**תיעוד + סיבוכיות זמן:**

**המחלקה FibonacciHeap :**

כללי:

לכל אובייקט במחלקה נתחזק את השדות הבאים:

* Size – מספר איברי הHeapNode במבנה הנתונים
* numTree- ספירת העצים תתוחזק ותישמר כשדה
* numMarked- בכל רגע נתון נדע את מספר הצמתים המסומנים
* Links- כמות החיבורים הכוללת שנעשתה
* Cuts- כמות הניתוקים הכולל
* Start- מצביע לעץ הראשון
* Tail- מצביע לעץ האחרון
* Min- HeapNode בעל מפתח מינימלי

לאורך כל הפונקציות שמשנות את מצב העץ נתחזק את כלל השדות הנ"ל.

כמו כן עבור המחלקה **HeapNode**, לכל אובייקט נתחזק את השדות:

* Child
* Parent
* Prev, next
* Marked
* Origin- נועד בשביל kMin, לכל צומת שנבנה נחזיק מצביע לצומת ה"אמיתית".
* Key
* rank

פונקציות:

Insert:

3 פונקציות בעלות אותו השם עם חתימות שונות (העמסה):

1. הפונקציה מקבלת מפתח ויוצרת Node חדש מסוג HeapNode. לאחר מכן קוראת לinsert עם הצומת.
2. פונקציה שמשמשת לkMin- מקבלת מפתח וצומת מקורי ומעדכנת את שדה הorigin של הצומת החדשה לצומת המקורית, לאחר מכן תקרא לinsert עם הצומת החדשה.
3. **Insert ראשית**- מקבלת צומת. אם העץ ריק או שהמפתח קטן מהמינימום היא תעדכן את המינימום, ותקרא לadd at start. כמו כן, תעדכן את מספר הצמתים (size) ומספר העצים.

פונקציות add וinsertbefore:

Add at start:

קוראת לaddat עם הצומת ומעדכנת את השדה start לצומת.

Add at tail:

קוראת לaddat ומעדכנת את השדה tail לצומת.

Addat:

אם העץ ריק תקרא לinsertBefore רק עם הצומת (כך שinsert תחבר את הצומת לעצמה- רשימה מקושרת),ותעדכן את שדות הstart וtail. אחרת תקרא לinsertBefore עם הצומת והצומת ההתחלתית של העץ (this.start).

insertBefore:

הפונקצייה מקבלת 2 צמתים – הצומת שיש להכניס-newNode והצומת שלפניה נכניס- position.

אם זו אותה צומת- נחבר אותה לעצמה בעזרת setnext, setprev.

אחרת, נחבר את הצומת לprev של position ולposotion בקשר כפול.

סיבוכיות כוללת של insert- .

בהתאם להסברים, כלל הפעולות מתבצעות בזמן קבוע, ללא לולאות ואיטרציות שתלויות בקלט.

DeleteMin:

הפעולה תקרא לפונקציות deleteMin, DeleteMinFunc, consolidation, link, put family, link and put, fixlistofroots.

נפרט על כל פונקציה בנפרד: