעצרת ברגע שבו היא רואה שהצומת הוא בן ישיר של שורש בערימה. כפי שלמדנו בכיתה, שורש אינו יכול להיות מסומן ולכן היא מורידה את הסימונים במידת הצורך.

סיבוכיות - מכיוון שהמתודה הינה רקורסיבית ובמקרה הגרוע ביותר נצטרך לעלות לגובה כל העץ נקבל סה"כ שהסיבוכיות חסומה ע"י גובה העץ, מכיוון שאנו יכולים ליצור מצב שיהיה שרוך וכולם מסומנים, למעט השורש, ע"י פעולות cut ו- delete ולכן הסיבוכיות היא .O(n) ובזמן אמורטייזד קבוע.

תי

אותיאור - מתודת עזר שמבצעת את החיתוך עצמו של הצומת וחיבורו כשורש חדש בערימה, בכך היא צריכה לעדכן את כל המצביעים לצמתים הסמוכים ואת גודל הערימה ועדכון מספר הcuts- ומספר העצים.

סיבוכיות - מכיוון שהמתודה מבצעת רק שינויי מצביעים ומעדכנת ערכים למשתנים קיימים,

זמן הריצה הינו קבוע .O(1)

○

○

cutNode12

○

○

potential .13

○ תיאור - המתודה משתמשת בערכים ידועים מראש ומבצעת חישוב אריתמטי פשוט של סכום

מספר העצים ופעמיים מספר הצמתים המסומנים.

סיבוכיות - מכיוון שהמתודה מריצה חישוב אריתמטי פשוט ומשתמשת במשתנים שערכם

כבר קבוע מראש במהלך הריצה, זמן הריצה הינו קבוע .O(1)

תתיאור - המתודה מקבלת שני עצים בינומים מדרגה זהה ובודקת איזה עץ צריך להיות גבוה יותר )כלומר לאיזה יש את המפתח המינימלי יותר.( לאחר פעולת הבדיקה יש חלוקה למקרה הפשוט שבו שני העצים הם מדרגה 0 ואז רק נותר לחברם לעץ מדרגה ,1 או לבצע

חיבור שין שני עצים ואז לבצע השמה מחדש של המצביעים של כל הבנים הרלוונטיים.

סיבוכיות - מכיוון שהמתודה מבצעת רק שינויי מצביעים ומעדכנת ערכים למשתנים קיימים,

זמן הריצה הינו קבוע .O(1)

○

1. linkHeaps

○

○

totalLinks .15

○ תיאור - מתודה זו מחזירה את מספר הלינקים שהתבצעו, מדובר בשדה שמתעדכן במהלך

הריצה ולכן אין צורך לעדכן.

○ סיבוכיות - שליפה של ערך מתבצעת בסיבוכיות .O(1)

totalCuts .16

תיאור - מתודה זו מחזירה את מספר החיתוכים שהתבצעו, מדובר בשדה שמתעדכן במהלך

הריצה ולכן אין צורך לעדכן דבר.

סיבוכיות - שליפה של ערך מתבצעת בסיבוכיות קבועה .O(1)

תיאור - המתודה בודקת אם k הינו מספר חיובי, אחרת מחזירה מערך ריק. אם k=1 אזי

נחזיר מערך שמכיל רק את האיבר המינימלי. במידה ולא נקלענו למקרים הפשוטים הללו, המתודה יוצרת ערימת עזר חדשה ומכניסה אליה את המינימום. מתבצעת לולאה של k איטרציות, ובכל איטרציה נכניס למערך את האיבר המינימלי בערימת העזר. לאחר ההכנסה תתבצע מחיקה של האיבר המינימלי ממערך העזר ונכניס את הילדים של האיבר המינימלי בערימה המקורית לערימת העזר. המטרה של המצביע של הcurrParent שמתעדכן בכל פעם היא לקשר בין שרשרת של ילדים בערימה העזר לשרשרת המתאימה לה בערימה המקורית. בכל איטרציה נבצע הכנסה שמתבצעת בזמן קבוע, ומספר הילדים חסום על ידי הדרגה של העץ

