**תיעוד וסיבוכיות המתודות – תרגיל מעשי 2**

**מגישים:**

|  |  |  |
| --- | --- | --- |
|  | **מגיש א'** | **מגיש ב'** |
| **שם פרטי** | אדם | אסף |
| **שם משפחה** | טובי | מיכאלוביץ' |
| **תעודת זהות** | 215334822 | 200637270 |
| **שם משתמש אוניברסיטאי** | adamtuby | michaelovits |

**סיבוכיות ותיעוד:**

**נוטציה**

הנוטציה בתרגיל זה היא:

* הוא מספר הצמתים במצב ההתחלתי של הגרף (לאחר ה-constructor)
* הוא מספר הצמתים בגרף כרגע.
* הוא מספר הקשתות בגרף כרגע.

**public** **class** Graph

**----------------------------------------------------------------------------------------------------------**חשוב לציין, שלשם השלמות של המימוש והכלליות, מימשנו את המחלקות hashMap<V>, maxheap<T>, DoublyLinkedList<T>, Cell<T>, hashCell<T>, heapNode<V> באופן גנרי. ועל כן, ניתן לראות בחלק משלבי המימוש את השורה @SuppressWarnings("unchecked") שהוספנו בכדי לעקוף אזהרות לא רצויות לגבי גנריות.  
ההחלטה לעשות מתודות אלו גנריות, היא שכן רצינו שכל מימוש של כל מבנה נתונים יהיה "בפני עצמו", ולכן עשינו אותם סטטיים שניתנים למימוש מחוץ למחלקה.   
נוסף על כך, **סיבוכיות כל מתודת getter/setter היא O(1), אלא אם כן צוין אחרת.**

**private** **final** hashMap nodesHash - שדה זה מחזיק את טבלת ההאש המבוקשת בהנחיות התרגיל, שבה נשמור את הצמתים בגרף, כאשר המפתחות של האיברים בטבלת ההאש יהיו ה'מזההים' שלהם. כלומר הטבלה ממפה ממזהה קודקוד לייצוגו בגרף ובערימת המקסימום.

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**private** **final** maxHeap nodesHeap - שדה זה מחזיק את ערימת המקסימום המבוקשת בהנחיות התרגיל, בה כל זוג מפתח-ערך הוא כך שהמפתח הוא סכום הסביבה של צומת כלשהו u, והערך הוא הצומת u.

------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**public** **boolean** isEmpty()

מתודה זו תחזיר true אם בגרף אין צמתים כלל, אחרת false. היא עושה זאת באמצעות פנייה לגודל ערימת המקסימום nodesHeap, ובודקת האם היא 0. אם היא אכן 0, זה אומר שלא קיימים צמתים בכלל בעץ ולכן תחזיר true, אחרת תחזיר false. נשתמש במתודה זו בכדי לחסוך שכפול קוד במתודות אחרות רבות, שמצריכות התייחסות למקרה הפרטי שהגרף ריק.

**סיבוכיות המתודה:** O(1)  
ובכן, התבקשנו לבצע מספר קבוע של פעולות השוואה וגישה למצביעים ולכן הסיבוכיות היא קבועה.

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**public** Graph(Node [] nodes)

הבנאי של המחלקה של התרגיל – Graph.  
הבנאי מקבל רשימה של צמתים (הצמתים הם בפורמט של צומת של גרף. פורמט זה הוא מחלקה סטטית Node של המחלקה Graph). הוא מגדיר את השדה nodesHash בכך שהוא שולח את מספר הצמתים שנשלחו, לבנאי של hashMap, שהיא מחלקה שמימשנו בתוך Graph, שממלאת את דרישת התרגיל שתהיה טבלת האש לגרף.  
הבנאי של hashMap לוקח את מספר הצמתים, ולוקח בהכרח את המספר כפקטור העומס הרצוי, ובונה לו את המערך לפי הצרכים שלו, כפי שראינו בכיתה. פירוט מעמיק על המחלקה בהמשך.  
נוסף על כך, הוא מגדיר את השדה nodesHeap בכך שהוא שולח גם את מספר הצמתים שנשלחו, לבנאי של maxHeap (שממש את ערימת המקסימום שבדרישת התרגיל) בשביל ש maxHeap יוכל לאתחל את ה priority-Queue שלו (כמערך). פירוט מעמיק על המחלקה maxHeap יבוא בהמשך.

