timore – 308145309 – תימור איזנמן
offekgil – 308315092 – אופק גיל

מחלקת העל – FibonacciHeap

שדות

`private static final`	`double PHI = 1.6180339887`	משתנה אשר יחזיק את יחס הזהב, לטובת חישובי גדלי המערכים בהמשך.
`Private`	`HeapNode min`	מצביע לאיבר בעל המפתח המינימלי בערימה.
`private`	`int size`	גודל הערימה.
`private static`	`int links = 0`	משתנה סטטי אשר יחזיק את כמות הלינקים שנעשו מתחילת התוכנית.
`private static`	`int cuts = 0`	משתנה סטטי אשר יחזיק את כמות החיתוכים שנעשו מתחילת התוכנית.
`Private`	`int numTrees = 0`	משתנה אשר יחזיק את כמות העצים בערימה.
`private`	`int numMarked = 0`	משתנה אשר יחזיק את כמות הצמתים המסומנים בערימה.

מתודות

בנאי ראשון

- קלט: אין.
- פלט: אין
- שיטת פעולה: הגדרת ערימה ריקה לחלוטין.
- סיבוכיות: $O(1)$.

בנאי שני

- קלט: מספר המייצג את המפתח של הצומת הראשון בעץ.
- פלט: אין.
- שיטת פעולה: יצירת HeapNode חדש עם המפתח החדש, כאשר באופן דיפולטי נגדיר לו next, prev כהוא עצמו, על מנת לשמור על קישוריות הרשימה.
- סיבוכיות: $O(1)$.

Empty()

- קלט: אין.
- פלט: ערך בוליאני אשר מייצג האם הרשימה ריקה או לא.
- שיטת פעולה: בדיקה האם הצומת המינימלי הוא null, אם כן אז ריק, אחרת לא ריק.
- סיבוכיות: $O(1)$.

FindMin()

- קלט: אין.
- פלט: עצם מסוג HeapNode המייצג את האיבר המינימלי בערימה.
- שיטת פעולה: מחזיר את העצם אשר המצביע min מצביע אליו.
- סיבוכיות: $O(1)$.

Size()

- קלט: אין.
- פלט: מספר המייצג את כמות האיברים בערימה.
- שיטת פעולה: מחזיר את הערך size של הערימה.
- סיבוכיות: $O(1)$.

Potential()

- קלט: אין.
- פלט: מחזיר את הערך מספרי של פונקציית הפוטנציאל.
- שיטת פעולה: מחשב את הערך על פי השדות השמורים של הערימה: numTrees, numMarked ומחזיר אותו.
- סיבוכיות: $O(1)$.

totalLinks()

- קלט: אין.
- פלט: מספר הלינקים שנעשו במהלך הרצת התוכנית.
- שיטת פעולה: מחזיר את הערך מהמשתנה הסטטי links של המחלקה FibonacciHeap.
- סיבוכיות: $O(1)$.

totalCuts()

- קלט: אין.
- פלט: מספר החיתוכים שנעשו במהלך הרצת התוכנית.
- שיטת פעולה: מחזיר את הערך מהמשתנה הסטטי cuts של המחלקה FibonacciHeap.
- סיבוכיות: $O(1)$.

SuccessiveLinking()

- קלט: אין.
- פלט: אין.
- שיטת פעולה:
  - תחילה אנו יוצרים מערך של HeapNode בגודל $\left(\left\lceil \log_{\phi} size \right\rceil + 1\right)$ אשר כל אינדקס של תא מייצג דרגה מסוימת.
  - כעת, עבור כל עץ בערימה, נבדוק את דרגתו, ונבדוק מול המערך האם יש צורך באיחוד או לא.
    - במידה ולא, התא המתאים ישמור מצביע לשורש העץ הנ״ל.
    - במידה וכן, נקרא למתודה mergeTrees עם העץ שנמצא בתא המתאים ועם העץ שבדקנו כעת, ונעביר אותו לתא המתאים לדרגה הקודמת + 1.
    - לאחר מכן, נבדוק האם יש צורך באיחודים נוספים, ונבצע אותם במידת הצורך באותה הצורה.
- סיבוכיות: $O(n)$.

