מחלקת Heapnode

תיאור כללי: מחלקה פנימית המייצגת צומת בערמת פיבונאצ'י.

שדות:

שם	תיאור
int key	המפתח של הצומת
int rank	הדרגה של הצומת
Boolean mark	אינדיקטור האם הצומת מסומן או לא
HeapNode child	הילד של הצומת
HeapNode next	מצביע לצומת הבא בכל רמה
HeapNode prev	מצביע לצומת הקודם בכל רמה
HeapNode parent	מצביע להורה של הילד
HeapNode node_reference	ישמש כמצביע מצומת בערמת העזר לצומת המקורית בפונקציה kMin

בנאי:

הבנאי מקבל רק מפתח. הוא מאתחל את המצביעים ל-null, את המפתח למפתח שהתקבל ואת הדרגה לאפס.

פונקציות::

לכל השדות יש פונקציית get ו-set.

מחלקת Fibonacci Heap

שדות:

שם	תיאור
HeapNode min	המינימום של הערמה
HeapNode first	מצביע לשורש של העץ הראשון משמאל
int size	מספר הצמתים בערמה
static final double GOLDEN_RATIO	יחס הזהב
int marked_count	מונה למספר הצמתים המסומנים
int num_of_trees	מונה מספר העצים בעץ
static int total_cuts	מספר החיתוכים שביצענו בתוכנית
static int total_links	מספר ה-links שעשינו בתוכנית

הבנאים:

בנאי שמאתחל את המצביעים ל-null, את הגודל ואת מספר העצים ל-0 ואת מספר המסומנים באפס.
המשתנים הסטטיים מאותחלים דיפולטית לאפס.

<u>פונקציות שהתבקשנו לממש:</u>

מס"ד	שם	פרמטרים	טיפוס החזרה	תיאור	סיבוכיות
1	isEmpty		boolean	מחזירה אמת אמ"מ הערימה ריקה.	$O(1)$

<u>תנאי קדם:</u> אין.

<u>אופן פעולה:</u> מחזיר אמת אמ"מ השדה first הוא null.

<u>ניתוח סיבוכיות:</u> כלל הפעולות בעלות קבועה ולכן הסיבוכיות היא $O(1)$.

מס"ד	שם	פרמטרים	טיפוס החזרה	תיאור	סיבוכיות
2	insert	int i	Heapnode	הפונקציה יוצרת צומת חדשה עם המפתח i, מכניסה אותו לערמה ומחזירה את הצומת שנוצר.	$O(1)$

<u>תנאי קדם:</u> אין

<u>אופן פעולה:</u> ההכנסה הינה הכנסה עצלה. מכניסים את הצומת כראשון משמאל בערמה ומעדכנים את השדות בהתאם.

<u>ניתוח סיבוכיות:</u> כלל הפעולות בעלות קבועה ולכן הסיבוכיות היא $O(1)$.

מס"ד	שם	פרמטרים	טיפוס החזרה	תיאור	סיבוכיות
3	deleteMin			מחיקת הצומת עם המפתח המינימלי מבין המפתחות שבערמה.	$O(\mathrm{n})$

<u>תנאי קדם:</u> אין

<u>אופן פעולה:</u> מוחקים את המינימום והופכים את ילדיו לשורשים של עצים חדשים. לאחר מכן מוצאים את המינימום החדש ומבצעים קונסולידציה.

<u>ניתוח סיבוכיות:</u> במקרה הגרוע נצטרך לעבור על $n-1$ צמתים כדי למצוא את המינימום החדש. לכן, סיבוכיות זמן הריצה במקרה הגרוע היא $O(n)$.

מס"ד	שם	פרמטרים	טיפוס החזרה	תיאור	סיבוכיות
4	findMin		Heap Node	החזרת הצומת שהמפתח שלו מינימלי מבין המפתחות שבערמה, מערמה ריקה מוחזר null.	$O(1)$

<u>תנאי קדם:</u> אין

אופן פעולה: מחזירים את שדה המינימום.

ניתוח סיבוכיות: כלל הפעולות בעלות קבועה ולכן הסיבוכיות היא $O(1)$.

מס"ד	שם	פרמטרים	טיפוס החזרה	תיאור	סיבוכיות
5	meld	FibonacciHeap heap2		מיזוג הערמה heap2 עם הערמה הנוכחית.	$O(1)$

תנאי קדם: אין

אופן פעולה: משרשרים יחד את שתי הערמות לערמה אחת-heap2 מימין לערמה המקורית.

