תיעוד חיצוני של המחלקה

## מחלקה – BinomialHeap

### פרמטרים:

* size (int) – מספר האיברים בערמה
* last (HeapNode) – מצביע לאיבר האחרון ברשימת העצים של הערימה
* min (HeapNode) – מצביע לאיבר עם המפתח המינימלי
* MIN\_VALUE (int) – שדה סטטי ששווה למינוס 1 בשימוש של המתודה delete.

### מתודות:

BinomialHeap()

מטרה: בנאי של המחלקה

דרך פעולה: בנאי ריק דיפולטי, מאתחל ערימה חדשה בגודל 0, וש-min ו-last מצביעים על null.

סיבוכיות זמן:

ניתוח סיבוכיות במקרה הגרוע: -

BinomialHeap(int size, HeapNode last, HeapNode min)

מטרה: בנאי של המחלקה

דרך פעולה: בנאי, מאתחל ערימה חדשה לפי הקלט שניתן.

סיבוכיות זמן:

ניתוח סיבוכיות במקרה הגרוע: -

rankToSize(int rank)

מטרה: בהנתן דרגה מחזיר את כמות האיברים

דרך פעולה: מחזיר , שזה מספר האיברים בערימה.

סיבוכיות זמן:

ניתוח סיבוכיות במקרה הגרוע: -

getLast()

מטרה: להחזיר מצביע ל-last

דרך פעולה: גישה לשדה

סיבוכיות זמן:

ניתוח סיבוכיות במקרה הגרוע: -

getMin()

מטרה: להחזיר מצביע ל-min

דרך פעולה: גישה לשדה

סיבוכיות זמן:

ניתוח סיבוכיות במקרה הגרוע: -

setSize(int size)

מטרה: מגדיר את שדה ה-size

דרך פעולה: השמה לשדה

סיבוכיות זמן:

ניתוח סיבוכיות במקרה הגרוע: -

setLast(HeapNode last)

מטרה: מגדיר את שדה ה-last

דרך פעולה: השמה לשדה

סיבוכיות זמן:

ניתוח סיבוכיות במקרה הגרוע: -

setMin(HeapNode min)

מטרה: מגדיר את שדה ה-min

דרך פעולה: השמה לשדה

סיבוכיות זמן:

ניתוח סיבוכיות במקרה הגרוע: -

insert(int key, String info)

מטרה: מכניס איבר בעל מפתח ומידע נתונים לערמה, ומחזיר את ה-HeapItem שנוצר.

דרך פעולה: יוצר HeapItem ו-HeapNode חדשים, ומקשר ביניהם. אם הערמה ריקה אז מגדירים את הצומת החדש שנוצר להיות המינימום וה-last של הערימה, ומגדיר את הגודל של הערימה להיות 1, ולבסוף מחזיר את הצומת הנ"ל.

אחרת, יוצרים ערימה בינומית חדשה, כאשר הצומת החדש הוא הצומת היחידה בערימה. בלולאה על רשימת העצים, מוצא את רצף העצים המקסימלי מדרגות עוקבות (אם רצף העצים המקסימלי הוא כל העצים אז עושים meld של הערימה הראשית עם הערימה בעלת הצומת החדש), ומנתק אותם מהערימה ושם אותם בערימה חדשה subheap. עושה meld של subheap ושל הערימה החדשה, ועם הערימה שנוצרה עושה meld לערימה הראשית. לבסוף מעדכן את הפרמטרים של הערימה הראשית, ומחזיר את הצומת החדש.

סיבוכיות זמן:

ניתוח סיבוכיות במקרה הגרוע: סיבוכיות זמן זהה למונה בינארי. המקרה הגרוע הוא כאשר הערימה הראשית "מלאה", במקרה הזה צריך לרוץ על הלולאה על איברים, ואז קוראים ל-meld שהסיבוכיות שלו ב-worst case היא גם כן . סה"כ עושים עבודה.

deleteMin()

מטרה: מוחקת את המינימום מתוך הערימה (לא מחזירה אותו)

דרך פעולה: המינימום הוא שורש של אחד העצים. לכן תחילה מנתקים אותו מהרשימה המקושרת של העצים (גורמים לכך שהעץ שלפניו יצביע על העץ שאחריו). לאחר מכן מתקנים את השדות של הערימה (שהורדנו ממנה את העץ) כך שהמינימום, הגודל וה-last שלה יהיה נכון (בשלב זה בודקים את היה רק עץ אחד שהוא העץ שהורדנו ולכן הערימה נותרה ריקה בלעדיו). לכן כעת הערימה ללא העץ של המינימום המקורי היא ערימה בינומית תקינה.