timore – 308145309 – תימור איזנמן
offekgil – 308315092 – אופק גיל

<table>
<tr><td><u>mergeTrees</u></td><td>• קלט: שני עצמים מסוג HeapNode המייצגים את שני השורשים של העצים עלינו לאחד.
• פלט: עצם מסוג HeapNode המייצג את השורש החדש לאחר האיחוד.
• שיטת פעולה:
○ תחילה, על מנת לשמור על אחידות, נייצר את המצב שבו a הוא בעל המפתח הקטן יותר ו-b הוא בעל המפתח הגדול יותר.
○ כעת, נשתמש במתודה skipNode על מנת להוציא את הצומת b מרשימת השורשים, ותיאום השורש הקודם והבא שלו להכיר אחד את השני.
○ לאחר מכן, נפצל לשני מקרים:
▪ אם שני הצמתים מדרגה 1, אזי נחבר ביניהם ביחסי אב-בן.
▪ אחרת, נשתמש במתודה listLinkNode על מנת להכניס את צומת b לרשימה המקושרת של ילדי a.
○ לבסוף, נעלה את דרגתו של a ב-1, מכיוון שקיבל אליו בן חדש ונחזיר אותו בחזרה.
• סיבוכיות: $O(1)$.</td></tr>
<tr><td>SkipNode()</td><td>• קלט: עצם מסוג HeapNode אשר מייצג את הצומת עליה אנו רוצים לדלג ברשימת השורשים.
• פלט: אין.
• שיטת פעולה:
○ משימים ל-next את prev כצומת קודם.
○ משימים ל-prev את next כצומת אוחר.
• סיבוכיות: $O(1)$.</td></tr>
<tr><td>listLinkmode()</td><td>• קלט: שני עצמים מסוג HeapNode האחד מייצג את הצומת החדש שצריך להיכנס לרשימה המקושרת, והאחר מייצג עצם מתוך הרשימה שבעזרתו נכניס את הצומת לרשימה.
• פלט: אין.
• שיטת פעולה:
○ "מכניסים" את הצומת החדש בין הצומת המייצג של הרשימה לבין קודמו על ידי השמות של next, prev מתאימים.
• סיבוכיות: $O(1)$.</td></tr>
<tr><td><u>countersRep</u></td><td>• קלט: אין.
• פלט: מערך המייצג עבור כל אינדקס, כמה עצים מדרגת האינדקס קיימים בעץ.
• שיטת פעולה:
○ עבור כל שורש ברשימת השורשים נבדוק את דרגתו ונעלה ב-1 את הערך של התא באינדקס המתאים.
• סיבוכיות: $O(n)$.</td></tr>
<tr><td><u>insert</u></td><td>• קלט: מספר המייצג את המפתח אותו אנו רוצים להכניס לערימה.
• פלט: עצם מסוג HeapNode המייצג את הצומת החדש שנכנס.
• שיטת פעולה:
○ תחילה, נבדוק האם הערימה ריקה, במידה וכן – נכניס אותו כ-min.
○ אחרת, תוך שימוש במתודה listLinkNode נכניס את הצומת החדש שיצרנו לתוך רשימת השורשים, ליד השורש min.
▪ לאחר מכן, נבדוק האם הצומת החדש הוא המינימלי, לעומת min, במידה וכן – נהפוך אותו ל-min.
○ כעת, נגדיל את כמות האיברים בערימה ב-1 ואת כמות העצים ב-1.
○ לבסוף, נחזיר את הצומת החדש שיצרנו.
• סיבוכיות: $O(1)$.</td></tr>
<tr><td><u>deleteMin</u></td><td>• קלט: אין.
• פלט: אין.
• שיטת פעולה:
○ תחילה, נבדוק האם הרשימה שלנו ריקה.
▪ במידה וכן, נעצור את פעילות המתודה.
○ לאחר מכן, נבדוק האם המינימלי הוא גם האיבר היחיד בערימה.
▪ במידה וכן, נאפס את הערימה בעזרת המתודה destroyHeap.
○ כעת, נבדוק האם למינימלי אין ילדים.
▪ במידה ויש לו ילדים:
• נשרשר את רשימת הילדים של המינימום לרשימת השורשים בעזרת המתודה meldLists המופעלת על המינימום, וה-child שלו.
• נמחק למינימלי את המצביע ל-child שלו, על מנת למנוע כפילות איברים באותה ערימה.
○ נגדיל את כמות העצים בערימה בדרגת האיבר המינימלי, המעידה על כמות הילדים שהיו לו, זאת אומרת כמות השורשים ששרשרנו לרשימת השורשים.
○ נקטין את כמות האיברים בערימה ב-1.
○ נמחק את האיבר המינימלי בעזרת המתודה skipNode.
○ נקרא למתודה successiveLinking על מנת להוריד את כמות העצים שיש לנו בערימה.</td></tr>
</table>

