**מבני נתונים - פרויקט מספר 2 - ערימת פיבונאצ'י**

**מגישים**:

שם: איתי צמח

שם משתמש: itaizemah

ת.ז: 209637453

שם: עודד כרמון

שם משתמש: odedcarmon

ת.ז: 208116517

**תיעוד המחלקה**:

המחלקה FibonacciHeap מממשת ערימת פיבונאצ'י המכילה מפתחות שלמים.

צמתי הערימה ממומשים על ידי המחלקה הפנימית HeapNode.

המחלקה FibonacciHeap

למחלקה 10 שדות:

* מספר שלם size המתאר את גודל הערימה, מאותחל כ-0.
* מצביעים לצמתים min,first,last המצביעים לצומת בעל המפתח הקטן ביותר, ולשורשי העצים השמאלי/ימני ביותר בהתאמה.
* מספרים שלמים numTrees, numMarked המתארים את כמות העצים בערימה וכמות הצמתים המסומנים בערימה, בהתאמה. משומשים לחישוב פוטנציאל העץ.
* מספרים שלמים סטטיים totalLinks, totalCuts המתארים את סה"כ כמות הקישורים והחיתוכים שהתבצעו על ידי כל המופעים של המחלקה עד כה, בהתאמה.
* קבועים סטטיים שלמים NEG\_INFTY, POS\_INFTY המתארים אינסוף חיובי ושלילי, לשימוש במחיקת איבר שרירותי (ראשית מקטינים את המפתח שלו למינוס אינסוף) וכאשר מחפשים את המינימום במהלך קונסולודיציה (משווים את המפתחות למועמד נוכחי למינימום שמאותחל לאינסוף).

מתודות המחלקה:

* int maxPossibleRank(): מחזיר חסם עליון הדוק אסימפטוטית לדרגה המקסימלית של עץ בערימה, בסיבוכיות , לשימוש ביצירת מערך הדליים בקונסולידציה ולחישוב countersRep().
* boolean isEmpty(): בודק בסיבוכיות האם הערימה ריקה על ידי השוואת הגודל שלה ל-0.
* void addRoot(HeapNode newNode): מוסיפה צומת נתון כשורש בתחילת הערימה, מעדכנת את כל המצביעים הדרושים, מגדילה את שדה numTrees ומעדכנת את המינימום אם השורש החדש שהוספנו הוא מינימום חדש. פועלת בסיבוכיות .
* void removeNode(HeapNode node): אם קוראים לפונקציה זו עם קלט שאינו הצומת האחרון בערימה – הצומת מוסר מרשימת האחים שלו, בין אם הוא שורש או צומת פנימי. פועלת בסיבוכיות .
* HeapNode insert(int key): יוצרת צומת חדש עם המפתח הנתון ומכניסה אותו באופן עצל לתחילת הערימה באמצעות addRoot, מחזירה מצביעה לצומת שהכנסנו כדי שיהיה אפשר לגשת אליו בהמשך. פועלת בסיבוכיות .
* HeapNode bruteFindMin(): מחפשת באופן נאיבי את המינימום על ידי מעבר סדרתי על כל השורשים בערימה והשוואת המפתחות שלהם מול מועמד נוכחי למפתח מינימלי. משומשת לאחר קונסולידציה למציאת המינימום החדש. פועלת בסיבוכיות .
* void deleteMin(): אם הערימה לא ריקה - מסירה את הצומת המינימלי מהערימה, אם היו לו ילדים מעלה אותם למעלה במקומו לרשימת השורשים. אם נשארו לאחר המחיקה צמתים בערימה מתבצעת קונסולידציה. פועלת בסיבוכיות worst case .
* void consolidate(): מבצע קונסולידציה כפי שראינו בכיתה – מאתחל מערך של "דליים" אליו העצים בערימה מוכנסים לפי הדרגה שלהם (כל עץ מוסר ראשית מהערימה עצמה לפני שמועבר למערך), במקרה של התנגשות שני העצים מקושרים באמצעות link ומועברים לדלי המתאים עד שאין יותר התנגשות. לאחר שכל העצים הועברו לדליים, בונים מחדש את הערימה לפי המערך, מהעץ בעל הדרגה הנמוכה ביותר אל העץ בעל הדרגה הגבוהה ביותר. התהליך עד כה עלה , כעת ישנם שורשים עליהם מבצעים bruteFindMin() למציאת המינימום החדש. סה"כ הסיבוכיות היא .
* void link(HeapNode node1, HeapNode node2): אם מקבל שורשים של של שני עצים בעלי אותה דרגה, תולה את העץ בעל המפתח הגדול יותר כבן חדש (שמאלי ביותר) של העץ בעל המפתח הקטן יותר. פועל בסיבוכיות .
* HeapNode findMin(): מחזיר את השדה min של הערימה בסיבוכיות .
* void meld(FibonacciHeap heap2): ממזג באופן עצל את הערימה הנוספת אל הערימה הנוכחית על ידי שרשור עצי הערימה הנוספת בסוף הערימה הנוכחית. פועל בסיבוכיות .
* int size(): מחזיר את השדה size של הערימה בסיבוכיות .
* int[] countersRep(): עובר סדרתית על שורשי הערימה ומעדכן מערך בגודל maxPossibleRank כך שיכיל לבסוף בתא ה- את מספר העצים מדרגה בערימה. פועל בסיבוכיות .
* void delete(HeapNode x): מוחק צומת מהערימה על ידי הקטנת המפתח שלו למינוס אינסוף ואז מחיקת המינימום. קורא לפונקציות decreaseKey ו-deleteMin שפועלות ב- ולכן גם פונקציה זו בעלת סיבוכיות .
* void decreaseKey(HeapNode x, int delta): מקטין את המפתח של הצומת הנתון ב-delta ומעדכן את המינימום במקרה הצורך. אם כעת המפתח של הצומת קטן מהמפתח של ההורה שלו, הצומת נחתך באמצעות קריאה ל-cascadingCuts. הפונקציה פועלת בסיבוכיות .
* void cascadingCuts(HeapNode x, HeapNode y): מקבלת צומת x בעלת הורה y וחותכת את x מההורה שלו באמצעות cut. אם ההורה y לא היה מסומן, מסמנים אותו כעת, ואם היה מסומן, מבצעים גם עליו cascadingCuts.
* void cut(HeapNode x, HeapNode y): מקבלת צומת x בעלת הורה y, מסירה את x מהילדים של y באמצעות removeNode ומוסיפה אותו כשורש חדש בתחילת הערימה באמצעות addRoot. פועלת בסיבוכיות .