לאחר מכן מנתקים את כל הבנים של המינימום ממנו כך שנקבל רשימה מקושרת של צמתים שכל אחד הוא שורש של עץ בינומי מדרגה אחרת ולכן הם מהווים ערימה בינומית בפני עצמם.

לבסוף נבצע meld בין שתי הערימות הללו כדי לקבל את ערימה בינומית אחת תקינה ללא המינימום שהיה.

סיבוכיות זמן:

ניתוח סיבוכיות במקרה הגרוע: ניתוק המינימום משאר העצים דורש להגיע עד לעץ שלפני המינימום ולכן עולה כמו כמות העצים שזה . גם פעולות עדכון הערימה לאחר מחיקת העץ עולה כך כי על מנת למצוא את המינימום החדש יש להשוות בין כל שורשי העצים שנותרו שזה גם .

כעת ניתוק המינימום מהבנים שלו עולה כמו הדרגה של המינימום כי זה כמות הבנים שלו. דרגת המינימום היא לכל היותר ולכן תהליך ניתן כל הבנים עולה

*בניית ערימה מתוך רשימה של עצים בינומיים שונים בדרגתם היא כי זה רק עדכון שדות ומצביעים.*

*כעת ביצוע meld של שתי ערימות עם לכל היותר n איברים עולה*

*אם נסכום את הכל נקבל שהסיבוכיות היא:*

findMin()

מטרה: מחזיר את האיבר המינימלי בערימה

דרך פעולה: קריאה למתודות שפועלת בזמן קבוע.

סיבוכיות זמן:

ניתוח סיבוכיות במקרה הגרוע: -

decreaseKey(HeapItem item, int diff)

מטרה: מקטין את ה-key ב-diff של צומת נתון

דרך פעולה: מקטין את הערך של ה-key ועושה sift up. כלומר, כל עוד המפתח של ההורה גדול מהמפתח של הצומת, מחליפים ביניהם על ידי קריאה למתודה swap שפועלת בזמן קבוע וממשיכים בלולאה.

בסוף הלולאה, אם המפתח החדש של הצומת קטן מהמינימום, קובעים את המינימום להיות הצומת הנוכחי.

סיבוכיות זמן:

ניתוח סיבוכיות במקרה הגרוע: הלולאה במקרה הגרוע תרוץ כמות איטרציות כגובה העץ, כלומר איטרציות. כל הפעולות בתוך ומחוץ ללולאה פועלות בזמן קבוע, אז נקבל סה"כ זמן ריצה של .

delete (node)

מטרה: מקבלת מצביע ל-node הערימה ומוחקת אותו (לא מחזירה אותו)

דרך פעולה: נבצע שתי פעולות: decrease-key למספר שהוא יותר קטן מהמינימום האפשרי של הערימה – 1- כי ידוע שכל איברי הערימה אי שליליים.

כעת אנו יודעים שהאיבר שרצינו למחוק הוא המינימום ועל כן נבצע deleteMin על מנת למחוק אותו

סיבוכיות זמן:

ניתוח סיבוכיות במקרה הגרוע: שתי הפעולות decrease-key ו-deleteMin עולות ולכן זו גם הסיבוכיות שלהן יחד.

meld(BinomialHeap heap2)

מטרה: מיזוג של 2 ערמות

דרך פעולה: מטפלים במקרי קצה, אם הערימה השנייה (או שתי הערימות) ריקה אז מסיימים. אם הערימה המקורית ריקה אז קובעים שהערימה השנייה היא הערימה הראשית.

ראשית, קובעים שהגודל של הערימה הוא סכום גודלי הערימות. קוראים ל-getTrees על שתי הערימות כדי להשיג מערכים של העצים בכל ערימה. בנוסף מאתחלים משתנה carry מסוג HeapNode. עוברים בלולאה על הדרגות, ובכל איטרציה מחברים שני עצים עם שארית, כלומר עושים link ללכל היותר שני העצים במקרה שאין carry. במקרה שיש carry אז עושים link ללפחות עץ אחד, ובמקרה שיש שלושה, משאירים את העץ מהערימה המקורית במקום וקובעים שה-carry החדש הוא link של העץ מהערימה השנייה וה-carry.