סיבוכיות - מכיוון שבכל איטרציה לבצע הכנסה שמתבצעת בזמן קבוע, ומספר הילדים חסום על ידי הדרגה של העץ degH אזי הסיבוכיות תהיה לכל היותר .O(k\*degH) בנוסף, בכל איטרציה ביצענו קריאה לdeleteMin )לא על הערימה המקורית כפי שהתבקשנו בתנאי השאלה.( מכיוון שהסיבוכיות של deleteMin פועלת בזמן אמורטייזד לוגריתמי במספר האיברים בערימה, ובכל איטרציה מספר האיברים חסום על ידי .k מספר ההכנסות חסום על ידי הדרגה degH ולכן גם deleteMin יהיה חסום ונקבל שהסיבוכיות נשארת .O(k\*degH)

kmin .17

○

○

1. size

○ תיאור – המתודה מחזירה את גודל הערימה, ע"י לקיחת הערך השמור בשדה size.

○ סיבוכיות- המטודה מחזירה ערך מצביע שמור לכן הסיבוכיות היא o(1).

1. Log2

○ תיאור – מתודת עזר המחזירה את הערך המתקבל ע"י ביצוע פעולת לוג בבסיס 2 על הקלט.

○ סיבוכיות- המתודה מבצעת חישוב אריתמטי פשוט ועל כן הסיבוכיות היא o(1).

1. gethead

○ תיאור – המתודה מחזירה את הצומת אשר המצביע לשורש מצביע עליה.

○ סיבוכיות- המתודה מחזיר ערך מצביע שמור ועל כן הסיבוכיות היא o(1).

## תיאור המחלקה HeapNode

**שדות**

.1 Key - המפתח של הצומת, מספר טבעי.

.2 Rank - דרגת הצומת )מספר הבנים שיש לה,( מספר טבעי.

.3 Marked - ערך מספרי שמגדיר אם הצומת מסומנת, אם אחד הבנים שלה "נחתך" במתודה .cut

.4 Child - מצביע עבור אובייקט צומת הבן )השמאלי ביותר( של הצומת.

.5 Prev - מצביע עבור אובייקט הצומת הקודם )השמאלי,( פועל באופן מעגלי.

.6 Next - מצביע עבור אובייקט הצומת הבא )הימני,( פועל באופן מעגלי.

.7 Parent - מצביע עבור אובייקט צומת האב של הצומת.

.8 currParent - מצביע עזר לצורך המתודה kmin המצביע לNULL- עד להפעלת המתודה.

תיאור - מתודה המאפשרת לקבל את המפתח של הצומת.

סיבוכיות - שליפה של ערך מתבצעת בסיבוכיות קבועה .O(1)

getKey

○

○

## מתודות

.1

# חלק ב'

א. זמן הריצה האסימפטומטי במונחי O, עבור רצף הפעולות הנ"ל הוא O(m). זאת מכיוון שאנו מתחילים מרצף של m+1 הכנסות, עם הפונקציה insert שמתבצעת בזמן קבוע ) .O(1)סה"כ עד כה קיבלנו סיבוכיות לינארית O(m) .

אחר מכן מתבצעת קריאה בודדת ל deleteMinוכפי שראינו בהרצאה, הסיבוכיות של מתודה זו הינה לינארית . O(m)מכיוון, שזו פעולת ה deletemin הראשונה שמתבצעת.

בסוף יתבצעו קריאות לdecreaseKey כאשר הסיבוכיות של המתודה הנ"ל חסומה ע"י מספר החיתוכים שהתבצעו. כאן ניתן לראות שמספר החיתוכים שהתבצעו הוא לוגריתמי כתלות ב m . ולכן הוספה של פעולות בסיבוכיות לוגריתמית אינה משנה את זמן הריצה במקרה הגרוע שהוא O(m) .

