כריסטיאן שקור

ת.ז 322828518

Class FibonacciHeap

תיאור המחלקה:

זו מחלקה שמיצגת Fibonacci Heap, ויש בה מחלקה פנימית HeapNode .

לכל ערמה יש את השדות:

- **Size:**
  מספר הצמתים בערמה.
- **Min:**
  מצביע על הצומת המינימלית.
- **First:**
  מצביע על הצומת הראשונה.
- **numOfTrees:**
  מספר העצים בערמה.
- **numOfMarked:**
  מספר הצמתים שמסומנים בערמה.

יש גם שני שדות סטטיות:

- **totalLinks:**

מספר כל פעולות החיבור שבוצעו מתחילת ריצת התוכנית.

- **totalCuts:**

מספר כל פעולות החיתוך שבוצעו מתחילת ריצת התוכנית.

מחלקה זו תומכת בפונקציות:

- **Insert, deleleMin, meld,counterRep, delete, decreaseKey, kMin ועוד.**

תיעוד פונקציות:

isEmpty()

- פונקציה זו תבדוק אם הערימה ריקה.
- אם מספר הצמתים בערמה שווה ל-0 אז מחזירים true. אחרת נחזיר false.

.O(1)

insert(int key)

- נוסיף לערמה צומת עם ערך key .
- נאתחל צומת חדשה עם ערך key ונוסיף אותה לתחילת הערמה (שמול) ע"י שינוי מספר קבוע של מצביעים.

.O(1)

deleteMin()

- נמחק את המינימלי ע"י שינוי מספר קבוע מצביעים.
- נחבר בין העצים בערמה לבני המינימלי ע"י שינוי מספר קבוע מצביעים.
- ונבצע successive_linking בסוף.

O(n) in Worst Case

אבל ראינו בהרצאה שזה O(logn) in amortize עבור פונקציית

פוטנציאל מתאימה (#trees+2*#Marked)

- **O(logn+n+1)=O(n) in Worst Case**
- **אבל ראינו בהרצאה שזה O(logn) in amortize עבור פונקציית**
- **פוטנציאל מתאימה (#trees+2*#Marked)**

nullParents_Mark(HeapNode node)

- מנתקים את node והאחים שלו מאביהם, ומוודאים שהם אינן marked.
- עובדים בלולאה עליהן.

O(node.getParent().getRank()):=O(k)

link(HeapNode x,HeapNode y)

- מאחדים בין שני עצים כאשר x,y הן השורשים שלהן.
- נחבר בין הצמתים כך שהקטן מבינם יהיה האב של השני.
- נחבר ע"י שינוי מספר קבוע של צמתים ונעדכן את השדות שצריך לעדכן.

O(1)

successive_linking()

- נחבר בין עצים עד שנקבל ערמה בינומית שהעצים שלה ממוינים מהקטן לגדול.
- נבנה מערך (arr)של צמתים בגודל דרגת העץ הגדולה ביותר שאפשר לקבל אחרי הפעלת הפונקציה(הערך ידוע כאשר יש לנו מספר הצמתים בערמה).
- נעבור בלולאה על העצים בערמה, כל עץ(tree) נכניס אותו למערך במקום ה-tree.rank כלומר: arr[tree.root.rank]=tree.root. O(n)
- אם במקום זה כבר יש עץ אז נחבר ביניהם ונקדם אותם תא אחד במערך.
- נעבור בלולאה על העצים שקיבלנו בסוף תהליך זה (לכל היותר log(n) עצים), ונסדר אותם ע"י שינוי מספר קבוע של מצבעים. O(logn)
- ובדרך מעדכנים את השדות איפה שצריך. O(1)

סך כל :

O(logn+n+1)=O(n) in Worst Case

אבל ראינו בהרצאה שזה O(logn) in amortize עבור פונקציית

פוטנציאל מתאימה (#trees+2*#Marked)

findMin()

- נחזיר null אם הערמה ריקה אחרת נחזיר השדה min.

O(1)

meld (FibonacciHeap heap2)

- נמזג שני הערמות ע"י שינוי מספר קבוע של צמתים ונעדכן את השדות שצריך לעדכן.

O(1)

size()

- נחזיר השדה size.

O(1)

כריסטיאן שקור
ת.ז 322828518

countersRep()

- בלולאה על כל העצים מוצאים את הדרגה המקסימלית בערמה. O(numOfTrees)=O(n)
- מותחלים מערך של מספרים בגודל הדרגה המקסימלית.
- בלולאה על העצים ממלים את המערך (arr[currNode.getRank]+=1) O(n)
- מחזירים את המערך.