לאחר מכן, קובעים שהעץ האחרון במערך העצים החדשה הוא ה-last. ואז עוברים בלולאה על מערך העצים שעברו link וקובעים שהם העצים בערימה. ולבסוף, קוראים ל-updateMin שפועל בזמן .

סיבוכיות זמן:

ניתוח סיבוכיות במקרה הגרוע: נסמן את מספר האיברים בערימה השנייה ב-m ונניח בה"כ כי . הקריאות ל-getTrees לכל ערימה לוקחות ו- זמן. הלולאה הראשית עושה איטרציות, ובכל איטרציה עושה פעולות שלוקחות זמן קבוע. לאחר מכן מבצעים עוד לולאה של איטרציות כדי לחבר מחדש את העצים בערימה, ולבסוף קוראים ל-updateMin שפועלת בזמן . סה"כ נקבל זמן ריצה של .

getTrees(int len)

מטרה: מחזירה מערך באורך נתון של העצים בערימה

דרך פעולה: מאתחלת מערך בגודל len ריק. עוברת בלולאה על כל העצים בערימה, ומכניסה כל עץ למערך לפי הדרגה של העץ

סיבוכיות זמן:

ניתוח סיבוכיות במקרה הגרוע: הלולאה רצה על כל העצים בערימה, כלומר מבצעת איטרציות, בכל איטרציה עושה פעולות בזמן קבוע. סה"כ נקבל זמן ריצה של .

updateMin()

מטרה: מעדכנת את המינימום של הערימה

דרך פעולה: מאתחלת משתנה cur\_min, רצה בלולאה על העצים בערימה ובכל איטרציה בודקת אם השורש של העץ קטן מהמינימום הנוכחי, אם כן מעדכנת. לבסוף קובעת את המינימום להיות הצומת cur\_min.

סיבוכיות זמן:

ניתוח סיבוכיות במקרה הגרוע: הלולאה רצה על כל העצים בערימה, כלומר מבצעת איטרציות, בכל איטרציה עושה פעולות בזמן קבוע. סה"כ נקבל זמן ריצה של .

size()

מטרה: מחזירה את הגודל של הערימה

דרך פעולה: גישה לשדה

סיבוכיות זמן:

ניתוח סיבוכיות במקרה הגרוע: -

empty()

מטרה: מחזירה True אם הערימה ריקה, False אחרת.

דרך פעולה: קריאה למתודה שפועלת בזמן קבוע ומשווה.

סיבוכיות זמן:

ניתוח סיבוכיות במקרה הגרוע: -

numTrees()

מטרה: מחזירה את מספר העצים בערמה

דרך פעולה: קוראת לפונקציה המובנית bitCount עם הגודל של הערימה, כאשר bitCount(n) מחזירה את מספר האחדות בייצוג הבינארי של n.

סיבוכיות זמן:

ניתוח סיבוכיות במקרה הגרוע: -

highest\_rank()

מטרה: מחזירה את הדרגה הכי גבוהה בערימה

דרך פעולה: מבצעת כאשר .

סיבוכיות זמן:

ניתוח סיבוכיות במקרה הגרוע: -

link(HeapNode x, HeapNode y)

מטרה: מחברת בין שני צמתים (שורשים) של עצים, ומחזירה את השורש של העץ שנוצר.

דרך פעולה: אם המפתח של צומת x גדול מהמפתח של צומת y, מחליף ביניהם. אם ל-x אין בן, אז ה-next של y יהיה y. אחרת, ה-next של y יהיה ה-next של הבן של x, וה-next של הבן של x יהיה y. אז קובעים את הבן של x להיות y, קובעים את ההורה של y להיות x, ומעלים את הדרגה של x ב-1. ומחזירים את x

סיבוכיות זמן:

ניתוח סיבוכיות במקרה הגרוע: -

## מחלקה – HeapNode

### פרמטרים:

* item (HeapItem) – מצביע לאיבר שהצומת מייצג
* child (HeapNode) – מצביע לבן של הצומת
* next (HeapNode) – מצביע לאיבר הבא של הצומת
* parent (HeapNode) – מצביע להורה של הצומת
* rank (int) – הדרגה של הצומת

### מתודות:

HeapNode()

מטרה: בנאי דיפולטי למחלקה

דרך פעולה: מגדיר את כל השדות להיות null.