ב.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| potential | Total cuts | Total links | Run\_time (ms) | m |
| 29 | 10 | 1023 | 5 | 2^10 |
| 44 | 15 | 32767 | 7 | 2^15 |
| 59 | 20 | 1048575 | 64 | 2^20 |
| 74 | 25 | 33554431 | 13508 | 2^25 |

ג. כפי שניתן לראות בטבלה, מספר פעולות ה link הוא O(n) ומספר החיתוכים הוא O(logn). כמו כן, מכיוון שהפוטנציאל הוא כמספר העצים + פעמיים מספר הצמתים המסומנים, הוא O(logn). וכעת נסביר, מבחינת פעולת ה link, ניתן להבחין כי כל m+1 האיברים נכנסים כשורשים נפרדים לערימה, כל אחד לצד השמאלי ביותר. אחרי ההכנסות יהיו לנו m+1 עצים בערימה, ולאחר קריאה למתודה deletemin, שקוראת למתודה link, מספר החיבורים יהיה כמספר האיברים פחות 1. ולכן הסיבוכיות תיהיה O(n).

מבחינת פעולות ה cut , נבצע decrease key , על עץ אחד, בגובה של logm (מכיוון שזהו העץ שהתקבל לאחר פעולת ה delete min). פעולת ה cut תתבצע בכל הפרה של הסדר של הערימה, ולכן היא חסומה בגובה העץ O(logm), ולכן ניתן לראות שכמות החיתוכים, היא logm

מבחינת הפוטנציאל, מספר העצים לאחר פעולות ה cut יהיה מספר פעולות ה cut +1, ומספר הצמתים המסומנים , יהיה מספר פעולות ה cut פחות 1. ולכן הפוטנציאל יהיה חסום ע"י 3logm כלומר O(logm).

ד-ו.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| decrease Key max cost | potential | Total cuts | Total links | case |
| - | 3logm -1 | logm | m-1 | (c)original |
| - | 3logm-2 | logm-1 | m-1 | (d) decKey(m-2^i) |
| - | m+1 | 0 | 0 | (e)remove line #2 |
| logm-1 | 2logm | 2logm-1 | m-1 | (f) added line #4 |

פירוט תוצאות הטבלה:

**Original**- בהתאם למה שענינו בסעיף לעיל.

**(d) decKey(m-2^i)** – מספר פעולות ה links אין שינוי משום שפעולת ה decreaskey לא משפיעה על מספר הקישורים.

מבחינת פעולות ה cuts- בהרצה זו לעומת ההרצה הראשונה, ביצענו את פעולת decrease key הראשונה על השורש של הערמה. ולכן לא ביצענו פעולת cut בפעולת decrease key זו, ולכן יש לנו פעולת cut אחת פחות.

מבחינת הפוטנציאל, משום שביצענו cut אחד פחות, מספר העצים קטן באחד מההרצה המקורית. בנוסף, מספר הצמתים המסומנים, נשאר זהה, מכיוון שהוא השורש.

**(e)remove line #2 –**

מבחינת פעולות ה cut ופעולות ה link, לא קיבלנו שום ערך, משום שלא ביצענו פעולת delete min, ועל כן לא ביצענו חיבורים ולכן גם לא ביצענו פעולת cut בdecreasekey משום שכולם היו שורשים בערימה.

מבחינת הפוטנציאל, יש לנו m+1 עצים, שאף אחד מהם לא מסומן, ולכן הפוטנציאל הוא m+1.

מבחינת מספר פעולות ה links אין שינוי משום שפעולת ה decreaskey לא משפיעה על מספר הקישורים.

מבחינת מספר פעולות ה cut- משום שהצומת שכעת אנו מבצעים עליה את פעולת decreaskey, נמצא ברמה 1 מעל העלים, ברגע שנבצע עליה את הפעולה, נביא לשרשרת של חיתוכים כגובה הערמה, לכן מספר החיתוכים גדל ב logm-1. שזהו גובה הערמה.

מבחינת הפוטנציאל- כעת מספר העצים גדל ב logm+1, אך מספר הצמתים המסומנים, קטן ב logm, ולכן מהגדרת הפוטנציאל, הפוטנציאל קטן ב logm-1, ולכן קיבלנו, 2logm.

**(f) added line #4**

מבחינת מספר פעולות ה links אין שינוי משום שפעולת ה decreaskey לא משפיעה על מספר הקישורים.

