תום סגל ת.ז 208945519
גל ורניק ת.ז 208884213
מרצה: פרופ' רני הוד
להגשה 6.6.2016

מבני נתונים- פרוייקט מעשי מס' 2

תיעוד המחלקה BinomialHeap

שדות

שם	טיפוס	נראות	תיאור
head	HeapNode	private	מצביע ל-HeapNode אשר מכיל את ראש הערימה, כלומר את השורש של העץ הבינומי הקטן ביותר בערימה.
min	HeapNode	private	מצביע ל-HeapNode אשר מכיל את הערך הקטן ביותר בכל הערימה.
size	int	private	ערך מספר שלם אשר מייצג את מספר האיברים בערימה.
map	HashMap	private	מפת Hash אשר מקבלת טיפוס מסוג int ומחזירה ב-O(1) את המצביע ל-HeapNode בערימה אליו שייך הערך המספרי.

מחלקות פנימיות

המחלקה HeapNode

נראות- public.

שדות

שם	טיפוס	נראות	תיאור
key	int	private	מספר שלם שמייצג את הערך של ה-Node הספציפי
next	HeapNode	private	מצביע לאובייקט HeapNode שמהווה הבן הבא בעץ/השורש הבא ברשימת השורשים. מופיע לפעמים בספרות כ-sibling.
parent	HeapNode	private	מצביע לאובייקט HeapNode שמהווה האב של אותו ה-Node.
child	HeapNode	private	מצביע לאובייקט HeapNode שמהווה הבן בעל הדרגה הגדולה ביותר של אותו ה-Node.
degree	int	private	מספר שלם המייצג את הדרגה של אותו ה-Node בערימה, כלומר מספר הבנים שלו.

מתודות

HeapNode

קלט: int key

פלט: Constructor

נראות: public

סיבוכיות: O(1)

המתודה היא מתודת בנאי רגילה לאובייקט מטיפוס HeapNode. המתודה מקבלת מספר שלם ובונה HeapNode שכל הערכים שלו מאותחלים ל-null, מלבד הערך המספרי שמאותחל ל-key.

toString

קלט: אין

פלט: String

נראות: public

סיבוכיות: O(1)

המתודה מחזירה ייצוג של ה-Node בתור מחרוזת. שימושית עבור ייצוג של הערימה.

מתודות

empty

קלט: אין

פלט: boolean

נראות: public

סיבוכיות: O(1)

המתודה מחזירה אמת אם ורק אם הערימה הבינומית ריקה.

insert

קלט: int value

פלט: אין

נראות: public

סיבוכיות: O(logn), Amortized O(1)

המתודה מקבלת ערך מספרי value. היא מכניסה אותו לתוך הערימה על ידי יצירת ערימה חדשה ש-value הוא הערך היחיד בה, וביצוע meld איתה ועם הערימה המקורית.

deleteMin

קלט: אין

פלט: אין

נראות: public

סיבוכיות: O(logn)

המתודה מוחקת את הערך המינימלי בערימה. היא הופכת את כל הבנים של ה-Node לו היה שייך הערך המינימלי לערימה חדשה. על מנת לשמור על תקינות הערימה המקורית, היא מאחדת את ערימה זו עם הערימה המקורית.

findMin

קלט: אין

פלט: int k

נראות: public

סיבוכיות: O(1)

המתודה מחזירה את הערך המינימלי בערימה כולה. היא עושה זאת על ידי קריאה ל-Node שמכיל אותו- ששמור כשדה של הערימה.