סיבוכיות זמן:

ניתוח סיבוכיות במקרה הגרוע: -

HeapNode(HeapItem item)

מטרה: בנאי למחלקה, מאתחל צומת עם מידע וללא קשרים

דרך פעולה: מגדיר את השדה item לפי הקלט, ושאר השדות מגדיר כ-null.

סיבוכיות זמן:

ניתוח סיבוכיות במקרה הגרוע: -

swap(HeapNode node)

מטרה: מחליף בין הצומת הנוכחי לצומת נתון

דרך פעולה: משנה מצביעים.

סיבוכיות זמן:

ניתוח סיבוכיות במקרה הגרוע: -

getItem()

מטרה: מחזיר את ה-item של הצומת

דרך פעולה: גישה לשדה

סיבוכיות זמן:

ניתוח סיבוכיות במקרה הגרוע: -

getChild()

מטרה: מחזיר את הבן של הצומת

דרך פעולה: גישה לשדה

סיבוכיות זמן:

ניתוח סיבוכיות במקרה הגרוע: -

getNext()

מטרה: מחזיר את ה-next של הצומת

דרך פעולה: גישה לשדה

סיבוכיות זמן:

ניתוח סיבוכיות במקרה הגרוע: -

getParent()

מטרה: מחזיר את ההורה של הצומת

דרך פעולה: גישה לשדה

סיבוכיות זמן:

ניתוח סיבוכיות במקרה הגרוע: -

getRank()

מטרה: מחזיר את הדרגה של הצומת

דרך פעולה: גישה לשדה

סיבוכיות זמן:

ניתוח סיבוכיות במקרה הגרוע: -

setItem(HeapItem item)

מטרה: קובע את שדה ה-item של הצומת

דרך פעולה: השמה לשדה

סיבוכיות זמן:

ניתוח סיבוכיות במקרה הגרוע: -

setChild(HeapNode child)

מטרה: מגדיר לצומת בן

דרך פעולה: השמה לשדה

סיבוכיות זמן:

ניתוח סיבוכיות במקרה הגרוע: -

setNext(HeapNode next)

מטרה: קובע את שדה ה-next של הצומת

דרך פעולה: השמה לשדה

סיבוכיות זמן:

ניתוח סיבוכיות במקרה הגרוע: -

setParent(HeapNode parent)

מטרה: מגדיר לצומת הורה

דרך פעולה: השמה לשדה

סיבוכיות זמן:

ניתוח סיבוכיות במקרה הגרוע: -

setRank(int rank)

מטרה: מגדיר דרגה לצומת

דרך פעולה: השמה לשדה

סיבוכיות זמן:

ניתוח סיבוכיות במקרה הגרוע: -

## מחלקה – HeapItem

### פרמטרים:

* node (HeapNode) – מצביע לצומת שהאיבר מייצג
* key (int) – מפתח
* info (String) - מידע

### מתודות:

HeapItem()

מטרה: בנאי דיפולטי של המחלקה

דרך פעולה: מגדיר אובייקט HeapItem עם מפתח 0, ועם info שהוא null.

סיבוכיות זמן:

ניתוח סיבוכיות במקרה הגרוע: -

HeapItem(int key, String info)

מטרה: בנאי למחלקה

דרך פעולה: מגדיר אובייקט HeapItem עם מפתח ו-info נתונים.