לאחר אתחול השדות הללו, הוא מוסיף באמצעות לולאה את כל הצמתים nodes, אחד אחד, לטבלת ההאש nodesHash, ולערימת המקסימום nodesHeap, לפי הדרישות של המחלקות.

**סיבוכיות המתודה:** O(N) בתוחלתסיבוכיות המתודה היא בתוחלת , כאשר n הוא מספר הצמתים שהכנסנו לגרף.  
ובכן, יצירת hashMap עם m=N (מספר התאים בטבלת ה chaining), הוא מסיבוכיות O(N) w.c..  
וכן, הכנסה של n מפתחות לערימת מקסימום (או מינימום), כמו שראינו בכיתה, בעלת סיבוכיות של O(n) w.c., וסיבוכיות ההכנסה של n מפתחות לטבלת האש לוקח O(n) EXPECTED בטבלת האש של Linked Lists (לפי מה שראינו בכיתה, כי דאגנו שפקטור העומס יהיה ), ולכן סה"כ סיבוכיות המתודה היא: O(n) EXPECTED + O(n) = O(n) EXPECTED.

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**public** Node maxNeighborhoodWeight()

מתודה זו מחזירה את הצומת בגרף, שמשקל הסביבה שלו הוא הגדול ביותר בגרף. משקל הסביבה הוא סכום המשקלים של השכנים של הצומת + ועוד המשקל של הצומת עצמו. מתודה זו עושה זאת באמצעות החזרת הצומת שנמצא בראש ערימת המקסימום שאנחנו מתחזקים nodesHeap.   
במידה ובגרף אין צמתים, המתודה תחזיר null.

**סיבוכיות המתודה:** O(1)סיבוכיות המתודה היא O(1), שכן כל מה שנדרש זה לשלוף את האיבר הראשון במערך שאנחנו מתחזקים כ priority Queue של הערימה.

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**public** **int** getNeighborhoodWeight(**int** node\_id)

מתודה זו מחזירה את משקל הסביבה של הצומת בגרף שבעל המזהה שמקבלת המתודה – node\_id.  
במידה ולא קיים צומת בגרף עם מזהה שכזה, נחזיר -1.  
נחפש את המזהה node\_id בטבלת ההאש, ובמידה ונמצא צומת שאכן עונה למזהה, נחזיר את משקל הסביבה שלו, אחרת נחזיר -1.  
נעשה זאת פשוט על ידי תחזוק שדה של משקל הסביבה עבור כל צומת (הרי במילא אנחנו צריכים לתחזק שדה זה עבור שלמות ערימת המקסימום שאנו צריכים לממש).  
ובכן, הגישה למשקל הסביבה תיקח לנו סיבוכיות של O(1), ולכן זניחה.

**סיבוכיות המתודה:** O(1) בתוחלתסיבוכיות המתודה היא O(1) EXPECTED, שכן החיפוש של המזהה node\_id בטבלת האש במימוש Linked List, הינו מסיבוכיות O(1) EXPECTED (נקבע לפי זמן החיפוש בתוחלת שעלינו לעבור).

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**public** **int** getNumNodes()

מתודה זו מחזירה את כמות הצמתים שקיימת בבגרף ברגע הקריאה למתודה.   
נעשה זאת באמצעות השדה size שאנחנו מתחזקים בערימת המקסימום nodesHeap שלנו, שבעצם סופר כמה צמתים קיימים בערימה. תיעוד של שדה זה ותחזוקו בהמשך התיעוד של מחלקת הערימת מקסימום.  
ובכן, פשוט נקרא ל getter של השדה size של nodesHeap, ונחזיר את התוצאה.

**סיבוכיות המתודה:** O(1)סיבוכויות המתודה היא כמובן O(1) שכן פשוט ניגשנו למצביעים.

------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**public** **int** getNumEdges()

מתודה זו מחזירה את כמות הקשתות הקיימות בגרף שלנו ברגע הקריאה למתודה.  
דבר זה נעשה בעצם באמצעות לולאה על כל הצמתים שנמצאים בגרף. ניתן לעשות את לולאה זאת על המערך של טבלת ההאש שמימשנו, או על המערך של ערימת המקסימום שמימשנו (שכן שני המימושים מכילים אפשרות לעבור אופן איטראטיבי על כל הצמתים בגרף).  
בחרנו בערימת המקסימום.  
ובכן, ניגש למערך של ערימת המסימום המייצג אותה, ונעבור צומת צומת, ונוסיף את כמות השכנים שיש לכל צומת לאיזשהו משתנה result אינטג'רי.  
ניתן לבדוק את כמות השכנים שיש לכל צומת על ידי קריאה לשדה length שיש לרשימה מקושרת דו-כיוונית שמימשנו.  
כעת, לאחר שהוספנו את כל כמויות השכנים ל result, בעצם נקבל את כמות הקשתות בגרף כפול 2, שכן כל קשת מיוצגת **בדיוק** פעמיים ברשימות השכנויות של הצמתים, ועל פי משפט ממבוא לקומבינטוריקה שאומר ש .  
ולכן, נחזיר את result/2.

**סיבוכיות המתודה:** O(n) w.c.  
סיבוכיות המתודה היא כמובן O(n) w.c., כאשר n הוא כמות הצמתים בגרף, שכן ביצענו n איטראציות כאשר כל אחת לקחה O(1).

**public** **boolean** deleteNode(**int** node\_id)

מתודה זו נועדה למחוק את הצומת בגרף שהמזהה שלו הוא node\_id.  
במידה ואין צומת כזה בגרף, תחזיר false, אחרת, תמחק את כל הקשתות המחוברות אל הצומת, ותשנה את כל המשקלים הסביבתיים של השכנים היו לצומת, ותחזיר true.

ובכן, נמצא את הצומת בטבלת ההאש לפי המזהה שקיבלנו.  
ראשית נמחק את הצומת לגמרי מטבלת ההאש (O(1) EXPECTED) ונמחק את הצומת מערימת המקסימום (O(log n)).  
כעת, ניגש לרשימת השכנויות של הצומת.   
נעבור בלולאה על כל האיברים שנמצאים ברשימת השכנויות ונעשה כך עבור כל איבר:  
- עבור כל איבר, שמייצג קשת, ניגש לייצוג המקביל של הקשת באמצעות השדה parallel, שיהיה בעצם איבר בתוך רשימת השכנויות של השכן שנמצא בקשת זו ספציפית – O(1)  
- האיבר שקיבלנו בתוך רשימת השכנויות של השכן שנמצא בקשת זו ספציפית, יהיה בעצם הצומת שאנו רוצים למחוק, בתוך רשימת השכנויות של השכן שלו בקשת  
- נמחק את איבר זה מן רשימת השכנויות של השכן הנ"ל, באמצעות מחיקה נורמטיבית מרשימה דו-כיוונית מקושרת – O(1)  
- נעדכן את המשקל הסביבתי של השכן הנ"ל, בכך שנחסיר מהמשקל הסביבתי של השכן את המשקל של הצומת שאנו רוצים למחוק - O(1)  
- נבצע פעולת Heapify לשכן הנ"ל בערימת המקסימום, שכן המשקל הסביבתי שלו השתנה - O(log (n-1))=O(log n)  
  
**סיבוכיות המתודה:** , כאשר היא דרגתו של הצומת  
סך הכל, כאשר היא דרגתו של הצומת שאותו אנחנו רוצים למחוק מן הגרף, סיבוכיות המתודה היא:

**public** **boolean** addEdge(**int** node1\_id, **int** node2\_id)

מתודה זו נועדה להוסיף קשתות חדשות לגרף.  
המתודה מחזירה true אם הוספנו קשת בין הצמתים שמחזיקים את המזההים שקיבלנו, אחרת false (תחזיר false במידה ואחד מהמזההים שקיבלנו לא נמצא בגרף).  
אומנם אין תת מחלקה רשמית של Graph שמייצגת 'קשת' כמו שאר האובייקטים שייצרנו, אך ייצוג הקשת הינו דווקא בתוך מימוש המחלקה Node, שהיא מחלקה המממשת צומת בגרף.  
ובכן, נסביר בקצרה על הייצוג של הקשת שבחרנו בו, ונשאיר את פרטי הפרטים לתיעוד המחלקה Node.  
אכן, כל צומת מכיל בתוכו שדה שהינו תת-מחלקה של Graph שמימשנו ששמה DoublyLinkedList<T>, שהיא בעצם מימוש של רשימה מקושרת דו-כיוונית שהאיברים שלה הם ממחלקה T (נתעד אותה בהמשך), אך במקרה זה T יהיה Node, שכשמה כן הוא, רשימה מקושרת דו-כיוונית של איברים מסוג Node.  
אחריות רשימה זו, הוא להכיל את השכנים של הצומת (לפי סדר הכנסתם).  
האיברים ברשימה המקושרת הדו-כיוונית ממומשים על ידי תת המחלקה של DoublyLinkedList<T>, הנקראת: DoublyLinkedCell (תת-מחלקה לא סטטית, שכן נקבעת על פי T, ושכן אנו צריכים להשתמש בDoublyLinkedCell.this בכדי למחוק איברים בעתיד מרשימות שכנויות).  
אז קשת בין הצומת , תיוצג על ידי הנוכחות של הצומת ברשימת השכנויות של , ובדומה תיוצג על ידי הנוכחות של הצומת ברשימת השכנויות של .  
עם זאת, כיוון שנדרשנו שלכל קשת יהיה מצביע דו-כיווני לייצוג ההפוך שלה (כפי שראינו לקשת יש שני ייצוגים), לכל איבר ברשימה המקושרת של , שמייצג קשת יהיה שדה parallel, המכיל מצביע לאיבר אחר מסוג DoublyLinkedCell, שיהיה בעצם האיבר ברשימה המקושרת של , כלומר ייצוג של הקשת .   
  
ובכן, נעשה זאת באמצעות חיפוש המזההים בטבלת ההאש nodesHash של הצמתים, ובמידה ומצאנו שני צמתים שתואמים את המזההים (אחרת מחזירים false), נוסיף את שני הצמתים לרשימת השכנויות של אחד השנייה, ונדאג להוסיף את שדה ה parallel לאיברים החדשים ברשימת השכנויות.  
  
לאחר ההוספה של הקשת וההוספה של הצמתים לרשימת השכנויות של אחד השני, המשקל הסביבתי של שני הצמתים כנראה השתנה, ועל כן יש חשש ל2 פעולות Heapify שיתבצעו בערימה. כל פעולה כזאת תעלה לנו כפי שראינו בכיתה.

**סיבוכיות המתודה:**  O(log n)סיבוכיות המתודה היא O(log n), שכן התבקשנו למצוא 2 איברים עם מפתחות בטבלת האש, ובנוסף על כך לבצע פעולות בעלות סיבוכיות זמן קבועה כגון שינויי מצביעים וכו'.

**public** **static** **class** Node

**----------------------------------------------------------------------------------------------------------מחלקה המממשת צומת של גרף Graph  
----------------------------------------------------------------------------------------------------------**

**private** **final** **int** id - שדה זה מחזיק את המזהה של הצומת בגרף   
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**private** **final** **int** weight - שדה זה מחזיק את המשקל של הצומת בגרף  
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**private** **int** vicinityWeight - שדה זה מחזיק את המשקל הסביבתי של הצומת בגרף  
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**private** **final** DoublyLinkedList<Node> Neighbours - שדה זה מחזיק מצביע אל רשימת השכנויות של הצומת בגרף  
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**private** maxHeap.heapNode heapForm - שדה זה מחזיק מצביע אל הייצוג של הצומת בערימת המקסימום של הגרף. שדה זה יאותחל ברגע שנכניס את הצומת לערימת המקסימום בבנאי של Graph. ובכן, מתודת ההכנסה של ערימת מקסימום, מחזירה מצביע למופע של הצומת בערימת המקסימום, ולכן, כאשר נוסיף צומת לערימת המקסימום, ניקח את הערך המוחזר, ונאתחל את heapForm להיות הערך המוחזר, ע"י המתודה setHashForm אותה נתעד בקרוב.  
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**private** hashMap.hashCell hashForm - שדה זה מחזיק מצביע אל הייצוג של הצומת בטבלת ההאש של הגרף. שדה זה יאותחל ברגע שנכניס את הצומת לטבלת ההאש בבנאי של Graph.  
ובכן, מתודת ההכנסה של טבלת האש, מחזירה מצביע למופע של הצומת בטבלת ההאש, ולכן, כאשר נוסיף צומת לטבלת ההאש, ניקח את הערך המוחזר, ונאתחל את hashForm להיות הערך המוחזר, ע"י המתודה setHashForm אותה נתעד בקרוב.

------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**public** Node(**int** id, **int** weight)

הבנאי של הצומת.  
הוא מאתחל את המזהה של הצומת להיות id, ומאתחל את המשקל של הצומת להיות weight, ומאתחל את המשקל הסביבתי של הצומת להיות 0, ומאתחל את רשימת השכנויות של הצומת להיות new DoublyLinkedList<Node>.

**סיבוכיות המתודה:** O(1)

------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**public** **int** getId() - ה getter של השדה id. מחזיר את המזהה של הצומת.  
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**public** **int** getWeight() - ה getter של השדה weight. מחזיר את המשקל של הצומת.  
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**public** **int** getVicinityWeight()- ה getter של השדה vicinityWeight. מחזיר את המשקל הסביבתי של הצומת.

**public** **void** UpdateVicinityWeight(**int** additionalWeight)

מתודה זו מעדכנת את המשקל הסביבתי של הצומת במידה ונוסף לו שכן.  
היא לוקחת את המשקל הסביבתי הנוכחי, ומוסיפה לו את המשקל additionalWeight (שאמור להיות המשקל של השכן שנוסף לצומת).  
בגלל שהמשקל הסביבתי השתנה, כעת יש אפשרות שצריך לבצע פעולת Heapify בערימת המקסימום, ואכן נעשה זאת.

**סיבוכיות המתודה:** O(log n)ובכן, התבקשנו לבצע פעולת Heapify, ועל כן סיבוכיות הפעולה תהיה O(log n), שכן יש n צמתים בערימת המקסימום.

------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**public** **void** addNeighbour(Node node)

מתודה זו מוסיפה את הצומת node להיות שכן של הצומת שלנו בגרף.  
מתודה זו נקראת מתוך מתודת הוספת קשת בגרף, כאשר הקשת מחברת בין node לבין הצומת של מופע המחלקה.  
אנו נוסיף את node לרשימת השכנויות של הצומת, ונוסיף למשקל הסביבתי של הצומת את המשקל של node (באמצעות UpdateVicinityWeight(node.getWeight())).