O(n+n)=O(n)

delete(HeapNode x)

- משתמשים ב decreaseKey כדי לוודא ש-x הוא המינימלי. O(n) WC/O(1) Amortize.
- deleteMin() מוחקת את x . . O(n) WC/O(logn) Amortize.

O(logn+n)=O(n) in Worst Case

אבל ראינו בהרצאה שזה O(logn) in amortize עבור פונקציית

פוטנציאל מתאימה (#trees+2*#Marked)

decreaseKey(HeapNode x, int delta)

- נבצע x.key=x.key-delta
- אם קיבלנו ערמה חוקית זה הוא, אחרת נבצע cascading_cut

O(logn+n+1)=O(n) in Worst Case

אבל ראינו בהרצאה שזה O(1) in amortize עבור פונקציית

פוטנציאל מתאימה (#trees+2*#Marked)

cut(HeapNode son,HeapNode parent)

- נחתוך את הקשת ונוסיף את העץ של son להתחלה ע"י מספר קבוע של שינוי משתנים.

O(1)

cascading_cut(HeapNode son,HeapNode parent)

- נסיר את הקשת בפעולה cut
- אם האב לא מסומן אז נסמן אותו.
- אחרת צריך למחוק את הקשת בין האב ואביו, נמשיך כך עד שנגיע לשורש או אב לא מסומן.

O(n)

potential()

- כיוון שתחזקנו את השדות numOfTrees/numOfMarked אז רק נחזיר : (#trees+2*#Marked)

O(1)

totalLinks()/totalCuts()

- מחזירים את השדה toatalLinks/totalCuts שתחזקנו אותו בכל המערכת

O(1)

kMin(FibonacciHeap H, int k)

- נחזיר מערך עם ה K האיברים הקטנים ביותר.
- נאתחל ערמת עזר ומערך שבו האיברים המועמדים להיות המינימום החדש אחרי שנמחק המינימום מערמת העזר.
- בלולאה של K אטרציות נמחק המינימלי מערמת העזר, נכניס את הערך שלו למערך שנרצה להחזיר בסוף, נכניס הבנים שלו לערמת העזר וגם למערך העזר, ונחזיק מצביע על האיבר המינימלי במערך העזר.

O(k)*O(H.first.getRank())=O(k*H.first.getRank())

getFirst()

- נחזיר null אם הערמה ריקה אחרת נחזיר השדה first.

O(1)

Class HeapNode

תיאור המחלקה:

מחלקה זו היא פנימית שמייצרת צמתים כדי להשתמש בהם במחלקת האב שלנו (FibonacciHeap).

לכל עצם במחלקה (node) זו יש את השדות:

- Key
- Rank
- Marked
- Child
- Parent
- Next
- Prev

לכל שדה מאלו יש פונקציית get וגם set .

תיעוד פונקציות:

נשים לב שכל הפונקציות במחלקה זו הן Getters and Setters לכן הן מחזירות או מעדכנות שדות, לכן הסיבוכיות שלהן היא O(1) .

- **get/set key**
- **get/set child**
- **get/set parent**
- **get/set next**
- **get/set prev**
- **get/set rank**
- **get/set marked**

כריסטיאן שקור
ת.ז 322828518

חלק תיאורטי

שאלה 1:

א) כיוון שמדובר בסדרת פעולות אזי נחשב לפי זמן amortized :

- m+1 פעולות insert וכל אחת לוקחת O(1).
- פעולת deleteMin אחת שלוקחת O(logm) amortized.
- Logm פעולות decreaseKey וכל אחת לוקחת O(1) amortized.

סך כל:

(m+1)*O(1)+O(logm)+logm*O(1)=O(m)

ב)

m	Run-Time(ms)	totalLinks	totalCuts	Potential
2^{10}	3	1023	10	29
2^{15}	27	32767	15	44
2^{20}	245	1048575	20	59
2^{25}	4947	33554431	25	74

כריסטיאן שקור
ת.ז 322828518

ג-ו)

m	totalLinks	totalCuts	Potential	decreaseKey max cost
(c)original	m-1	logm	(3logm)-1	(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	2logm-1	2logm	Logm-1

נימוק הטבלה:

- **totalLinks:**
  אחרי ההכנסות נקבל m+1 עצים מדרגה 0 (צמתים בודדים).
  החיבורים יתבצעו רק אחרי פעולת deleteMin.
  c+d+f) כיוון שהוספנו m+1 ואחר כך מחקנו המינימלי אז נשאר m צמתים, וכיוון m הוא חזקה שלמה של 2 אזי אחרי ה-successive Linking נקבל עץ יחיד, זה אומר שחיברנו כל m הצמתים לעץ בינומי חוקי, אז ביצענו m-1(מספר הקשתות) חיבורים (אפשר להוכיח באינדוקציה בקלות).
  e) לא בצענו deleteMin אז לא מבצעים successive Linking וזה אומר שאין חיבורים 0.

- **totalCuts:**
  נשים לב שאחרי בצוע deleteMin (כל המקרים למעט e) נקבל עץ אחד בינומי מדרגה logm ששורשו הוא 0 כי סדר ההכנסות שלנו היה יורד.
  עץ זה הוא חיבור שני עצים A, B מדרגה logm-1 , ב A יש את האיברים מ- 0 עד- (m/2)-1 וב- B יש את האיברים מ- m/2 עד- m-1 (זה נובע מסדר הכנסת האיברים).
  שורש A הוא 0, ושורש B הוא m/2.
  לכן בעץ כולו ל-0 יש את הבנים 0 ו- m/2.
  ובאינדוקציה אפשר לקבל שלצומת m(1-(1/2^i)) יש את הילדים:
  m(1-(1/2^i))+1 ו- m(1-(1/2^(i+1))).

כריסטיאן שקור
ת.ז 322828518

c) במקרה זה בכל אטירציה של decreaseKey אנחנו חותכים קשת אחת לכן סך כל חתכנו logm קשתות.
d) נשים לב שבאטרציה ראשונה מקטינים את השורש ואחר כך הבן שלו וכך הלא, הקטנו באותו מספר לכן ההפרש ביניהם לא השתנה, וכיוון שהתחלנו בשורש וירדנו למטה אז העץ נשאר חוקי כל הזמן ולכן לא מבצעים חתוכים 0.
e) במקרה זה לא בצענו deleteMin לכן לא בצענו חיבורים, אז אין קשתות למחק, ולזה מספר החיתוכים הוא 0.
f) ה-decreaseKey האחרון יצר קריאה ל- cascading-cuts. ביצענו שרשרת של cuts בגודל logm עד השורש. וזו היא בעולת ה-decreaseKey היקרה ביותר שלוקחת logm.
לכן בצענו logm-1+logm=2logm-1 חיתוכים.

- **Potential:**

c) התחלנו עם עץ אחד, ובכל חיתוך הוספנו עץ וסימנו צומת למעט השורש, ועשינו logm חיתוכים לכן הפוטנציאל הוא :

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

d) אין חיתוכים אזי לא משתנה הפוטנציאל, נשאר 1 (עץ יחיד).
e) יש לנו m+1 עצים ושורשים, לכן אין צמתים שמסומנים, לכן הפוטנציאל הוא שווה למספר העצים והוא m+1.
f) בשרשרת ה-cascading-cuts הופכת כל הצמתים להיות לא מסומנים, לכן הפוטנציאל הוא שווה למספר העצים והוא

1+2logm-1- =2logm

שאלה 2:

א)

m	Run-Time(ms)	totalLinks	totalCuts	Potential
728	4	723	0	6
6560	13	6555	0	6
59048	96	59040	0	9
531440	426	531431	0	10
4782968	3877	4782955	0	14

ב) כיוון שמדובר בסדרת פעולות אזי נחשב לפי זמן amortized :

- m+1 פעולות insert וכל אחת לוקחת O(1).
- 3m/4 פעולות deleteMin וכל אחת לוקחת O(logm) amortized.

סך כל:

$$(m+1)O(1)+\frac{3m}{4}O(logm)=O(mlogm)$$

ג)

- **totalCuts:**
  נשים לב שלא בצענו שום פעולת decreaseKey ורק שם אפשר להתבצע חיתוכים, לכן מספר החיתוכים הוא 0.

- **totalLinks:**
  מספר החיבורים חסום ע"י O(m) וזה כי סדר ההכנסות מונוטוני ומספר המחיקות.

- **Potential:**
  לא בצענו decreaseKey אז אין חיתוכים ואין צמתים מסומנים.
  לכן : Potential=#trees.

הוספנו m+1 איברים ומחקנו 3m/4 לכן נשאר (m/4)+1איברים, ובכל מצב של העץ אחרי ה-deleteMin הראשון עד סוף סדרת הבעולות יש לנו עץ בינומי חוקי, לכן בסוף סדרת הפעולות יש לנו עץ בינומי חוקי עם (m/4)+1 צמתים, זה אומר שמספר העצים בערמה הוא מספר הביטים הדלוקים 1 בייצוג הבינארי של המספר (m/4)+1.
לכן :

Potential=# ה-1-ים בייצוג הבינארי של המספר (m/4)+1

כריסטיאן שקור
ת.ז 322828518