סיבוכיות זמן:

ניתוח סיבוכיות במקרה הגרוע: -

getNode()

מטרה: מחזיר את הצומת של האיבר

דרך פעולה: גישה לשדה

סיבוכיות זמן:

ניתוח סיבוכיות במקרה הגרוע: -

getKey()

מטרה: מחזיר את המפתח של האיבר

דרך פעולה: גישה לשדה

סיבוכיות זמן:

ניתוח סיבוכיות במקרה הגרוע: -

getInfo()

מטרה: מחזיר את המידע של האיבר

דרך פעולה: גישה לשדה

סיבוכיות זמן:

ניתוח סיבוכיות במקרה הגרוע: -

setNode(HeapNode node)

מטרה: מגדיר צומת לאיבר

דרך פעולה: השמה לשדה

סיבוכיות זמן:

ניתוח סיבוכיות במקרה הגרוע: -

setKey(int key)

מטרה: מגדיר מפתח לאיבר

דרך פעולה: השמה לשדה

סיבוכיות זמן:

ניתוח סיבוכיות במקרה הגרוע: -

setInfo(String info)

מטרה: מגדיר מידע לאיבר

דרך פעולה: השמה לשדה

סיבוכיות זמן:

ניתוח סיבוכיות במקרה הגרוע: -

חלק תיאורטי

שאלה 1

ניסוי 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| מספר סידורי | גודל מערך | זמן ריצה (מילישניות) | מספר החיבורים הכולל | מספר העצים בסיום | סכום דרגות הצמתים שמחקנו |
| 1 | 728 | 1.9762 | 723 | 5 | 0 |
| 2 | 2186 | 1.238 | 2182 | 4 | 0 |
| 3 | 6560 | 3.4639 | 6555 | 5 | 0 |
| 4 | 19682 | 10.8913 | 19675 | 7 | 0 |
| 5 | 59048 | 30.8587 | 59040 | 8 | 0 |
| 6 | 177146 | 40.4558 | 177134 | 12 | 0 |

ניסוי 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| מספר סידורי | גודל מערך | זמן ריצה (מילישניות) | מספר החיבורים הכולל | מספר העצים בסיום | סכום דרגות הצמתים שמחקנו |
| 1 | 728 | 4.0151 | 3296 | 5 | 2937 |
| 2 | 2186 | 4.0427 | 11489 | 4 | 10400 |
| 3 | 6560 | 6.2799 | 39744 | 5 | 36469 |
| 4 | 19682 | 26.0782 | 135049 | 7 | 125215 |
| 5 | 59048 | 105.3636 | 450934 | 8 | 421418 |
| 6 | 177146 | 334.7665 | 1499759 | 12 | 1411198 |

ניסוי 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| מספר סידורי | גודל מערך | זמן ריצה (מילישניות) | מספר החיבורים הכולל | מספר העצים בסיום | סכום דרגות הצמתים שמחקנו |
| 1 | 728 | 1.9253 | 723 | 5 | 697 |
| 2 | 2186 | 1.8003 | 2182 | 5 | 2156 |
| 3 | 6560 | 4.6755 | 6555 | 5 | 6529 |
| 4 | 19682 | 13.2586 | 19675 | 5 | 19649 |
| 5 | 59048 | 28.3171 | 59040 | 5 | 59014 |
| 6 | 177146 | 79.3916 | 177134 | 5 | 177108 |

שאלה 2

חלק א

ננתח את זמן הריצה לכל אחד מהניסויים.

בניסוי הראשון ביצענו הכנסה בסדר עולה של n איברים. כל הכנסה היא כמו הסיבוכיות של ביצוע increment במונה בינארי. הוכחנו בתרגול כי ביצוע increment למונה בינארי הוא אמורטייזד ולכן נצפה שזמן הריצה עובר ביצוע של כך n פעמים יהיה .

מצד שני ביצענו n איטרציות ולכן זמן הריצה חייב להיות לפחות n, כלומר ולכן נקבל כי הזמן הריצה האסימפטוטי הוא

ניתן לראות כי יחסית זה נשמר בכך שבמעבר בין כל שתי שורות, הגודל גדל בערך פי 3 וגם זמן הריצה גדל בערך בקצב הזה.

בניסוי השני הכנסנו בהתחלה גם n איברים, מה שעולה (אין חשיבות משמעותית לסדר ההכנסה משום שהסדר בין האיברים נבדק רק כאשר מבצעים link ותמיד מבצעים בדיקה זו).