BinomialHeapMerge

קלט: BinomialHeap h

פלט: אין

נראות: private

סיבוכיות: O(logn)

מתודת העזר הפנימית מקבלת ערימה בינומית h. היא משרשרת את רשימת השורשים של הערימה עצמה ושל h תוך שמירה על סדר עולה של דרגות, ומחזירה את ה-Node של תחילת הרשימה.

meld

קלט: BinomialHeap h

פלט: אין

נראות: public

סיבוכיות: O(logn)

המתודה מקבלת ערימה בינומית h. היא משרשרת את רשימת השורשים של הערימה עצמה ע"י שימוש ב-BinomialHeapMerge, ואז מתקנת את הרשימה ע"י איחוד עצים בינומיים בעלי אותה דרגה. כמו כן היא מתקנת בהתאם את שדות ה-size, map, min של הערימה החדשה.

size

קלט: אין

פלט: int n

נראות: public

סיבוכיות: O(1)

המתודה מחזירה את מספר האיברים בערימה, השמור כשדה של הערימה.

arrayToHeap

קלט: int [] array

פלט: אין

נראות: public

סיבוכיות: O(n)

המתודה מקבלת מערך שלם מספרים שלמים array. היא בונה ערימה בינומית חדשה, ומכניסה את כל איברי המערך לערימה בסיבוכיות O(n) כפי שנלמד בכיתה. לבסוף, היא מעדכנת את כל השדות של הערימה המקורית להיות השדות של הערימה החדשה (ובכך הופכת את עצמה להיות היא, תוך שהיא מאבדת את כל הערכים הקודמים), ומסיימת.

minTreeRank

קלט: אין

פלט: int k

נראות: public

סיבוכיות: O(1)

המתודה מחזירה את הדרגה של העץ הבינומי הקטן ביותר בערימה. היא עושה זאת ע"י קריאה לשדה degree של ה-Node שמהווה את ראש הערימה (head).

binaryRep

קלט: אין

פלט: boolean [] arr

נראות: public

סיבוכיות: O(logn)

המתודה מחזירה את הייצוג הבינארי של הערימה. כלומר, היא מחזירה מערך בוליאני, המוגדר מאינדקס 0 עד הדרגה המקסימלית שקיימת בעץ, ועבור כל עץ שקיים בערימה, הערך באינדקס של הדרגה יהיה True.

isValid

קלט: אין

פלט: boolean

נראות: public

סיבוכיות: O(n)

המתודה מחזירה True אם ורק אם הערימה היא ערימה בינומית תקינה. כלומר- כלל הערימה מתקיים, וכל עץ בערימה הוא עץ בינומי תקין. המתודה אינה מסתמכת על אף מתודה אחרת במחלקה, מלבד מתודות העזר שהוגדרו עבורה. היא עוברת על כל עץ בערימה, ומחשבת את מספר הילדים שלו באמצעות המתודה הפנימית numOfChildren. לאחר מכן, היא בודקת אם העץ תקין (כלומר עץ מדרגה numOfChildren) באמצעות

המתודה isValidTree. היא מוודאת שאין שני עצים מאותה דרגה. לבסוף, היא מוודאת שאכן המינימום בעץ הוא המינימום השמור, ושאכן מספר האיברים בעץ הוא ה-size השמור.

numOfChildren

קלט: HeapNode node

פלט: int k

נראות: private

סיבוכיות: O(logn)

המתודה מקבלת שורש של עץ בינומי node. היא עוברת על כל הילדים של node ומחשבת כמה מהם יש- זו הדרגה של העץ ששורשו node. היא מחזירה את מספר זה.

isValidTree

קלט: HeapNode node, int degree

פלט: boolean

נראות: private

סיבוכיות: O(n)

המתודה מקבלת שורש של עץ בינומי node ומספר שלם degree. המתודה בודקת אם node הוא שורש של עץ בינומי תקין מדרגה degree. היא עושה זאת כך: היא עוברת על כל הילדים של node- לכל אחד מהם, היא מפעילה את הקריאה הרקורסיבית על עצמה בשביל לוודא שהם עצים מהגודל המתאים. כמו כן, היא מוודאת שהם מקיימים את כללי העץ הבינומי. לבסוף היא מוודאת שאכן עברה על degree בנים, ומחזירה אמת אם ורק אם העץ הבינומי תקין.

delete

קלט: int i

פלט: אין

נראות: public

סיבוכיות: O(logn)

