**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:

isEmpty():

בודקת האם השדה this.start==null אם כן, תחזיר true, כלומר העץ ריק, אחרת false.

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,makeUnmarked, link, put family, linkAndPut, fixListOfRoots.

נפרט על כל פונקציה בנפרד:

deleteMin:

מעטפת, אם העץ אינו ריק, נקרא לdeleteMinFunc, נוריד את גודל העץ ב1 ונקרא לconsolidation.

deleteMinFunc:

נחלק למקרים:

* מספר העצים הוא 1- אם אין ילדים- העץ יהיה ריק ולכן נעדכן את ההתחלה והסוף לnull. אחרת, נגדיר את הילד השמאלי להיות הstart והprev שלו לtail.
* אחרת, אם האיבר בstart נגדיר את start להיות הnext שלו, ובאותו אופן לגבי tail. כעת נבדוק לגבי ילדיו, אם אינם קיימים רק נחבר את הprev עם הnext שלו, אחרת, נקרא לputFamily עם הprev, next והילד השמאלי.

*putFamily:*

*פונקציה שנועדה, בעת מחיקת השורש, לחבר את הילדים לprev ולnext של השורש.*

*הפונקציה מקבלת 3 HeapNode- prev,next,child ומעדכנת את החיבור של ה"משפחה" במקום הצומת שנמחק.*

*כמו כן, הפונקציה תעדכן את שדות הstart והprev במידת הצורך.*

*Consolidation:*

*מטרתה לבצע successive linking ולמצוא את המינימום החדש.*

*הפונקציה בונה מערך של צמתים בגודל 2\*log(this.size). כפי שראינו בהרצאה, לכל הצמתים בעץ יש rank שאינו עולה על 1.4404 log2n, ולכן לכל עץ יהיה מקום ברשימה ע"פ הrank שלו.*

*כעת נעבור על הרשימה המקושרת החל מthis.start ועד הtail, נבצע לכל השורשים- makeUnmarked ונאתחל את הparent שלהם לnull.*

*לבסוף נקרא לlinkAndPut- פונקציה שתשים את הצמתים במערך לפי הrank, ובמידת הצורך- אם יש 2 צמתים באותה משבצת, תבצע להם link ותעבירם למשבצת הבאה.*

*כעת, לאחר שסיימנו ליצור את המערך, נקרא לfixListOfRoots עם המערך כך שנאתחל את העץ מחדש עם העצים החדשים שחיברנו.*

*makeUnmarked:*

*מקבלת צומת ומעדכנת את השדה marked שלה לfalse. כמו כן, הפונקציה בודקת אם השדה היה true ואם כן מורידה את numMarked ב1.*

*linkAndPut:*

*פונקציה רקורסיבית.*

*הפונקציה מקבלת את המערך basket שיצרנו בconsolidation וצומת חדש שיש להכניס למערך במיקום במערך השווה לrank של הצומת.*

*כעת, אם המקום מלא- היא תקרא לlink של הצומת בשמערך והצומת שלנו, תהפוך את המקום במערך לnull ותקרא לlinkandput עם הצומת לאחר חיבור העצים.*

*אם המקום ריק, פשוט תמקם את הצומת במערך במקום זה.*

*Link:*

*חיבור 2 עצים עם שורשים בעלי אותו הrank.*

*תחילה הפונקציה תעדכן את שדה הlinks לפלוס אחד והnumTrees למינוס אחד.*

*לאחר מכן תגדיר את node1 להיות הצומת עם המפתח הקטן מבין השניים.*

*כעת, אם הrank גדול מאפס יש לחבר את node2 לילדים של node1, אחרת יש לחבר את node2 לעצמה ולכן נקרא לinsertBefore שתבצע זאת.*

*לבסוף נבצע את החיבורים בין node1 לnode2 כך שnode1 יהיה האבא של node2, נגדיל את הrank של node1 ב1 ונחזיר את שורש העץ החדש- node1.*

*fixListOfRoots:*

*לאחר ששמנו את כל העצים במערך וביצענו את החיבורים הנדרשים, נותרנו עם לכל היותר עץ אחד מכל rank.*