כעת בוצעו מחיקות של המינימום. מכיוון שסדר ההכנסה הוא רנדומלי לא ניתן לדעת באיזה עץ נמצא המינימום (שהוא השורש של עץ זה). עלות המחיקה מערימה בגודל n הוא תמיד (תמיד צריך לחפש בכל העצים את המינימום ולכן זה לפחות (ולכן עלות כל המחיקות מ-n ועד היא:

*ומצד שני:*

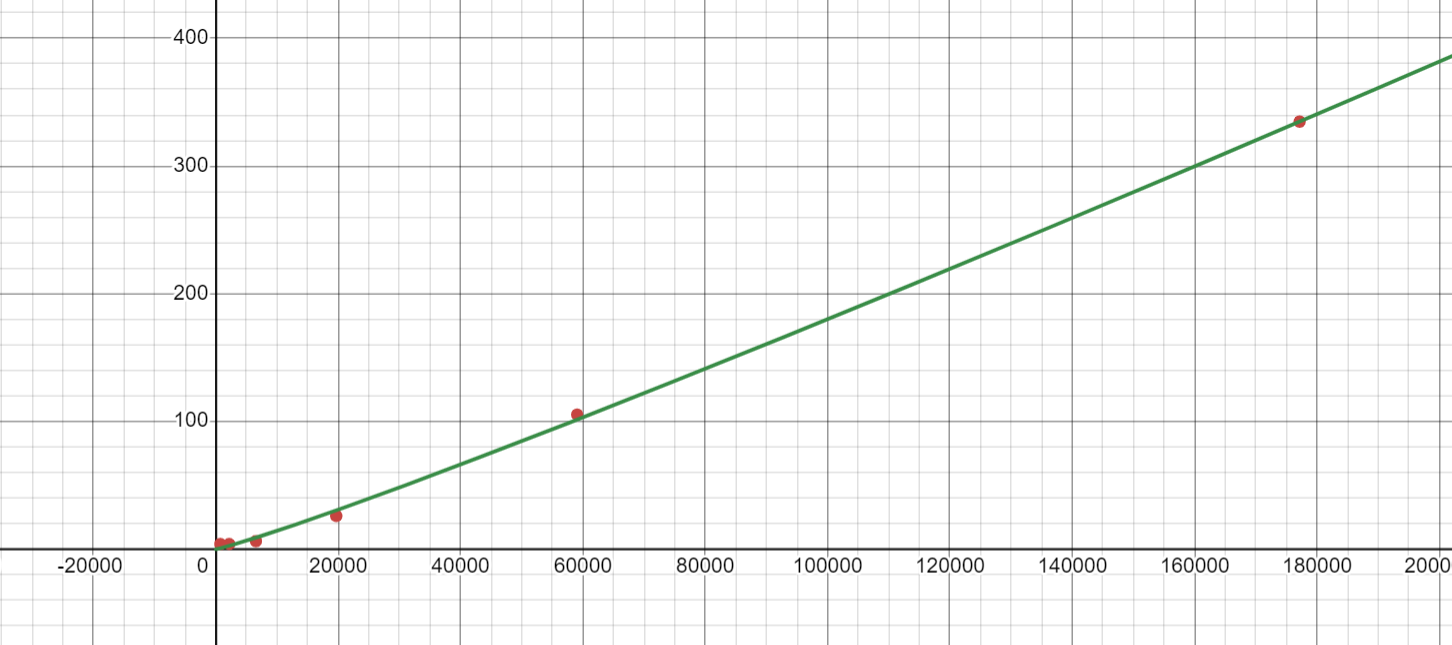
*לכן עלות כל המחיקות היא*

*ההכנסה עולה ולכן יחד הסיבוכיות היא*

*אכן, תוצאות זמן הריצה של ניסוי 2 מתאימות לקצב הגידול הזה.*

*בגרף להלן סימנו את זמני הריצה בניסוי 2 כתלות בגודל n והגרף הירוק הוא של הפונקציה*

*ניתן לראות כי הגרף די תואם לנקודות של המדידות, כלומר אכן מתקיים שזמן הריצה הוא .*



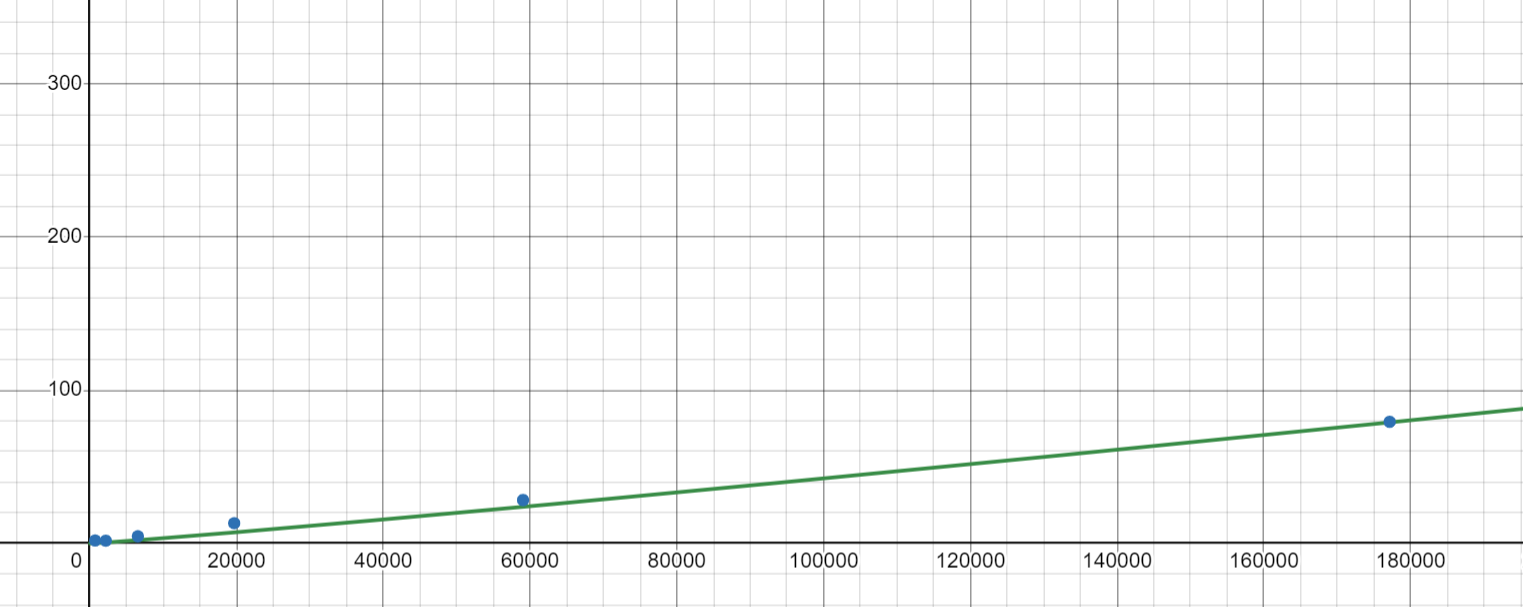
*בניסוי השלישי מבצעים n הכנסות ו- הוצאות, מגודל n עד גודל 31.*

*לכן עלות ההכנסות היא ועלות ההוצאות היא:*

*השתמשנו בכך שראשית לכל n שהיה בניסוי ולאחר מכן השתמשנו בתוצאת הסכום הזה שכבר ניתחנו בסעיף קודם.*

*כך קיבלנו כמו קודם כי עלות המחיקות היא ולכן כמו קודם זו הסיבוכיות של ניסוי זה.*

*שוב ניתן לראות שזה תואם את תוצאות זמן הריצה שיצאו בפועל בניסוי זה אם נציג זאת על הגרף:*



*כאשר הגרף הירוק הוא .*

חלק ב

*כעת נתאר את הקשר בין מספר החיבורים הכולל, מספר העצים בסיום וסכום דרגות העצים שמחקנו:*

*נשים לב ראשית כי בכל רגע נתון, כאשר יש n איברים בערימה אז מתקיים:*

*זאת משום שניתן לחשוב על כל לינק בתור קשת שמגיעה לבן (כל לינק מגיע לבן אחד בדיוק) ולכן מספר הלינקים הוא מספר הבנים שיש בערימה. הצמתים היחידים שהם אינם בנים בערימה הם שורשי העצים וכאלו יש בדיוק כמספר העצים בערימה.*

*כעת בכל פעם שמבצעים מחיקה מוחקים גם לינקים. כמות הלינקים שנמחקים מהעץ הם בדיוק הדרגה של השורש שמחקנו (כי זה כמות הבנים שיש לו). עם זאת בתהליך ה-meld נעשה עוד לינקים.*