ניתוח סיבוכיות: כלל הפעולות בעלות קבועה ולכן הסיבוכיות היא $O(1)$.

מס"ד	שם	פרמטרים	טיפוס החזרה	תיאור	סיבוכיות
6	size		int	החזרת מספר האיברים בערמה.	$O(1)$

תנאי קדם: אין

אופן פעולה: החזרת השדה המתאים.

ניתוח סיבוכיות: כלל הפעולות בעלות קבועה ולכן הסיבוכיות היא $O(1)$.

מס"ד	שם	פרמטרים	טיפוס החזרה	תיאור	סיבוכיות
7	countersRep		int[ ]	הפונקציה מחזירה מערך ממוין המכיל את כל המפתחות בעץ, או מערך ריק אם העץ ריק.	$O(n)$

תנאי קדם: אין

אופן פעולה: מחפשים את הדרגה המקסימלית בעץ ומאתחלים מערך חדש בגודל שהדרגה ועוד 1. לאחר מכאן עוברים על כלל הצמתים. כל אינדקס במערך מייצג דרגה, בודקים את הדרגה של כל צומת ומעלים את הערך באינדקס המתאים בדרגה שלה ב-1.

ניתוח סיבוכיות: אנחנו עוברים על כל הצמתים פעמיים. כלומר מצבעים $2n$ איטרציות עם עלות קבועה בכל איטרציה. לכן סיבוכיות זמן הריצה היא $O(n)$.

מס"ד	שם	פרמטרים	טיפוס החזרה	תיאור	סיבוכיות
8	delete	HeapNode x		מוחקת את x מהערמה.	$O(n)$

תנאי קדם: אין

אופן פעולה: נבצע decreaseKey לצומת אותו אנו רוצים למחוק למינימום פחות אחד. לאחר מכאן נבצע deleteMin לצומת זו.

ניתוח סיבוכיות: שתי הפעולות שאנו מבצעים הן מסיבוכיות $O(n)$ ולכן זו סיבוכיות זמן הריצה שלנו.

מס"ד	שם	פרמטרים	טיפוס החזרה	תיאור	סיבוכיות
9	decreaseKey	HeapNode x int d		מורידה את הערך של המפתח של x ל-d	$O(n)$

תנאי קדם: אין

אופן פעולה: מקטינים את המפתח לערך הדרוש. אם תכונת הערמה מופרת מבצעים cascading Cuts (פונקציית עזר מספר 4).

ניתוח סיבוכיות: במקרה הגרוע אנחנו במצב של שרוך באורך n. לכן כאשר נבצע cascading Cuts נעבור על n צמתים במקרה הגרוע וסיבוכיות זמן הריצה תהיה $O(n)$.

מס"ד	שם	פרמטרים	טיפוס החזרה	תיאור	סיבוכיות
10	potential		int	מחזירה את הפוטנציאל של הערמה.	$O(1)$

תנאי קדם: אין.

אופן פעולה: מחשבים ומחזירים את הביטוי $\#trees + 2 * \#marked$

ניתוח סיבוכיות: כלל הפעולות בעלות קבועה ולכן הסיבוכיות היא $O(1)$.

מס"ד	שם	פרמטרים	טיפוס החזרה	תיאור	סיבוכיות
11	totalLinks		int	מחזירה את מספר ה-links שבוצעו בערמה.	$O(1)$

תנאי קדם: אין.

אופן פעולה: מחזירים את השדה.

ניתוח סיבוכיות: כלל הפעולות בעלות קבועה ולכן הסיבוכיות היא $O(1)$.

מס"ד	שם	פרמטרים	טיפוס החזרה	תיאור	סיבוכיות
12	totalCuts		int	מחזירה את מספר ה-cuts שבוצעו בערמה.	$O(1)$

תנאי קדם: אין

אופן פעולה:. מחזירים את השדה.

ניתוח סיבוכיות: כלל הפעולות בעלות קבועה ולכן הסיבוכיות היא $O(1)$.

מס"ד	שם	פרמטרים	טיפוס החזרה	תיאור	סיבוכיות
13	kMin	FibonacciHeap H int k	int[ ]	מקבלת ערימה H שהיא עץ מדרגה deg(H) ומספר חיובי k שקטן מהגודל של H. הפונקציה מחזירה מערך ממוין של K הצמתים הקטנים ב-H.	$O(k \cdot \deg(H))$

<u>תנאי קדם:</u> אין

<u>אופן פעולה:</u>. נאתחל ערמת עזר ריקה. נתחיל מהמינימום ונכניס את ילדיו לערמת העזר. נבצע מחיקת מינימום מערמת העזר, נכניס אותו לסוף המערך הממוין ואת ילדיו נכניס לערמת העזר. נמשיך כך באופן איטרטיבי עד שנמלא את המערך המוחזר ב-k איברים.