timore – 308145309 – תימור איזנמן
offekgil – 308315092 – אופק גיל

	◦ לבסוף, כשכמות השורשים שלנו קטנה יותר, נמצא את האיבר המינימלי, נכון לעכשיו, בערימה בעזרת המתודה findNewMin. • סיבוכיות: $O(n)$. $amortized\big(O(logn)\big)$.
Delete	• קלט: עצם מסוג HeapNode אותו נמחק. • פלט: אין. • שיטת פעולה: ◦ תחילה נוריד את המפתח של הצומת הנדרש להיות 1-, נעשה זאת בעזרת המתודה decreaseKey. ◦ לאחר מכן, נשתמש במתודה deleteMin על מנת למחוק אותו. • סיבוכיות: $O(n)$, $amortized\big(O(logn)\big)$.
Meld	• קלט: עצם מסוג FibonacciHeap אשר מייצג את הערימה עלינו למזג לערימה הנוכחית שלנו. • פלט: אין. • שיטת פעולה: ◦ תחילה, נבדוק האם הערימה שקיבלנו היא ריקה. ▪ במידה וכן, אנו לא צריכים לעשות שום פעולה, ונעצור את המתודה. ◦ לאחר מכן, נבדוק אם הערימה שמהמופע שלה הפעלנו את המתודה היא ריקה. ▪ במידה וכן, נשים את המצביע של min בערימה שלנו להצביע על min בערימה השנייה. ◦ כעת, במידה ושתי הערימות לא ריקות נבצע שרשור של רשימת השורשים בערימה שקיבלנו לתוך רשימת השורשים בערימה שלנו בעזרת המתודה meldLists המופעלת על min של כל אחת מהערימות. ◦ לאחר מכן, נבדוק מי מהמינימליים בשתי הערמות קטן יותר ונצביע אליו עם min. ◦ לבסוף, נגדיר את כמות האיברים בערימה שלנו בכמות האיברים בערימה שמיזגנו. ◦ נגדיל את כמות העצים ברשימה שלנו בכמות העצים בערימה שמיזגנו. ◦ לפני סיום המתודה, נשמיד את הערימה השנייה בעזרת המתודה destroyHeap. • סיבוכיות: $O(1)$.
decreaseKey	• קלט: עצם מסוג HeapNode לו נוריד את הערך, ומספר אשר ייצג את הכמות אשר נוריד מהערך. • פלט: אין. • שיטת פעולה: ◦ דבר ראשון, נוריד את הערך על ה-HeapNode שקיבלנו בכמות אשר התבקשנו. ◦ כעת, נבדוק האם ה-HeapNode שקיבלנו הוא שורש, במידה וכן – עלינו לבדוק האם הוא מינימלי אחרי ההורדה ולשנות בהתאם. מכיוון שלא חותכים שורש נסיים את פעילות המתודה. ◦ לאחר מכן, נבדוק האם ההורדה שברה את יחס הסדר של ערימה, במידה ולא – נסיים את פעולת המתודה. ◦ אחרת, עלינו לבצע חיתוך לצומת שלנו בעזרת המתודה cut, ולאחר מכן, כל עוד אנו מגיעים לצומת שאיננו מסומן. ▪ את חלק זה של האלגוריתם נבצע בלולאה אשר תעלה כלפי מעלה ותעשה את הבדיקה. ▪ כל עוד יימצא צומת מסומן, נשתמש במתודה cut כדי לחתוך אותו מהעץ. ◦ נבדוק האם ה-HeapNode שהורדנו לו את הערך הוא המינימום החדש, ונחליף בהתאם. ◦ נשים לב כי שורש לא יכול להיות מסומן, על כן אם בסוף הלולאה הגענו לשורש, ניתן לסיים את המתודה. ◦ אחרת, נסמן את הצומת אליו הגענו ונעלה את כמות המסומנים ב-1. • סיבוכיות: $O(n)$, $amortized\big(O(1)\big)$.
cut	• קלט: עצם מסוג HeapNode המייצג את האיבר אותו אנחנו צריכים לחתוך ממקומו. • פלט: אין. • שיטת פעולה: ◦ תחילה, נבדוק האם ה-HeapNode שצריך לחתוך הוא מוגדר כ-child של אביו, במידה וכן – נעביר אותו לאחיו הצמוד אם קיים. אם לא מוגדר child, לא נעשה דבר. ◦ בנוסף, נגדיר ל-HeapNode שקיבלנו אבא null, מכיוון שהוא הופך להיות שורש. ◦ נשתמש במתודה skipNode על מנת להוציא את ה-HeapNode שקיבלנו מהרשימה המקושרת אליה היה שייך, ונשרשר אותו לתוך רשימת השורשים בעזרת המתודה listLinkNode עם המשתנה min. ◦ לבסוף, נעלה ב-1 את כמות העצים ואת כמות החיתוכים שביצענו. • סיבוכיות: $O(1)$.
destroyHeap	• קלט: אין. • פלט: אין. • שיטת פעולה: משימים את המצביע של min להיות null. • סיבוכיות: $O(1)$.
findNewMin	• קלט: אין. • פלט: עצם מסוג HeapNode המייצג את האיבר השני בגודלו בערימה. • שיטת פעולה: ◦ נעבור על כל רשימת השורשים החל מה-next של המינימום ועד המינימום (לא כולל אותו), בכל פעם נבדוק האם המפתח קטן מהמפתח המינימלי החדש. ▪ במידה וכן, נחליף את המצביע של המינימום החדש. ◦ לבסוף, נחזיר את האיבר השני בגודלו בערימה.

