Empy() – אם הmin מצביע לnull אז empty.

למה זה לא יעבוד : אולי לא עדכנו את המינימום.(השדה של min אמור להיות מאותחל לnull).

**שדות נוספים: marked וtrees.**

Insert(int keyToBe) – מכניסים איבר כעץ בודד לרשימה Q, מעדכנים את השדה (לאיבר( min ( בהשוואה לאיבר שהכנסנו).

-לעדכן את הSIZE, trees=trees+1

למה זה לא יעבוד :לא חיברנו את האיבר לרשימה המקושרת מבחינת המצביעים (prev,next).

**Delete-min() –**

אם הsize==1 אז min=null.

אחרת, נבדוק אם לmin יש ילדים- אם כן -נעלה את הילדים של המינימום להיות איברים ברשימה הראשית(באמצעות concate(min.child,true) ונעדכן את הmark של כל הילדים לfalse ונוריד את השדה mark כמספר הילדים **המסומנים**. נשמור את originMin = this.min , נגדיר את השדה MIN להצביע על this.min.next , נבצע disconnect(originMin) ונחפש את המינימום החדש בלולאה באמצעות **checkNDupdateMin, ונקרא לsuccesiveLinking(heap).**

-לעדכן את הSIZE, trees= trees-1+numOfNode.Child

למה זה לא יעבוד : בעייה בconcate, checkNDupdate, successiveLinking.

**find-min()** – return min

למה זה לא יעבוד : לא עידכנו את הmin (שדה) – בdeletemin בdecreasekey(רק אם הdecreased key הוא המינימום- אחרי ההורדגה), ובInsert (אם האיבר שהוכנס קטן מהמינימום) ובmeld (השוואת שני הmins וקביעת המינימום החדש)ובdelete (אם מכניסים מפתח שהוא הmin).

**checkNDupdateMin(heapNode nodeToCheck)** – השוואה של ה nodeToCheck.key לthis.min.key והחלפת המינימום בהתאם.

למה זה לא יעבוד : בדיקה האם הthis.min!=null אז נשווה, אחרת, this.min= nodeToCheck.key.

**Concate(LinkedList listToConnect, Boolean concateAllList) -** תחבר את הרשימה שמקבלת לThis (גם רשימה מקושרת), בלי לשנות שדה מינימום.

למה זה לא יעבוד : לconcate יכולה להיות nullPointer אם שלחנו רשימה ריקה או שהרשימה שלנו ריקה (אי אפשר לעשות .next ). לא יעצור אחרי מעבר של פעם אחת על הרשימה (אין תנאי עצירה נכון)- רשימה דו-כיוונית).

**Meld(F-heap** **heapForMeld)** – יקרא לconcate(heapForMeld ,true) עם **heapForMeld** וישנה את המצביע של המינימום בקריאה ל **checkNDupdateMin**.

למה זה לא יעבוד : שגיאה מהconcate.

Size() – מחזיר את מספר האיברים בערימה (מספר הnodes)– שדה של הHeap.

למה זה לא יעבוד :לא עידכנו את השדה.

countersRep() – אתחול מערך בגודל logn של מונים (לקרוא לחבילה חיצונית math), ומעבר בלולאה על שורשי העצים (הרשימה המקושרת) ובדיקת מספר הילדים (השדה rank) והגדלת המונה באינדקס הרלוונטי ב-1.

למה זה לא יעבוד : rank לא מעודכן נכון, נעבור על איבר ברשימה יותר מפעם אחת (לא נעצור), נקבל רשימה ריקה (null.rank).

delete(heapNode nodeToDelete) – נבצע:

decreaseKey(nodeToDelete,1+ nodeToDelete.key-min) ונעשה deleteMin עם ה nodeToDelete

-ה-SIZE אמור להתעדכן בdeletemin.

למה זה לא יעבוד :בעיה בdercreaseKey או בdeletemin.

**decreaseKey**(HeapNode node, int delta) – אם הnode הוא שורש עושים decreaseKey בלי לנתק, כלומר רק משנים את הkey.

מעדכנים את הnode.key להיות שווה לnode.key-delta. ובודקים האם node.key (המעודכן) קטן מהnode.parent, אם כן – קוראים לcascading(node), אחרת סיימנו. ומעדכנים את הmin בערימה **checkNDupdateMin(node).**

**Cascading**(HeapNode node) – ,parent = node.parent ומוסיפים לרשימה באמצעות

Concate( **Disconnect(heapNode node)**,false) (לנתק מparent) לעדכן את הmark של node להיות false (כי הוא שורש כעת) ולהגדיל את השדה cuts+1– נעדכן את השדה marked פחות 1 ממה שהיה. ואז בודקים האם האב היה מסומן, אם לא- בודקים האם הוא שורש – אם כן שורש, לא מסמנים, אחרת:( לא שורש) מסמנים - (mark=true – חשוב שיהיה מאותחל לfalse) ומוסיפים אחת לשדה marked- וסיימנו

לעדכן את הrank של הparent, אחרת , קוראים רקורסיבית ל cascading(parent).

למה זה לא יעבוד :

**Disconnect(heapNode node) –** מנתק את node מהרשימה המקושרת שבה הוא נמצא (עדכון מצביעים) ומחזיר את הnode (כדי שאח"כ נוכל להשתמש בו ב-

concate(**Disconnect(heapNode node)**,false)). צריך להוריד את הrank של הparent ב-1.

למה זה לא יעבוד: לא שינינו את המצביעים מהnodeToDisconnect.

potential() – return this.trees +2\*this.marked

למה זה לא יעבוד : עידכונים של השדות או רשימה ריקה.

**successiveLinking-** יש פסאודו קוד במצגת

totalLinks() – מחזיר שדה links.

למה זה לא יעבוד : עדכון לא מלא של השדה.

totalCuts() – מחזיר את שדה cuts.

למה זה לא יעבוד : עידכון לא מלא של השדה.