<u>ניתוח סיבוכיות:</u> נסמן $\deg(H) = h$. הערמה מכילה עץ בינומי יחיד ומתקיים דרגת כל צומת בעץ קטנה או שווה ל- h (דרגת השורש) מכיוון שלמדנו בכיתה כי דרגת השורש היא הגבוהה ביותר בעץ בינומי. טענה: בכל איטרציה כמות הצמתים בערמת העזר חסומה מלמעלה ע"י $h \cdot i$ כאשר i זה מספר האיטרציה הנוכחית. ואכן באיטרציה $i-1$ הוכנסו הבנים של הצומת שביצענו לו *deleteMin*, כלומר הוכנסו לכל היותר h איברים. האיברים ללא h האיברים שעתה נוספו נמצאים בלכל היותר $\log(h \cdot i)$ עצים. לפיכך, כאשר נבצע $deleteMin$ באיטרציה ה-i, עלותו תהיה:

$$O(h + \log(h \cdot i) = O(h + \log i + \log h) = O(h + \log k)) = O(h)$$

* $i < k$

* כפי שנלמד בכיתה בעץ בינומי יש 2^h צמתים ולכן $k < 2^h$.

לאחר פעולות אלו, נכניס $O(h)$ בנים לערמת העזר כאשר כל הכנסה בעלות קבועה. לכן עלות ההכנסות הכוללת היא $O(h)$. אנחנו מבצעים k איטרציות והלולאות מקוננות, ומשום כך סיבוכיות זמן הריצה הכוללת של הפונקציה היא: $O(h \cdot k) = O(k \cdot \deg(H))$ כנדרש.

<u>פונקציות עזר:</u>

מס"ד	שם	פרמטרים	טיפוס החזרה	תיאור	סיבוכיות
1	consolidation			ביצוע קונסולידציה לערמה כפי שנלמד בכיתה.	$O(\mathrm{n})$

<u>תנאי קדם:</u> אין

<u>אופן פעולה:</u> מאתחלים מערך בגודל של הדרגה המקסימלית האפשרית. עוברים על הערמה ומחברים עצים מאותה דרגה.

ניתוח סיבוכיות: במקרה הגרוע אנחנו נבצע חיבור ל-n צמתים מדרגה 0, פעולה שדורשת $n-1$ פעולות *link* (פונקציית עזר 2). לכן, סיבוכיות זמן הריצה היא $O(n)$.

מס"ד	שם	פרמטרים	טיפוס החזרה	תיאור	סיבוכיות
2	link	HeapNode smaller HeapNode bigger	HeapNode	חיבור שני עצים מאותה דרגה לעץ אחד.	$O(1)$

תנאי קדם: $a,b\ !=null\ \&\&\ a.rank=b.rank$

אופן פעולה:. בודקים מי העץ עם מספר הצמתים הגדול ביותר. הוא יהיה מצד ימין בזמן ה-link. מעדכנים את המצביעים בהתאם.

ניתוח סיבוכיות: כלל הפעולות בעלות קבועה ולכן הסיבוכיות היא $O(1)$.

מס"ד	שם	פרמטרים	טיפוס החזרה	תיאור	סיבוכיות
3	cut	HeapNode x HeapNode y			$O(1)$

תנאי קדם: $y\ !=null$

אופן פעולה: מנתקים את הצומת ממקומו המקורי, מורידים את מספר המסומנים אם x היה מסומן. מעדכנים את המצביעים ומכניסים את x כראשון בערמה.

ניתוח סיבוכיות: כלל הפעולות בעלות קבועה ולכן הסיבוכיות היא $O(1)$.

מס"ד	שם	פרמטרים	טיפוס החזרה	תיאור	סיבוכיות
4	cascading cuts	HeapNode x HeapNode y		ביצוע cascading cuts כפי שנלמד בכיתה.	$O(n)$

תנאי קדם: $y\ !=null$

אופן פעולה: חותכים את הצמתים המסומנים, וממשיכים מעלה לאבות שלהם עד שנתקלים בצומת לא מסומן.