המתודה מקבלת מקבלת מספר שלם אותו עליה למחוק מהערימה. אם לא קיים איבר כזה בערימה, המתודה לא תעשה כלום. אם הוא קיים, היא תקרא ל-decreaseKey(i,(-infinity)), כלומר, היא תוריד את הערך שלו לנמוך ככל הניתן, ואז תקרא ל-deleteMin על מנת למחוק אותו.

decreaseKey

קלט: int i, int j

פלט: אין

נראות: public

סיבוכיות: O(logn)

המתודה מקבלת שני מספרים שלמים i,j. היא משתמשת בטבלת הגיבוב של הערימה על מנת למצוא את ה-node שערכו i. אם לא נמצא כזה, המתודה תעצור. אם מצאה, היא תשנה את ערכו ל-j תוך תיקון הערימה כפי שנלמד בכיתה. כמו כן היא דואגת לעדכן את הטבלה, ואת השדה min במידת הצורך.

binominalLink

קלט: HeapNode y, HeapNode z

פלט: אין

נראות: private

סיבוכיות: O(1)

המתודה מקבלת שני שורשים של עצים בינומים y ו-z. היא מקשרת אותם יחדיו כך ש-y הוא הבן הגדול ביותר של z. היא מקפידה לעדכן את כל השדות של שני ה-node-ים בהתאם, ולהעלות את ה-degree של z באחד.

מדידות

חלק א

.1
נבחר את המספר k להיות המספר היחיד המקיים: m =^ 2^k > (m/2). מספר זה הוא בעצם floor(log(m)). נכניס לערימה (2^k - 1) איברים. המטרה בכך היא לבנות את הערימה הגדולה ביותר שניתן, כך שיהיה לה עץ מכל דרגה (בדומה למונה בינארי שבו כל הערכים הם 1).

נקבל שגודל הערימה הוא O(2^k) = O(m) אמורטייזד. לאחר מכן נוציא ונכניס את האיבר האחרון שהכנסנו שוב ושוב, עד שנבצע בסה"כ m-1 פעולות (בשלב זה לא יתבצעו עוד לינקים).

לבסוף, נכניס איבר נוסף לערימה, כך שהוא יגדיל אותה לגודל 2^k ויגרום ל- k = Θ(logm) לינקים.

.2
זמן הריצה הכולל יהיה: O(m) + Θ(logm) = O(m).

.3
הייצוג הבינארי של הערימה לפני הפעולה האחרונה יהיה רצף של k אחדות ואילו לאחר הפעולה האחרונה הספרה הראשונה תהיה 1 ואחריה k אפסים (אנחנו מגדילים את היצוג ב-1).

חלק ב

.1

לאחר ביצוע סדרת הפעולות, קיבלנו את התוצאות הבאות:

m	linking after m-1	linking after m
1000	502	511
2000	1013	1023
3000	2036	2047

נתבונן בטבלה. ניתן לראות שבכל ניסוי, מספר הלינקים ב- (m-1) הפעולות הראשונות הוא:

$$2^k - 1 - k = O(2^k) = O(m)$$

כפי שציפינו, ואילו בפעולה האחרונה מספר הלינקים הוא k = Θ(logm).

.2

תוצאות binaryRep לאחר (m-1) ו-m פעולות:

binary rep after m	binary rep after m-1	m
[F, F, F, F, F, F, F, F, F, T]	[T, T, T, T, T, T, T, T, T, T]	**1000**
[F, F, F, F, F, F, F, F, F, F, T]	[T, T, T, T, T, T, T, T, T, T, T]	**2000**
[F, F, F, F, F, F, F, F, F, F, F, T]	[T, T, T, T, T, T, T, T, T, T, T, T]	**3000**

התוצאות כמו שציפינו- הערימה תהיה "מלאה" לאחר (m-1) הכנסות, ובהכנסה ה-m בדיוק כל העצים בה יתאחדו לעץ יחיד.