מבחינת מספר פעולות ה cut- משום שהצומת שכעת אנו מבצעים עליה את פעולת decreaskey, נמצא ברמה 1 מעל העלים, ברגע שנבצע עליה את הפעולה, נביא לשרשרת של חיתוכים כגובה הערמה, לכן מספר החיתוכים גדל ב logm-1. שזהו גובה הערמה.

מבחינת הפוטנציאל- כעת מספר העצים גדל ב logm+1, אך מספר הצמתים המסומנים, קטן ב logm, ולכן מהגדרת הפוטנציאל, הפוטנציאל קטן ב logm-1, ולכן קיבלנו, 2logm.

שאלה 2

א.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| potential | Total cuts | Total links | Run time (ms) | m |
| 6 | 0 | 723 | 1 | 728 |
| 6 | 0 | 6555 | 4 | 6560 |
| 9 | 0 | 59040 | 30 | 59048 |
| 10 | 0 | 531431 | 120 | 531440 |
| 14 | 0 | 4782955 | 2290 | 4782968 |

ב.

זמן הריצה האסימפטוטי עבור רצף הפעולות הנ"ל הינו .O(mlogm +m) זאת מכיוון שאנו מתחילים מרצף של m+1 הכנסות, עם המתודה insert שמתבצעת בזמן קבוע .O(1)

סה"כ עד כה קיבלנו סיבוכיות לינארית .O(m) לאחר מכן, מתבצעת הקריאה הראשונה לdeleteMin וכפי שראינו בהרצאה, הסיבוכיות של מתודה זו הינה לינארית .O(m) לאחר קריאה זו נישאר עם ערימה בינומית תקינה, ולכן שאר הקריאות לdeleteMin יהיו בעלות מינימלית לוגריתמית, כלומר . O(logm). וכאמור פעולה זו מתבצעת פעמים. סך הכל קיבלנו שהסיבוכיות תהיה הסכום של זמני הריצה הנ"ל כלומר .O(mlogm+m)=O(mlogm)

ג.

נבחין כי אנו מכניסים את האיברים לערמה מהקטן לגדול, ולכן בסוף כל ההכנסות אנו נקבל רשימה מקושרת שמתחילה מהאיבר הכי גדול משמאל ומסתיימת עם האיבר הקטן ביותר מימין.  
לאחר פעולת delete-min אחת אנו נבצע קונסולידציה שבסופה, בגלל סדר ההכנסה של האיברים נקבל שהמפתחות הנמוכים הם בעלי הדרגות בנמוכות והמפתחות הגבוהים בעלי הדרגות הגבוהות. אנו נקבל ערמה בינומית תקינה בגודל m.  
נבחין כי לא נבצע יותר פעולות link , מכיוון שכל הפעולות הבאות ימחקו את איבר המינימום, אך לא יבצעו link מכיוון שאנו נמחק צמתים עם דרגות נמוכות שלא יתפרקו לעצים עם דרגות שיש בערמה, כי אם הם היו, הם היו מתחברים בdelete min הראשון.

ולכן, כל פעולות הlink מבוצעות בהתחלה.  
בסוף הdelete-min הראשון יהיו מספר עצים כמספר ה'1'-ים בייצוג הבינארי של m ולכן, מכיוון שאנו נבצע link רק בפעולה הראשונה, מספר הlinks הכולל יהיה:

מכיוון שאנו מתחילים עם ערימה בעלת m צמתים מדרגה 0, ומה שלא חיברנו נשארו עצים ומיוצגים כ'1' במספר הבינארי של m.  
  
מספר פעולות הcut הוא 0, מכיוון שאין decreaseKey.

מכיוון שאין פעולות cut גם אין עצים מסומנים ולכן, הפוטנציאל תלוי רק במספר העצים שיש בסוף כל הפעולות. בסוף הפעולות, מספר הצמתים שיש הוא צמתים, ולכן מספר העצים יהיה כמות ה'1'-ים שיש בייצוג הבינארי של כמות הצמתים בסוף וזה גם הפוטנציאל.