ניתוח סיבוכיות:. במקרה הגרוע אנחנו במצב של שרוך באורך n. לכן כאשר נבצע cascading Cuts נעבור על n צמתים במקרה הגרוע וסיבוכיות זמן הריצה תהיה $O(n)$.

מס"ד	שם	פרמטרים	טיפוס החזרה	תיאור	סיבוכיות
5	addAsFirst	HeapNode x		מוסיפה את x כ-first.	$O(1)$

תנאי קדם: $x\ != null$

אופן פעולה: הוספת x כראשון בערמה ועדכון מצביעים בהתאם.

ניתוח סיבוכיות: כלל הפעולות בעלות קבועה ולכן הסיבוכיות היא $O(1)$.

חלק ניסויי/תאורטי

שאלה 1:

א)

- אנחנו מבצעים m הכנסות בעלות $O(1)$, סה"כ $O(m)$.
- *DeleteMin*- נעבור על כל צומת פעם אחת, ונחבר כל צומת פעם אחת לכל היותר, סה"כ $O(m)$.
- כעת פעולת *decreaseKey* מתבצעת על עלים בלבד, יש לנו $\log{(m)}$ פעולות. נעשה אותה, לאחר מכן נעשה cut והכנסה בהתחלה בעלות $O(1)$. סה"כ $\log{(m)}$.
  לכן נקבל כי העלות הכוללת היא $O(m)$.

ב)

m	Runtime(ms)	totalLinks	totalCuts	Potential
2^{10}	2.0	1023	10	29
2^{15}	15.0	32767	15	44
2^{20}	103.0	1048575	20	59
2^{25}	2720.0	33554431	25	74

ג) totalLinks- לאחר פעולת ה-insert נקבל $m+1$ צמתים מדרגה 0. לאחר מחיקת הצומת המינימלית נבצע consolidate על m צמתים (m הוא חזקה של 2) אשר תיתן לנו בסופה עץ בינומי יחיד. לכן ביצענו $m-1$ פעמים link.

totalCuts- כל הצמתים עליהם נעשה DecreaseKey יחתכו מכיוון שבביצוע הפעולה נהפוך אותם לשליליים ולפיכך הם בהכרח יהיו קטנים מההורים שלהם. בנוסף, כל הצמתים הם עלים ולכן לא ייוצר מצב שבו נבצע cut להורה מסומן. כלומר נבצע חיתוכים כמספר הdecreaseKey, log(m) פעמים.

Potential- הפוטנציאל הוא סכום מספר העצים ופעמיים מספר המסומנים. מספר העצים גדול ב-1 ממספר הפעמים שביצענו cut כלומר $\log(m)+1$ ומספר המסומנים הוא $\log(m)-1$ מכיוון שנסמן כל הורה שביצענו לו cut מלבד השורש, אותו לא נסמן. לכן סה"כ נקבל שפוטנציאל הוא:

$$1+\log m+2(\log m-1)=3\log(m)-1$$

ד) totalLinks- זהה לסעיף ג', עד שלב זה מבצעים את אותן הפעולות. 😊

totalCuts- כעת פעולת decreaseKey מתבצעת על הדופן השמאלית של העץ מלמעלה למטה. מכיוון שנוריד את **אותו** הערך מכל צומת ונתחיל את הפעולה מהצמתים העליונים יותר, כלל הערמה ישמר כנדרש ולא יתבצעו חיתוכים.

Potential- מכיוון שלא בוצעו חיתוכים נישאר עם עץ אחד ואפס צמתים מסומנים ולכן הפוטנציאל הוא 1.

ה) totalLinks- מכיוון שלא ביצענו deleteMin לא נבצע את פעולת ה-link. כלומר נבצע 0 פעולות link.

totalCuts- מכיוון שיש לנו $m+1$ עצים נפרדים מדרגה 0, בפעולת decreaseKey כלל הערמה נשמר ויתבצעו אפס חיתוכים.

Potential- לא ביצענו חיתוכים ולכן יהיו 0 מסומנים. בנוסף לא ביצענו מחיקה ולכן נשארנו עם $m+1$ עצים לאחר ההכנסה. כלומר סה"כ הפוטנציאל הוא $m+1$.

