ערימה בינומית

פרטים :

name1 – Mahajna Mohamad
id1 – 324887488
username1 – mahajna3

name2 – Amana Essa
id2 - 213127905
username2 – amanaessa

המחלקה HeapItem

במחלקה זו אין מתודות רק בנאים :

1. **public HeapItem(int key, String info, HeapNode node)**

   בנאי שמקבל מפתח ,ומידע info מסוג מחרוזת ו צומת מסוג Heap node.

   הבנאי עושה השמה לערכים שהוא קיבל .

   סיבוכיות זמן : $O(1)$

   ניתוח סיבוכיות : כל השמה לוקחת זמן קבוע , יש 3 כאלה לכן סה"כ זמן קבוע .

2. **Public HeapItem()**

   בנאי ריק שמאתחל את המפתח ל -1 , זאת אינקציה לכך ש Item עדיין לא מאותחל .

   סיבוכיות זמן : $O(1)$

   ניתוח סיבוכיות : יש השמה אחת לערך המפתח , זה לוקח זמן קבוע .

המחלקה HeapNode

במחלקה זו אין מתודות רק בנאים :

1. **public HeapNode(int key, String info)**

   בנאי שמקבל מפתח ,ומידע info מסוג מחרוזת.

   הבנאי עושה השמה לערכים שהוא קיבל .

   סיבוכיות זמן : $O(1)$

ניתוח סיבוכיות : כל השמה לוקחת זמן קבוע , יש 2 כאלה לכן סה"כ זמן קבוע .

2. **Public HeapNode()**

בנאי ריק שמאתחל את ה rank ל -1 , זאת אינקציה לכך שהצומת עדיין לא מאותחל .

סיבוכיות זמן : $O(1)$

ניתוח סיבוכיות : יש השמה אחת לערך המפתח , זה לוקח זמן קבוע .

המחלקה BinomialHeap

Public BinomialHeap() :

בנאי שמאתחל את הגודל ל 0 והשדות last , min ל null .

סיבוכיות זמן : $O(1)$

ניתוח סיבוכיות :השמות הזמם קבוע .

public HeapNode link(HeapNode x, HeapNode y):

הפונקציה מקבלת שני צמתים ומחברת אותם כפי שנלמד בכיתה ומעדכנת את השדות : , child rank,parent של הצומת x , הפונקציה מוודא שהצומת x בעל מפתח מינימלי ע"י קריאה לעצמה עם שינוי פרמטרים אם במקרה שהמפתח של x גדול משל y, בסוף מחזירים את x .

סיבוכיות זמן : $O(1)$

ניתוח סיבוכיות : הפונקציה מחברת בין הצמתים בזמן קבוע ומעדכנת את השדות בזמן קבוע אם המפתח של x גדול ממפתח של y , אחרת קוראת לעצמה עם החלפת פרמטרים , ולכן סה"כ ביצענו פעולות בזמן קבוע .

public BinomialHeap Insert_Helper(HeapNode node):

הפונקציה מקבלת צומת מסוג HeapNode , מאתחלת ערימה בינומית ריקה ומעדכת את השדות size , last, min ,next ומחזירה את הערימה הבינומית מגודל 1 (B0) .

סיבוכיות זמן : $O(1)$

ניתוח סיבוכיות : הפונקציה מאתחלת ערמה בינומית ריקה בזמן קבוע , מעדכת 4 שדות שכל אחד מהם לוקח זמן קבוע ולכן סה"כ הסיבוכיות היא קבועה .

public HeapItem insert(int key, String info):

הפונקציה מקבלת מפתח ומידע , מאתחלת צומת חדש מסוג HeapNode וקוראת ל insert_helper ועושה meld לערימה הראשית עם הערימה המוחזרת מהפונקציה insert_helper . לבסוף הפונקציה מחזירה node.item .

סיבוכיות זמן : $O(\log n)$

ניתוח סיבוכיות : הפונקציה מאתחלת צומת בזמן קבוע , קוראת לפונקצית העזר שלוקחת זמן קבוע , אחר כך עושים meld שלוקח $O(\log n)$ ומחזירים את item בזמן קבוע ולכן סה"כ הסיבוכיות היא לוגריתמית.

public HeapItem findMin():

הפונקציה בודקת אם הערימה ריקה אז מחזירים null , אחרת מחזירים min.item.

