**פרויקט 2**

**Biesan assiביסאן עאסי 208145268**

**yazanzoabiיזן זועבי 209105790**

**מבני נתונים**

**תיעוד חיצוני:**

**HeapNode**

מחלקה שמגדרה הצומת בערימת פיבונצ"י.

**public** **int** key;

המפתח של הצומת

**private** **int** rank;

הדרגה של הצומת.

**private** **int** mark;

אם mark=1 נקבל שנחתך בן מעץ הזה אחרת העץ לא נחתך לא אף בן "חוץ אם היה שורש".

**private** HeapNode child;

הבן של הצומת.

**private** HeapNode next;

האח הימני של הצומת.

**private** HeapNode prev;

האח השמאלי לצומת.

**private** HeapNode Parent;

האב של הצומת.

הפקודות של הצומת: כלן בסיבכיות של O(1).

**public** HeapNode(**int** key)

מחלקת הבנאי לצומת מגדירה הערך של המפתח עבור הצומת והמאתחלת המצביעים next ו prev להצביע על הצומת עצמה.

**public** **int** getKey()

מחזירה המפתח של הצומת.

**public** **boolean** equals(HeapNode t)

בודקת אם אנחנו באתו צומת ע"י בדיקת עירך המפתח כך אם יש להן אותו מפתח אז אנחנו באותו צומת "מבחנת הנתון שכל הצמתים מכילים מספרים שונים".

**FibonacciHeap**

המחלקה המייצגת ערימה פיבונצי כפי שלמדנו בקורס:

**private** HeapNode Min

מצביע על העץ בעל שורש של האיבר המינימאלי בערימה.

**private** **int** size

מס הצמתים בערימה.

**private** **int** numOfTree

מספר העצים בערמה.

**private** **int** numOfMarked

מספר הצמתים שנחתך להם בן אחד כפי שלמדנו.

**private** **static** **int** *TotalCut*

מספר הפעמים שעשינו חיתוך מהתחלת בניית הערמה.

**private** **static** **int** *TotalLinks*

מספר החיבור בין שתי צמתים שעשינו מהתחלת בניית הערימה.

פקודות של הערימה:

**public** **boolean** isEmpty()

הפונקציה בודקת אם הערימה ריקה או לא כלומר בודקת אם Min==null.

סיבכיות הזמןW.C:O(1).

סיבכיות הזמן amort:O(1).

**public** HeapNode insert(**int** key)

הפונקציה יוצרת צומת מסוג HeapNode שמכיל את המפתח ומכניסה אותו לערימה. ומחזירה את הצומת שנוצר ושמכיל את המפתח .

סיבכיות הזמןW.C:O(1).

סיבכיות הזמן amort:O(1).

**public** **void** deleteMin()

מחיקת הצומת שהמפתח שלו מינימלי מבין המפתחות שבערימה כלומר חותכת את האיבר המינימאלי ואז עושהmeld של הילדים עם הערימה המקורית ומעדכנת את השדה min ואז נקרא למתודה Consolidating() וגם למתודה updateMin().

סיבכיות הזמןW.C:O(n).

סיבכיות הזמן amort:O(log(n)).

**private** **void** updateMin()

הפעולה זו מעדכנת האיבר המינימלי בעץ "שומרת על מינימאליות" משתמשים בה לפני סיום במתודה של ()deleteMin.

סיבכיות הזמןW.C:O(log(n)).

סיבכיות הזמן amort:O(log(n))

**private** **void** Consolidating()

המתודה עושה פעולת consolidate כפי שלמדנו בכיתה המתודה מגדירה מערך בגודל לוגרתמי במספר האברים ואז עוברת על כל עץ בערימה ומכניסה אותו לתא שבגודל הדרגה שלו אם יש עץ בתא זו המתודה עושהlink לשני העצים ומכניסה את התוצאה לתא הגדול ממנו באחד.

סיבכיות הזמןW.C:O(n).

סיבכיות הזמן amort:O(log(n))

**private** **void** link(HeapNode x, HeapNode y) {

ממזגת את העץx לעץy כפי שלמדנו כלומר בודקת את השורש המנימלי בין שני השורשים וממחברת את השורש השני.

