מחלקת BinomialHeap

|  |  |  |
| --- | --- | --- |
| שדה | ערך דיפולטי | תפקיד |
| Size | 0 | כמות הצמתים בערימה |
| Last | null | מצביע לעץ הבינומי בעל הדרגה המקסימלית בערימה |
| min | Null | מצביע לעץ הבינומי בעל השורש עם המפתח הקטן ביותר בערימה |
| numTrees | 0 | מספר העצים הבינומים בערימה |

פונקציות נדרשות –

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| שם הפונקציה | תפקיד | ערך החזרה | תיאור הפעולה | סיבוכיות |
| Insert(key, info) | הכנסת צומת חדש בעל מפתח key וערך info לערימה | HeapItem | 1. יצירת HeapNode וHeapItem   עבור המפתח והערך (שאר השדות דיפולטיים)   1. יצירת ערימה חדשה שmin וlast שלה הם הheapNode החדש 2. Meld של הערימה החדשה לתוך הערמה הthis | סעיפים 1 ו2 בסיבוכיות  כי הבנאים של המחלקות דיפולטיים.  סעיף 3 בסיבוכיות  כפי שנוכיח בתיעוד הפונ' meld. |
| deleteMin | מחיקת הצומת בעל הערך המינימלי בערימה | אין | הצומת המינמילי הוא שורש של עץ בינומי (מכלל הערימה) ושמור במצביע min. כעת:   1. יצירת heap1 שתכיל את כל העצים הבינומים בערימה מלבד העץ ששורשו הוא המינימום (מתבצע בעזרת לולאה על כל העצים, שרשור העצים המתאימים ועדכון של min/last). 2. יצירת heap2 שתכיל את כל הילדים הישירים של השורש המינימלי (מתבצע בעזרת לולאה מmin.child על כל הילדים בעזרת next, שרשור הילדים לרשימה ועדכון min/last) 3. Heap1.meld(heap2) 4. עדכון הערימה (this) להיות heap1 | סעיף 1 דורש  *עבודה כי מספר העצים בערימה חסום ע"י log(n) כפי שראינו בהרצאה*.  סעיף 2 גם דורש  *עבודה כי המקרה הכי גרוע הוא שהעץ בעל השורש המינימלי הוא העץ בעל הדרגה הגבוה ביותר, במקרה זה כמות הבנים שלו חסומה על ידי הדרגה שלו שהיא* log(n) *כפי שראינו בהרצאה.*  סעיף 3 בסיבוכיות  כפי שנוכיח בתיעוד הפונ' meld.  סעיף 4 דורש O(1) עבודה כפי שמנומק בתיעוד פונ' עזר. |
| findMin() | החזרת מצביע לצומת המינימלי בערימה | HeapItem | החזרת המצביע ששמור בשדה min |  |
| decreaseKey  (HeapItem item, int diff) | הקטנת המפתח של item בdiff ושמירה על תקינות הערימה | void | 1. הקטנת המפתח של item בdiff 2. אם לitem אין parent – העץ של item עדיין תקין (לפי כלל הערימה הוא עדיין קטן מילדיו). בדיקה האם הוא צריך להיות המינמום החדש והחלפה במידת הצורך 3. אחרת, כל עוד לitem יש parent וגם המפתח של parent גדול יותר מהמפתח של item:    1. החלף את המפתח והערך של item וparent    2. קדם את item להיות הparent (כך שיחזור להצביע על הצומת עם המפתח והערך המקוריים שלו שעלה בעץ) | סעיפים 1 ו2 דורשים זמן קבוע.  סעיף 3 - במקרה הכי גרוע שבו כל הצמתים באותו העץ וitem הוא עלה שדורש תיקון עד לשורש, כמות העבודה תיהיה חסומה על ידי גובה העץ.  גובה העץ חסום על ידי דרגתו שחסומה *ע"י* log(n) *כפי שראינו בהרצאה.* |
| Delete  (HeapItem item) | מחיקת הצומת item ושמירה על ערימה תקינה | אין | 1. הקטן את המפתח של item במפתח הנוכחי שלו (כך שיקבל את הערך 0) בעזרת הפונ' decreaseKey 2. קרא לdeleteMin() שימחק את item כי הוא קיבל בסעיף 1 את המפתח 0 שהוא בהכרח הכי נמוך בערימה (כי הוגדר שהערמה לא תומכת בהכנסה של מפתח 0). | סעיפים 1 ו2 שניהם דורשים  *עבודה כפי שהוכחנו עבור שתי הפונקציות.* |