ו) totalLinks- זהה לסעיף ג', עד שלב זה מבצעים את אותן הפעולות.

totalCuts- ראשית נבצע חיתוכים כמו בסעיף ג, $log(m)$ פעמים. בפעולה שהוספנו נבצע decreaseKey על העלה הכי שמאלי וגם אותו נחתוך מכיוון שנקבל מספר שלילי ושאר הערכים בעץ אי שליליים. כעת כל ההורים במסלול מהעלה אל השורש מסומנים (מלבד השורש) ולכן נחתוך גם אותם. מספר הצמתים שחתכנו כולל הראשון שגרם לשרשרת החיתוכים הוא $log(m)-1$. לכן סה"כ נבצע $\log(\mathrm{m})+\log(m)-1=2\log(m)-1$

Potential- כל חיתוך יצור לנו עץ חדש ולכן יהיו $1+2\log(\mathrm{m})-1$ עצים. מספר המסומנים הוא 0 מכיוון שחתכנו את כל המסומנים בשרשרת חיתוכים האחרונה. לכן הפוטנציאל הוא $2\log(\mathrm{m})$

DecreaseKey max cost- ה-decreaseKey בשלב 3 חותך רק את העלים ומסמן את ההורים שלהם. לכן הdecreaseKey המקסימלי הוא בפעולה האחרונה בה נבצע $\log(\mathrm{m})-1$ חיתוכים וזו העלות המקסימלית.

טבלה מסכמת לסעיפים ג'-ו':

case	totalLinks	totalCuts	Potential	decreaseKey max cost
(c) original	m-1	$\log m$	$3\log(m)-1$	(Skip)
(d) decKey(m-2^i)	m-1	0	1	(Skip)
(e) remove line #2	0	0	m+1	(Skip)
(f) added line #4	m-1	$2\log(m)-1$	$2\log(m)$	$\log(m)-1$

שאלה 2:

א)

m	Runtime(ms)	totalLinks	totalCuts	Potential
3^6-1	4	723	0	6
3^8-1	9	6555	0	6
$3^{10}-1$	40	59040	0	9

$3^{12}-1$	230	531431	0	10
$3^{14}-1$	2243	4782955	0	14

ב) אנו מבצעים $m+1$ פעולות insert שכל אחת עולה $O(1)$, לכן העלות הכוללת היא $O(m)$.
פעולת deleteMin הראשונה תעלה $O(m)$ משום שיש לנו $m+1$ עצים שכן העצים מדרגה 0.
לאחר מכן, כאשר נבצע פעולת *consolidate* יהיו לנו $O(\log m)$ עצים מכיוון שאנו מוחקים $\frac{3}{4}m$ צמתים ולפיכך יש לנו לכל הפחות $\frac{m}{4}$ צמתים. כלומר, פעולות ה-*deleteMin* יעלו $O(\log m)$. אנו מבצעים $\frac{3m}{4}-1$ פעולות כאלו. כלומר סיבוכיות זמן הריצה הכוללת היא $O(m\log(m)+1)$.

ג)

$$totalCuts = 0$$

$$potential = number\ of\ 1\ in\ binary\ representation\ \ of\ \frac{m}{4}+1$$

$$totalLinks = m - \ number\ of\ 1\ in\ binary\ representation\ \ of\ m$$

לא מבצעים *cuts* מכיוון שמוחקים את המינימום שהוא השורש ולכן *totalCuts* הוא אפס. כתוצאה מכך, לא מסמנים צמתים. כלומר, הפוטנציאל בסוף התהליך הינו מספר העצים. מספר הצמתים לאחר ההכנסות הוא $m+1$. מחקנו $\frac{3m}{4}$ צמתים ובסוף התהליך נשארו עם $\frac{m}{4}+1$ צמתים. לאחר סדרת מחיקות המינימום מקבלים ערמה בינומית. כפי שלמדנו בכיתה, מספר העצים בערמה בינומית שווה למספר האחדים בייצוג הבינארי של מספר הצמתים. לכן, הפוטנציאל הוא מספר אחדות בייצוג הבינארי של $\frac{m}{4}+1$.
את ההכנסות ביצענו בסדר עולה, ולכן לאחר המחיקה הראשונה, המינימום יהיה בערמה הקטנה ביותר (השמאלית ביותר), וכאשר נמחק את המינימום נקבל עצים שקטנים משאר העצים בערמה, ולכן לא יתכן שיהיו לנו עצים באותו הגודל. לפיכך, ה-*links* היחידים שנבצע יקרו לאחר פעולת המחיקה הראשונה. לאחר שמחקנו את המינימום אנחנו מחברים m עצים. בסוף התהליך נקבל ערמה בינומית. כפי שהסברנו מקודם, מספר העצים בערמה בינומית הוא מספר האחדות. לכן, מספר ה-*links* הוא m פחות מספר האחדות בייצוג הבינארי של m.