*כעת, הפונקציה תעדכן את העץ כך שתעבור על מערך הbasket.*

*תחילה נאתחל את הערימה כך ששדות הstart וtail הם null.*

*כאשר ניתקל באיבר הראשון במערך שאינו null נעדכן את הstart להצביע על הצומת הזו.*

*ומכאן והלאה נחבר את כל הצמתים במערך (שאינם null), לפי הסדר ברשימה מקושרת דו כיוונית החל מהstart.*

*לבסוף נעדכן את tail להיות הצומת האחרון.*

*במהלך המעבר על הצמתים נחפש את הצומת עם המפתח המינימלי ונעדכן את המינימום להצביע אליו.*

*סיבוכיות....*

*findMin:*

*אם העץ ריק, תחזיר null אחרת תחזיר את this.min – שדה אשר מתוחזק בעת פעולות על הערימה.*

*סיבוכיות: O(1)*

*Meld:*

*מקבלת ערימה נוספת וממזגת עם הערימה הקיימת.*

*אם הערימה הקיימת ריקה, תעעדכן את שדות הstart והtail לשדות אלה בערימה החדשה.*

*אחרת, תקרא לputFamily עם הtail,start,heap2.getstart כך שנחבר לרשימה המקושרת של הערימה הנוכחית את החדשה בסופה.*

*נעדכן את הsize, numtree, nummarked לחיבור השדות של שתי הערימות.*

*כל הפעולות נעשות בזמן קבוע, סיבוכיות: O(1)*

*Size:*

*מחזירה את this.size – מספר הצמתים בערימה.*

*סיבוכיות: O(1)*

*Delete:*

*נגדיר את diff להיות ההפרש בין המפתח של הצומת למפתח של המינימום ונבצע decreaseKey לצומת עם 1+diff. כך, הצומת יהיה בעל המפתח המינימלי וכשנקרא לdeletemin נמחק אותו.*

*decreaseKey:*

*תחילה הפונקציה תעדכן את המפתח של הצומת לערך החדש, ע"י setKey.*

*אם המפתח החדש קטן מהמינימום, נעדכן את המינימום אליו.*

*כמו כן, אם הצומת אינה שורש והמפתח החדש שלה קטן מאביה אז נקרא לcascadingCut.*

*cascadingCut:*

*פונקציה רקורסיבית.*

*נבצע cut בין הצומת לאביו.*

*כעת, נבדוק האם האב מסומן. אם לא, נסמנו ונעלה את מונה הnumMarked.*

*אם כן, נקרא רקורסיבית לcascadingCut עם האב ואביו.*

*Cut:*

*ביצוע החיתוך עצמו בין הצומת לאב.*

*נעדכן את שדה הparent של הצומת לnull ונקרא לmakeUnmarked.*

*נוריד את הrank של האב של הצומת ב1.*

*כעת נחלק למקרים לפי מספר הילדים של האב:*

*אם זה הבן היחיד, נקבע את הילד של האב לnull.*

*אחרת אם הchild של האב זה הצומת נעדכן לnext שלו.*

*בנוסף, נחבר את האחים של הnode אחד לשני ברשימה המקושרת (נמחק את הnode משם).*

*נקרא לaddAtStart עם הצומת ונעלה את מספר העצים והcuts.*

*Potential:*

*נחזיר את numTree+2\*numMarked.*

*שדות של הערימה אותם תחזקנו לאורך הפונקציות השונות.*

*סיבוכיות: O(1)*

*totalLinks:*

*פונקצייה סטטית שתחזיר את כלל החיבורים שבוצעו.*

*לשם כך אתחלנו שדה סטטי ובכל פעולת לינק אנו מעלים את ערכה ב1.*

*סיבוכיות: O(1)*

*totalCuts:*

*בדומה, כל פעולות החיתוך שהתבצעו.*

*אתחלנו שדה סטטי ובכל פעולת חיתוך נעלה את ערכה ב1.*

*סיבוכיות: O(1)*