| meld(BinomialHeap heap2) | איחוד heap2 לתוך הערימה this | אין | 1. בדיקת תוכן –    1. אם שתי הערימות ריקות – אפשר לסיים    2. אם heap2 ריקה – אפשר לסיים    3. אם this ריקה – החלף את this להיות heap2 ואפשר לסיים 2. אתחל שני מערכים באורך (הדרגה המקסימלית של שתי הערמות + 2) 3. עבור מערך 1 – הכנס לכל תא i מצביע לשורש העץ בדרגה i אם קיים כזה בערימה this – אחרת הכנס צומת "וירטואלי" 4. אותו דבר כמו סעיף 3 עבור heap2 5. אתחל מערך "תוצאה" באותו האורך 6. אתחל מערך סכימה באורך 3 7. אתחל carry להיות צומת וירטואלי. 8. אתחל toInsert להיות צומת וירטואלי. 9. עבור בלולאה מ0 עד (הדרגה המקסימלית של שתי הערמות + 2)    1. אם יש צומת אמיתי במערך 1 במקום i – הכנס אותו למערך סכימה (בעזרת פונ' עזר ins2arr3)    2. אותו דבר עבור מערך 2    3. הכנס שארית    4. אם במערך הסכימה יש 3 צמתים אמיתיים - הכנס לtoInsert את הצומת האחרון במערך הסכימה    5. אם במערך הסכימה יש 2 צמתים אמיתים (יכול לקרות גם בנוסף לסעיף d) –   הכנס לcarry את האיחוד של שניהם (בעזרת פונ' העזר link)   * 1. toInsert הוא הצומת הראשון במערך   2. הכנס את toInsert למערך התוצאות בתא הi  1. עבור על מערך התוצאות וחבר את השדה next בין צמתים בדרגה עולה (שאינם וירטואלים), מצא את המינימום, חשב את size, numTrees וסמן את last בתור העץ האחרון במערך התוצאות. 2. עדכן את this לקבל את min, last, size, numTrees של הערימה החדשה | סעיף 1 דורש עבדוה קבוע  סעיפים 2,3,4,5 חסומים ע"י הדרגה המקסימלית + 2, שזה חסום ע"י log(n)  סעיפים 6,7,8 מבצעים כמות עבודה קבועה  בסעיף 9 מספר האיטרציות של הלולאה חסום ע"י log(n) ובכל ריצה מתבצעת כמות עבודה קבועה (כולל link שמנומק בטבלת פונ' עזר)  בסעיף 10 מספר האיטרציות של הלולאה גם חסום ע"י log(n) ובכל איטרציה יש כמות עבודה קבועה (תיקון מצביעים והעלאת counterים).  סעיף 11 מעדכן 4 מצביעים ולכן גם בסיבוכיות קבועה |
| size | מחזירה את מספר הצמתים בערימה | int | החזרת השדה this.size |  |
| empty | מחזירה האם הערימה ריקה | boolean | החזרת הערך הבוליאני של הביטוי size==0 |  |
| numTrees | מחזירה את מספר העצים הבינומים בערימה | int | החזרת השדה numTrees |  |

פונקציות עזר -

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| שם הפונקציה | תפקיד | ערך החזרה | תיאור הלוגיקה | סיבוכיות |
| switchHeaps(BinomialHeap heap2) | מעדכנת את הערימה this להיות heap2 | אין | החלפת ארבעת השדות של המחלקה |  |
| ins2arr3(HeapNode[] arr, HeapNode n) | מקבלת מערך (בסיסי של ג'אווה) באורך 3 וheapNode ומכניסה אותו לתא הראשון הפנוי (הראשון שנמצא ללא צומת) | אין | ריצה בלולאה על המערך ובדיקה האם התא הנוכחי פנוי להכנסה) | באופן כללי הפונקציה לינארית באורך המערך, אבל זו פונקצית עזר פרטית שנקראת בפרוייקט שלנו רק עבור מערך באורך 3 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |