חלק תיאורטי-שמות המגישים:

תומר רודניצקי (20662787820)- rudnitzky

עידו רוזינר (209617000)- idorosiner

**להלן תיעוד FibonacciHeap:**

|  |  |  |
| --- | --- | --- |
| **חישוב סיבוכיות זמן** | **תיאור** | **פונקציה** |
|  | הפונקציה מחזירה את הצומת המינימלי בערימה– עידו רוזינר | getMin() |
|  | הפונקציה מחזירה את האיבר הראשון בערימה – עידו רוזינר | getFirst() |
|  | הפונקציה מחזירה את מספר האיברים בערימה– עידו רוזינר | getSize() |
|  | הפונקציה מחזירה את מספר העצים בערימה – עידו רוזינר | getNumOfTrees() |
|  | הפונקציה מחזירה את מספר הצמתים הלא מסומנים בערימה – עידו רוזינר | getNonMakred() |
|  | הפונקציה מחזירה את הדרגה הגבוה ביותר בערימה – עידו רוזינר | getHighestDeg() |
|  | הפונקציה מחזירה את מספר החתכים בערימה. עידו רוזינר. | getCuts() |
|  | הפונקציה מחזירה את מספר הקשרים שיש בערימה– עידו רוזינר | setlinks() |
|  | הפונקציה מעדכנת את הצומת המינימלי בערימה | SetMin(HeapNode min) |
|  | הפונקציה מעדכנת את הצומת הראשון בערימה. | setFirst(HeapNode first) |
|  | הפונקציה מקבעת את גודל הערימה להיות size. – עידו רוזינר | setSize(int size) |
|  | הפונקציה מקבעת את את מספר העצים להיות numOfTrees . – עידו רוזינר | setNumOfTrees(int numOfTrees) |
|  | הפונקציה מקבעת את מספר הצמתים שלא סומנו להיות nonMarked. – עידו רוזינר | setNonMarked(int nonMarked) |
|  | הפונקציה מקבעת את הדרגה הגבוהה ביותר להיות highestDeg – עידו רוזינר | setHighestDeg(int highestDeg) |
|  | הפונקציה מעלה ב-1 את מספר החתכים | incCuts() |
|  | הפונקציה מעלה ב-1 את מספר הקשרים | intLinks() |
|  | הפונקציה מחזירה true אם הערימה ריקה, אחרת מחזירה false. | isEmpty |
|  | להוסיף | insert(int key) |
| להוסיף | הפוקנציה קוראת ל removeMIn() ולconsolidate() | deleteMin() |
| להוסיף |  | removeMin() |
| להוסיף |  | Consolidate() |
|  | במידה וisEmpty תחזיר true נחזיר null, אחרת נחזיר את העריך מgetmin()- – עידו רוזינר | findMin() |
|  | הפונקציה מחברת בין שני עצים לעץ אחד- עידו רוזינר | Meld(fibonacciHeap heap2) |
|  |  | meldnode |
|  | הפונקציה מחזירה מעריך שבמקום הi מחזירה את מספר העצים מסדר i בערימה. עידו רוזינר | countersRep() |
| כי במקרה הגרוע ביותר נצטרך לעבור על כל האיברים, אבל כפי שלמדנו לאחר המחיקה מספר העצים יצטמצם ולכן זמן הamortized יהיה | הפונקציה מעדכנת את הערך השל העץ להיות כזה שיגרום לו להיות המינימלי, ואז מפעילה את deleteMin(אם הוא היה כזה מלחתחילה, פשוט נפעיל ישר את deletemin). תומר רודניצקי | delete() |
| במקרה הגרוע ביותר  אך זמן האמורטייז יהיה o(1) כפי שלמדנו. | הפונקציה מקטינה את ערך המפתח בdelta. מתבצעת קריאה לcascadeCut | decreaseKey(HeapNode x, int delta) |
|  |  | cascadeCut(HeapNode x) |
|  |  | cutAndInsertHeap(HeapNode x |
|  | מפעילה את getNonMakred עידו רוזינר | nonMarked() |
|  | הפונקציה מחזירה את הפוטנציאל של הערימה על פי הנוסחה. עידו רוזינר | Potential() |
|  | הפונקציה קוראת לgetLinks()-עידו רוזינר | totalLinks() |
|  | הפונקציה קוראת לgetCuts()-עידו רוזינר | totalCuts() |
| O(k\*deg(H)) | הפונקציה מחזירה מערך המכיל את את k הערכים הקטנים בערימה. עבור k=0 וk=1 טיפלנו ידנית. אחרת, נאתחל ערימת פיבונאצי של מועמדים להכנסה לרשימה. נתחזק מצביע לערך המינימלי של הערימה. ואז בדומה מאוד לתרגיל בית 4, נמחק ונעביר את האיבר המינימלי בערימת המועמדים ונעבירו למערך. התהליך חוזר על עצמו. | kMin(FibonacciHeap H, int k) |

**להלן תיעוד HeapNode:**

|  |  |  |
| --- | --- | --- |
| **חישוב סיבוכיות זמן** | **תיאור** | **פונקציה** |
|  | הפונקציה מחזירה את המפתח של הצומת – עידו רוזינר | getKey() |
|  | הפונקציה מחזירה את הדרגה של הצומת– עידו רוזינר | getRank() |
|  | הפונקציה מחזירה אם הצומת מסומן או לא – עידו רוזינר | getMark() |
|  | הפונקציה מחזירה את הבן של הצומת, במידה ואינו קיים מחזירה null – עידו רוזינר | getChild() |
|  | הפונקציה מחזירה את הצומת הבא אחרי הצומת node- עידו רוזינר | getNext() |
|  | הפונקציה מחזירה את הצומת שבא לפני הצומת node- עידו רוזינר | getPrev() |
|  | הפונקציה מחזירה את ההורה של הצומת node- עידו רוזינר | getParent() |
|  | הפונקציה מחזירה את המצביע שנעזר בו עבור kmin – תומר רודניצקי | getpointerKMin() |
|  | הפונקציה מעדכנת את ערך המפתח של הצומת להיות key– עידו רוזינר | setKey(int key) |
|  | הפונקציה מעדכנת הדרגה של הצומת להיות rank- עידו רוזינר | setRank(int rank) |
|  | הפונקציה מעדכנת אם הצומת מסומנת או לא בהתאם לmark– עידו רוזינר | setMark(boolean mark) |
|  | הפונקציה מעדכנת את הבן של הצומת להיות child-– עידו רוזינר | setChild(HeapNode child) |
|  | הפונקציה מעדכנת את הצומת הבאה אחרי הצומת הנתונה להיות child- – עידו רוזינר | setNext(HeapNode next) |
|  | הפונקציה מעדכנת את הצומת שמגיעה לפני הצומת הנתונה להיות prev- – עידו רוזינר | setPrev(HeapNode prev) |
|  | הפונקציה מעדכנת את ההורה של הצומת הנתונה להיות parent – עידו רוזינר | setParent(HeapNode parent) |
|  | הפונקציה מעדכנת את המצביע שנעזר בו עבור kmin – תומר רודניצקי | setpointerKMin(HeapNode node) |
|  | הפונקציה מקשרת בין שני עצים באמצעות קישור בין שני צמתים כפי שלמדנו בהרצאה– עידו רוזינר | link(HeapNode x, HeapNode y) |