*אנו יודעים כי:*

*לכן:*

*נסמן ב-new\_links את כמות הלינקים החדשים שנוצרים בתהליך ה-meld.*

*לכן:*

*שזה:*

*אם נבצע סכימה על פני כל המחיקות נקבל (נשים לב שיש לנו טור טלסקופי באגף השמאלי):*

*לכן נקבל:*

*וכאמור שמאל הוא כל הלינקים שעשינו לאורך כל התהליך ולכן:*

*כזכור הקשר בין גודל הערימה למספר העצים הוא שמספר העצים הוא מספר ה-1ים בייצוג הבינארי של גודל הערימה ולכן נקבל:*

*כלומר הקשר בין שלושת הגורמים הוא:*

*כאשר הפונקציה count\_ones מחזירה כמה פעמים מופיעה הספרה 1 בייצוג הבינארי של המספר שקיבלה כקלט.*

חלק ג

*לסיום, נחקור את סכום דרגות הצמתים שמחקנו בניסוי השלישי. בניסוי זה הכנסנו את הצמתים בסדר יורד. לכן המינימום הוא תמיד השורש של העץ הכי קטן בכל רגע נתון כי זהו סדר ההכנסה שלהם. ביצוע מחיקה של המינימום הוא למעשה שחזור מדויק אחורה של סדר ההכנסה כי אנו מוציאים בסדר הפוך מההכנסה (מכניסים בסדר יורד ומוציאים בסדר עולה כי מוציאים כל פעם את המינימום).*

*לכן בכל מחיקה של המינימום, הדרגה שלו היא הדרגה הנמוכה ביותר של העצים של הערימה באותו רגע.*

*נשים לב שבגלל שזה המצב, שאר העצים מדרגה יותר גבוהה משל המינימום ולכן לאחר שמוחקים אותו ומפרקים את הבנים שלו לרשימה של עצים, דרגתם קטנה מהדגרה המקורית של העץ שלהם, שהיא הייתה קטנה משאר העצים ולכן למעשה ה-meld שעושים בזמן המחיקה לא באמת מחבר את העצים בכלל וכל צומת שלא נמחק, נשאר באותה דרגה לאחר המחיקה של המינימום. לכן לאורך כל המחיקות, כל צומת נשאר עם הדרגה שלו עד לרגע שהוא נמחק.*

*אנו מבצעים מחיקות. לכן אנו מוחקים את האיברים 1 עד . לפיכך סך דרגות הצמתים שמחקנו הוא סך הדרגות של צמתים אלו בערימה שבנינו בהתחלה.*

*כלומר במצב שאחרי הבנייה של הערימה ולפני כל המחיקות, הערך המבוקש לחשב הוא*

*סכום של כל הדרגות של צמתים בעץ בינומי הוא סכום מספרי הבנים של כל צומת (כי הדרגה של צומת שווה למספר הבנים שלו) וזה מספר הצמתים בעץ פחות 1 (כי השורש איננו בן של אף אחד ולכן לא נספר).*

*לכן סך הדרגות בעץ מדרגה הוא*

*לכן סך הדרגות בערימה בינומית הוא כמות הצמתים פחות כמות העצים משום שכמות העצים מייצגת את מספר השורשים שהם צמתים שהם לא בנים ולכן לא נספרים בסכום הדרגות.*

*לכן סך הדרגות לאחר n הכנסות הוא (שמספר העצים הוא מספר האחדות בייצוג הבינארי של n).*

*סך הדרגות של צמתים 1 עד שווה לסך דרגות הצמתים מ-1 עד n פחות סך דרגות הצמתים מ- עד n.*

*נשים לב שצמתים אלו הם 31 הצמתים הראשונים שהכנסנו לערימה. לכן ברגע מסוים, לאחר הכנסה של 31 איברים סך הדרגות של הצמתים האלו הוא גודל הערימה פחות מספר העצים שלה שזה שווה ל-. מכיוון שהסברנו שדרגות הצמתים לא משתנות עם המחיקות אז סכום הדרגות של הצמתים האלו נשאר כך לאורך כל המחיקות.*

*לכן, סך הדרגות של הצמתים שמחקנו הוא:*

*לדוגמה עבור נקבל ומספר האחדות בייצוג הבינארי של n הוא 8 ולכן סך הדרגות הנמחקות הוא:*

*שזו בדיוק התוצאה שיצאה בניסוי.*