timore – 308145309 – תימור איזנמן
offekgil – 308315092 – אופק גיל

	• סיבוכיות: $O(n)$.
meldLists	• קלט: שני עצמים מסוג HeapNode המייצגים מצביעים בשתי רשימות מקושרות שונות. • פלט: אין. • שיטת פעולה: ○ נכניס את הרשימה השנייה שקיבלנו בין המצביע שקיבלנו מהרשימה הראשונה ל-next שלו. ▪ נבצע את ההשמות הנדרשות ל-next, prev על האיברים: • זה שקיבלנו וה-next שלו ברשימה הראשונה • זה שקיבלנו וה-prev שלו ברשימה השנייה. • סיבוכיות: $O(1)$.

timore – 308145309 – תימור איזנמן
offekgil – 308315092 – אופק גיל

מחלקה מקוננת – HeapNode

שדות

	Public	int key	מייצג את מפתח האיבר.
	Private	int rank	מייצג את כמות הילדים שלו, הדרגה.
	Private	boolean marked = false	מייצג האם הוא מסומן או לא.
	Private	HeapNode child	מצביע לילד שלו.
	Private	HeapNode parent	מצביע להורה שלו.
	Private	HeapNode next	מצביע לבא בתור ברשימה המקושרת.
	Private	HeapNode prev	מצביע לקודם ברשימה המקושרת.

מתודות

בנאי	• קלט: מספר המייצג את המפתח של האיבר. • פלט: אין. • שיטת פעולה: יצירת איבר חדש עם המפתח שהתקבל. • סיבוכיות: $O(1)$.
Getters	• קלט: אין. • פלט: השדה הרלוונטי לאותה המתודה. • שיטת פעולה: החזרת העצם התואם למתודה. • סיבוכיות: $O(1)$.
Setters	• קלט: עצם מהסוג הרלוונטי לאותה המתודה. • פלט: אין. • שיטת פעולה: השמת העצם החדש למשתנה המתאים למתודה. • סיבוכיות: $O(1)$.

timore – 308145309 – תימור איזנמן
offekgil – 308315092 – אופק גיל

<u>מדידות</u>:

1. Sequence 1

m	Run-Time	totalLinks	totalCuts	Potential
1000	Milliseconds – 0	0	0	1000
2000	Milliseconds – 1	0	0	2000
3000	Milliseconds – 2	0	0	3000

- ציפיות על בסיס רקע תיאורטי:
  - אנו מצפים כי זמן הריצה האסימפטוטי יהיה $O(m)$, מכיוון שכל הכנסה היא $amortized(O(1))$ ואנו מבצעים m הכנסות.
  - מכיוון ופעולות insert לא גורמות ללינקים או חיתוכים, אזי אנו מצפים כי הכמות של כל אחד מהם תהיה 0.
  - מכיוון ופונקציית הפוטנציאל היא: $\Phi = \#trees + 2 * \#marked$, וכמו שטענו קודם, מכיוון שאין מחיקות שהן לא מינימום, אזי כמות האיברים המסומנים תהיה 0, על כן אנו מצפים כי $\Phi = \#trees = m$ במקרה זה.
- תוצאות המדידות:
  - מכיוון ותוצאות זמן הריצה הן במילי שניות, אזי אנו לא מקבלים רמת דיוק מספקת, במספרים אלה, על מנת להוכיח את הטענה במלואה, אך ניתן לראות כי קיימת גדילה כמצופה.
  - כצפוי, כמות הלינקים והחיתוכים הינה 0, והפוטנציאל הוא ככמות ההכנסות m.

2. Sequence 2

m	Run-Time	totalLinks	totalCuts	Potential
1000	Milliseconds – 7	2461	0	6
2000	Milliseconds – 8	5350	0	6
3000	Milliseconds – 9	7803	0	7

- ציפיות על בסיס רקע תיאורטי:
  - אנו מצפים כי זמן הריצה האסימפטוטי יהיה $O(mlogm)$, מכיוון שכל הכנסה היא $amortized(O(1))$ ואנו מבצעים m הכנסות, לאחר מכן אנו מבצעים $\frac{m}{2}$ מחיקות מינימום, שכל פעולה כזו היא בסיבוכיות של $amortized(O(logm))$, ועל כן נקבל כי סה"כ סיבוכיות הריצה תהיה: $O\left(m + \frac{m}{2}logm\right) = O(mlogm)$.
  - מכיוון ופעולות insert וdelete-min לא גורמות חיתוכים, אזי אנו מצפים כי הכמות תהיה 0.
  - מכיוון ורק פעולת delete-min תגרום ללינקים, אנו מבצעים $\frac{m}{2}$ פעולות כאלה, ופעולה זו מתבצעת בסיבוכיות $amortized(O(logm))$, אזי אנו מצפים כי כמות הלינקים תהיה $O(mlogm)$.
  - מכיוון ופונקציית הפוטנציאל היא: $\Phi = \#trees + 2 * \#marked$, וכמו שטענו קודם, מכיוון שאין מחיקות שהן לא מינימום, אזי כמות האיברים המסומנים תהיה 0. בנוסף, לאחר כל פעולת delete-min קיים עד עץ אחד מכל דרגה, עד לדרגה $\log_\phi n$ (כמות האיברים בהתאם למחיקה), על כן אנו מצפים כי הפוטנציאל יהיה ככמות ה-1 בייצוג הבינארי של $\frac{m}{2}$, מכיוון שלאחר כל המחיקות ניוותר עם כמות כזאת של עצים. $\Phi = \#trees = \#of\ 1\ in\ binary\ rep$ במקרה זה.
- תוצאות המדידות:
  - מכיוון ותוצאות זמן הריצה הן במילי שניות, אזי אנו לא מקבלים רמת דיוק מספקת, במספרים אלה, על מנת להוכיח את הטענה במלואה, אך ניתן לראות כי קיימת גדילה כמצופה.

timore – 308145309 – תימור איזנמן
offekgil – 308315092 – אופק גיל

- כצפוי, כמות החיתוכים היא 0.
- כמות הלינקים בסדר גודל $mlogm$, והגדילה בערך לינארית.
- הפוטנציאל הוא מספר ה-1 בייצוג הבינארי של $\frac{m}{2}$:
  - 500 – 111110100 = 6 1-ים.
  - 1000 - 1111101000 = 6 1-ים.
  - 1500 – 10111011100 = 7 1-ים.