סיבוכיות זמן : $O(1)$

ניתוח סיבוכיות : הבדיקה בהתחלה מתבצעת בזמן קבוע , אם הערימה לא ריקה ניגשים לשדה this.min ומחזירים item וזה לוקח זמן קבוע.

public void decreaseKey(HeapItem item, int diff) :

הפונקציה מקבלת item ו diff שמסמן בכמה להפחית את ערך המפתח , הפונקציה מפחיתה את הערך של המפתח , אחר כך "נתקן" את הערימה ע"י לולאה כך שעולים למעלה להורה של הצומת אם המפתח גדול מהמפתח של ההורה ומחליפים ביניהם , אם לא אז נצא מהלולאה . בסוף נבדוק אם עלינו לעדכן את שדה ה min .

סיבוכיות זמן : $O(\log n)$.

ניתוח סיבוכיות : במקרה הגרוע עלינו לעלות עד השורש ז"א עלינו לעלות $O(\log n)$ פעמים כגובה של העץ הבינומי , בכל שלב מבצעים זמן קבוע , בנוסף הפחתת ערך המפתח והבדיקה בסוף דורש זמן קבוע ולכן סה"כ נקבל $O(\log n)$.

public void deleteMin() :

בהתחלה נבדוק שני מקרים מיוחדים , אם הערימה ריקה לא נעשה כלום , אם הערימה יש בה רק צומת אחד אז פשוט נעדכן השדות min,last = null , size=0 .

אחרת , אם למינימום יש בן אז נחפש את המינימום החדש מהילדים של הצומת , נגדיר ערימה חדשה כדי לשמור את התת עץ של הצומת שנמחק , נעדכן את הגודל שלה, אחר כך נבדוק אם הערימה היתה עץ בינומי אחד ונעדכן שדות , אחר נקרא ל delete_min_helper כדי למחוק את הצומת ואחר כך נעשה *meld*.

אחרת נקרא ל delete_min_helper.

סיבוכיות זמן : $O(\log n)$.

ניתוח סיבוכיות : במקרה הגרוע הפונקציה נכנסת ללואה כדי לחפש את **המינימום מילדים של הצומת הנמחק** , זה לוקח זמן לוגריתמי , אחר כך אם נקרא לפונקציה delete_min_helper אז יתווסף לזמן הריצה $O(\log n)$ ובסוף נקרא ל meld שלוקחת $O(\log n)$ ולכן סה"כ נקבל $O(\log n) = 3O(\log n)$.

private void delete_Min_helper():

זאת פונקציית עזר של delete_min , אנחנו נקרא לפונקציה אחרי שמצאנו את האיבר המינימלי , הפונקציה מוחקת אותו מהערימה ומעדכנת את המינימום החדש ואת הגודל של הערימה .

סיבוכיות זמן : $O(\log n)$.

ניתוח סיבוכיות : במקרה הגרוע הפונקציה נכנסת ללואה כדי לחפש את **המינימום החדש** (שים לב להבדל בין הפומקציה הזו לקודמת) , זה לוקח זמן לוגריתמי , אחר כך נעדכן את השדות כמו הגודל ומצביעים אלה לוקחים זמן קבוע ולכן סה"כ נקבל $O(\log n)$ *

public void delete(HeapItem item):

הפונקציה מקבלת Item , בהתחלה נבדוק אם Item == null אז לא נעשה כלום , אחרת נקרא לפונקציה decrease_key עם diff = item.key+2, אז תמיד המפתח הוא 2- אחרי הקריאה הזו .

כיוון שהמפתחות אינן שליליות (כתוב בשלד) אז בוודאות הצומת הזה בעל מפתח מינימלי נמחק אותו ע"י קריאה ל delete_min .

סיבוכיות זמן : $O(\log n)$.

ניתוח סיבוכיות : נקרא לפונקציה decrease key ו delete_min , כל אחת מהם לוקחת $O(\log n)$ ולכן סה"כ: $O(\log n) = 2O(\log n)$.

public int size():

הפונקציה מחזירה את שדה size

סיבוכיות זמן: $O(1)$

ניתוח סיבוכיות : מחזירים את השדה בזמן קבוע.

public boolean empty():

בודקים אם שדה הsize שווה ל 0 .

סיבוכיות זמן :$O(1)$

ניתוח סיבוכיות : בדיקה פשוטה בזמן קבוע.

public int numTrees():

עוברים על השורשים ומעלים את הcounter באחד בכל פעם שביקרנו בשורש , נעצור כאשר השלמנו מעגל

סיבוכיות זמן :$O(\log n)$

ניתוח סיבוכיות : נעבור על השורשים , יש $O(\log n)$ שורשים , כל שאר הפעולות לוקחות זמן קבוע ולכן סה"כ :$\log(n) * O(1) = O(\log n)$.

public void meld(BinomialHeap heap2):

הפונקציה ממזגת שתי ערימות בינומיות ביעילות. היא משווה את האיברים המינימליים של שתי הערימות, מעדכנים את המינימום עבור הערימה הנוכחית, ממזג את העצים של הערמות בדרגה שווה, ומטפל במקרים שבהם ערימה אחת כוללת עצים בדרגה נמוכה יותר. לבסוף, הפונקציה מתאימה מצביעים ומעדכנת את גודל הערימה הממוזגת.

סיבוכיות זמן :$O(\log n)$

ניתוח סיבוכיות : בתחילה, הקוד בודק ערימות ריקות ומשווה את האלמנטים המינימליים, זה לוקח זמן קבוע. החלק העיקרי של האלגוריתם כולל איטרציה בין העצים של שתי הערמות כדי למזג אותם.

איטרציה זו מוגבלת במספר העצים בערימה הגדולה יותר, כלומר $O(\log n)$ במקרה הגרוע .

בתוך כל איטרציה, מיזוג עצים עם אותה דירוג או טיפול במקרים שבהם יש ערימה אחת עם עצים בדרגה נמוכה יותר כרוך בפעולות זמן קבועות, כגון עדכון מצביעים ובדיקת דרגות. כתוצאה מכך, סיבוכיות הזמן הכוללת של פעולת המיזוג היא : $O(\log n)$

חלק שני

ניסוי 1

מספר סידורי	זמן ריצה (מילישניות)	מספר החיבורים הכולל	מספר העצים בסיום	סכום דרגות הצמתים שמחקנו
1	0.6	723	5	0
2	0.8	2182	4	0
3	1.3	6555	5	0
4	1.7	19675	7	0
5	6.6	59040	8	0
6	18.3	177134	12	0

ניסוי 2

- **הערה** : מספר החיבורים והדרגות לא שלמים כי עשינו את הניסוי כמה פעמים ועשינו ממוצע

מספר סידורי	זמן ריצה (מילישניות)	מספר החיבורים הכולל	מספר העצים בסיום	סכום דרגות הצמתים שמחקנו
1	0.9	3275.9	5	2916.9
2	1.9	11498.7	4	10409.7
3	4	39794.7	5	36519.7
4	12.6	135194.3	7	125360.3
5	42.9	450835.5	8	421319.5
6	147.3	1499138.8	12	1410577.8

ניסוי 3

מספר סידורי	זמן ריצה (מילישניות)	מספר החיבורים הכולל	מספר העצים בסיום	סכום דרגות הצמתים שמחקנו
1	0.7	723	5	697
2	1.3	2182	5	2156
3	3.1	6555	5	6529
4	4.5	19675	5	19649
5	15.3	59040	5	59014
6	41.5	177134	5	177108

שאלה 2

- **ניסוי ראשון:**
  - **סיבוכיות-$O(n)$:**

נבצע n פעולות insert , בתוך ה insert הפעולה היחידה שתדרוש סיבוכיות שהיא לא קבועה היא פעולת ה-meld. נדרש לבצע פעולות meld שסך העבודה בהם יהיה זהה לכמות הפעולות הנדרשות בהגדלת מונה בינארי n פעמים ב-1. וראינו בתרגול שביצוע n פעולות של הגדלת מונה בינארי ב-1 היא לינארית, ולכן הסיבוכיות היא לינארית.

משוואה:

$$num\ of\ nodes\ in\ the\ end = num\ of\ links +\ num\ of\ trees$$

  - **נימוק :**

נסמן את כמות הצמתים ב- n.

נשים לב שמספר הקשתות מייצג את כמות החיבורים שיש בגרף, כי כל לינק שמבוצע מוסיף קשת לגרף, שכן מחברים שני עצים בינומיים מגודל זהה בעזרת קשת אחת .

בנוסף, בכל עץ כמות הקשתות היא ככמות הצמתים בעץ פחות אחד כי לכל צומת יש קשת אחת ויחידה לאבא שלה, ולשורש אין אבא, ולכן אין קשת כלפי מעלה ומכיוון שיש רק שורש אחד אז נקבל את כמות הצמתים בעץ פחות אחד.

לכן, ניקח את מספר הקשתות, ונוסיף את מספר השורשים בערימה נגיע למספר הצמתים הכולל. כלומר, בהינתן k עצים אז כמות השורשים היא k בהתאמה, ומההבחנה שביצענו למעלה כמות הקשתות היא n-k שזוהי גם כמות הלינקים ולכן נקבל המשוואה למעלה.