סיבכיות הזמןW.C:O(1).

סיבכיות הזמן amort:O(1).

**public** HeapNode findMin()

מחזיר את הצומת (מטיפוס HeapNode) שהמפתח שלו מינימלי מבין המפתחות שבערימה כלומר מחזר המפתח של Min שהגרנו בהתחלה.

סיבכיות הזמןW.C:O(1).

סיבכיות הזמן amort:O(1).

**public** **void** meld (FibonacciHeap heap2)

מיזוג הערימה עם ערימה נוספת heap2.

סיבכיות הזמןW.C:O(1).

סיבכיות הזמן amort:O(1).

**public** **int** size()

הפונקציה מחזירה את מספר האיברים בערימה כלומר מחזירה הערך של Size שהגדרנו בהתחלה ובכל פעולת הוספה גידלנו אותו ובכל פעולת מחיקנה הקטנו ב1.

סיבכיות הזמןW.C:O(1).

סיבכיות הזמן amort:O(1).

**public** **int**[] countersRep()

הפונקציה מחזירה מערך מונים כך שבאינדקס שמור כמה עצים יש בערימה שהסדר שלהם הוא . כלומר, היא מחזירה מערך של integers, כך שלכל אינדקס בין עד הדרגה המקסימלית של עץ שקיימת בערימה, הערך שמוחזר במערך הוא מספר העצים שקיימים בערימה מסדר .

סיבכיות הזמןW.C:O(n).

סיבכיות הזמן amort:O(n).

**public** **void** delete(HeapNode x)

מחיקת הצומת x מהערימה כלומר המתודה עושהdecresekey לx כך שהערך

שנקטין הערך שלו להיות פחות מהערך המינימלי בערימה ואז עושים deletemin().

סיבכיות הזמןW.C:O(n).

סיבכיות הזמן amort:O(log(n)).

**public** **void** decreaseKey(HeapNode x, **int** delta)

ערכו של המפתח של הצומת x יופחת ב . כלומר, מתבצע , ומעדכן את מבנה הערימה בהתאם לשינוי זה.

סיבכיות הזמןW.C:O(1).

סיבכיות הזמן amort:O(1).

**private** **void** cascading\_cut(HeapNode x, HeapNode y)

עושה cascadingCut כפי שלמדנו בכיתה המתודה חותכת את x מy ומסמנת את y ההורה של x אם ההורה

שלו גבר מסומן היא חותכת אותו באופן רקורסיבי.

סיבכיות הזמןW.C:O(log(n)).

סיבכיות הזמן amort:O(1).

**private** **void** cut(HeapNode x, HeapNode y)

חותכת את הx מהעץ ומוסיפה אותו כעץ חדש בערימה כלומר עושים meld.

סיבכיות הזמןW.C:O(1).

סיבכיות הזמן amort:O(1)

**public** **int** potential()

הפונקציה מחזירה את ערך הפוטנציאל הנוכחי של הערימה.

הפונטנציאל, כפי שהוגדר בשיעור, הינו

Potential = #trees + 2\*#marked.

סיבכיות הזמןW.C:O(1).

סיבכיות הזמן amort:O(1)

**public** **static** **int** totalLinks()

פונקציה סטטית זו מחזירה את מספר כל פעולות הלינק שבוצעו מתחילת ריצת התוכנית. פעולת לינק הינה הפעולה שמקבלת שני עצים מאותו סדר ומחברת אותם.

סיבכיות הזמןW.C:O(1).

סיבכיות הזמן amort:O(1)

**public** **static** **int** totalCuts()

פונקציה סטטית זו מחזירה את מספר כל פעולות הcut שבוצעו מתחילת ריצת התוכנית. פעולת cut קורת בזמן decreaseKey, כאשר מנתקים תת-עץ מהאבא שלו (כולל cascading cuts).

סיבכיות הזמןW.C:O(1).

סיבכיות הזמן amort:O(1)

**public** **static** **int**[] kMin(FibonacciHeap H, **int** k)