- **ניסוי שני:**
  - **סיבוכיות -** $\boldsymbol{O(n\log n)}$ **:** הכנסה של n כפי שניתחנו בניסוי 1 היא O(n). כעת נבצע $\frac{n}{2}$ מחיקות.

    בכל מחיקה, נצטרך למצוא מינימום חדש מבין כלל העצים האחרים, כמות העצים במקרה הגרוע ביותר היא מסדר לוגריתמי במספר האיברים ולכן במקרה הגרוע, מציאת המינימום בכל מחיקה היא בסיבוכיות לוגריתמית.

    בנוסף, במידה ולמינימום יש ילדים, נמצא את המינימלי מבין הילדים (במקרה הגרוע גם כן לוגריתמי ב-n) ולאחר מכן לבצע פעולת Meld עם שאר העצים בערימה. פעולה זו היא גם כן מסיבוכיות לוגריתמית לכל היותר.

    לכן הסיבוכיות של כל מחיקה תהייה מסדר לוגריתמי ובסה״כ מכיוון שמבצעים $\frac{n}{2}$ מחיקות,

    נקבל כי סיבוכיות הניסוי היא $O(n\log n)$.

משוואה:

$$num\ of\ nodes\ in\ the\ end = \frac{n}{2} = (num\ of\ links - \ ranks\ deleted) + \ num\ of\ trees$$

נימוק :

- כמות הצמתים לפני המחיקות מיוצגת ע״י n. נסביר מדוע מספר הלינקים הכולל פחות סכום הדרגות שנמחקו שווה ממש למספר הקשתות בעץ לאחר המחיקות.
- נשים לב ש מספר הלינקים הוא כמספר הקשתות שנוספו לעץ בסך הכל (מתבסס על הניתוח של ניסוי 1). בנוסף נשים לב שכאשר מוחקים צומת מדרגה k, מוחקים k קשתות (לפני ביצוע פעולת meld). לכן, מספר הקשות בעץ לאחר המחיקה הוא כמספר הקשתות שהוספנו (מס׳ הלינקים) פחות מספר הקשתות שמחקנו (דרגות הצמתים שנמחקו).

  אז, לאחר מחיקת $\frac{n}{2}$ צמתים קיבלנו כי מספר העצים בסוף הוא $\frac{n}{2}$ ולפי המשוואה מניסוי 1 נקבל כי מספר הצמתים בסיום הוא כמספר הקשתות בסיום ועוד מספר העצים.

- **ניסוי שלוש:**
  - **סיבוכיות -** $O(n\log n)$ **:**

    ניתוח : הכנסה של n כפי שניתחנו בניסוי 1 היא $O(n)$. כעת נבצע $n-(2^5-1)$ מחיקות. למרות, שהמינימום הוא תמיד האיבר השמאלי ביותר בעץ הקטן ביותר, הקוד עובר על כלל השורשים של הערימה על מנת למצוא את המינימום ולכן יידרש בזמן לוגריתמי על מנת למצוא את המינימום החדש, ומכאן הניתוח יהיה זהה לניסוי השני.

  - **נימוק הדרגות שנמחקו:** סכום הדרגות הוא כמספר הקשתות שנמחקו.
    - בערימה מגודל 2^5-1 איברים יש חמישה עצים בינומיים בערימה מדרגות 0-4. ולכן מהמשוואה נובע כי מספר הקשתות בערימה הסופית הוא 2^5-6 .

      בהכנסה הדטרמיניסטית ההפוכה אנו תמיד מוחקים את הצומת השמאלית ביותר, מכיוון שזה הצומת שנמחק לא נוצרים לינקים חדשים. לכן, מספר החיבורים הכולל הוא מספר החיבורים שביצענו בהכנסה. שהוא כמספר הקשתות בעץ לפני המחיקות.

      כדי להגיע ל-2^5-6 קשתות מהערימה בגודל n נצטרך למחוק קשתות עד שנגיע למספר זה. אזי, כמות הקשתות שנמחק היא $num\ of\ links-(2^5-6)$ שזה בדיוק סכום מספר הדרגות שנמחקו לפי העברת אגפים במשוואה.
  - **משוואה:** הקשר זהה לניסוי השני ולכן המשוואה זהה.

$$num\ of\ nodes\ in\ the\ end=(num\ of\ links-\ ranks\ deleted)+\ num\ of\ trees$$