**סיבוכיות המתודה:** O(log n)  
ובכן, ההוספה של השכן לרשימת השכנויות הוא פשוט הוספה של איבר לסוף הרשימה המקושרת הדו-כיוונית, ועל כן פעולה זו בעלת סיבוכיות קבועה.  
עם זאת, עשינו גם שינוי במשקל הסביבתי, באמצעות המתודה UpdateVicinityWeight, וסיבוכיות מתודה זו היא O(log n), ולכן סיבוכיות המתודה addNeighbour תהיה O(1)+O(log n) = O(log n).

------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**public** **void** setHeapForm(maxHeap.heapNode heapForm)

מתודה נקראת פעם אחת בלבד במהלך כל המופע של Graph.  
תפקיד מתודה זו הוא לאתחל את המצביע heapForm של הצומת, ככה שנוכל לגשת באמצעות המופע של Node בלבד, למופעים של הצומת בתוך ערימת המקסימום ובתוך טבלת ההאש לשם שלמות המימוש ונגישות המימוש. ובכן, נאתחל את **this**.heapForm, , להיות heapForm שקיבלנו כארגומנט.  
מתודה זו נקראת רק מתוך הבנאי של המחלקה Graph, בה אנו מקבלים N צמתים ומוסיפים אותם לטבלת ההאש nodesHash ולערימת המקסימום nodesHeap.

**סיבוכיות המתודה:** O(1)  
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**public** **void** setHashForm(hashMap<Node>.hashCell<Node> hashForm)

מתודה נקראת פעם אחת בלבד במהלך כל המופע של Graph.  
תפקיד מתודה זו הוא לאתחל את המצביע hashForm של הצומת, ככה שנוכל לגשת באמצעות המופע של Node בלבד, למופעים של הצומת בתוך ערימת המקסימום ובתוך טבלת ההאש לשם שלמות המימוש ונגישות המימוש. ובכן, נאתחל את **this**.hashForm, , להיות hashForm שקיבלנו כארגומנט.  
מתודה זו נקראת רק מתוך הבנאי של המחלקה Graph, בה אנו מקבלים N צמתים ומוסיפים אותם לטבלת ההאש nodesHash ולערימת המקסימום nodesHeap.

**סיבוכיות המתודה:** O(1)

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**public** hashMap<Node>.hashCell<Node> getHashForm()

ה getter של השדה hashForm.

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**public** maxHeap<Node>.heapNode<Node> getHeapForm()

ה getter של השדה heapForm.

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**public** **int** getNeighboursAmount()

מתודה זו מחזירה את כמות השכנים שיש לצומת. ובכן, אנו עושים זאת ע"י גישה לרשימת השכנויות של הצומת מהשדה של הצומת: Neighbours, וניגשים לשדה length של Neighbours, ע"י המתודה Neighbours.length(), שמחזירה את מספר האיברים ברשימה - שהוא גם מספר השכנים של הצומת, ועל כן נחזיר את זה.

**סיבוכיות המתודה:** O(1)

**public** **static** **abstract** **class** Cell<T>

**----------------------------------------------------------------------------------------------------------  
מחלקה המממשת תא כלשהו שעוטף ערך כלשהו מהמחלקה T, ומפתח כלשהו key  
מחלקה זה מהווה מחלקה אבסטרקטית שחוסכת שכפול קוד עבור תא כלשהו בטבלת hash, ועבור צומת כלשהו בערימת מקסימום.  
----------------------------------------------------------------------------------------------------------**

**protected** **int** key – שדה המחזיק את המפתח של התא.

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**protected** T value – שדה המחזיק את הערך אותו עוטף התא. הערך הוא מופע של המחלקה T.

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**protected** Cell(**int** key, T value)

בנאי המחלקה.   
הוא מאתחל את השדה key להיות הארגומנט שקיבלנו key, ומאתחל את השדה value להיות הארגומנט שקיבלנו value.

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**protected** **int** getKey() – ה getter של השדה key.

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**protected** T getValue() – ה getter של השדה value.

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**protected** **void** setKey(**int** key) – ה setter של השדה key.

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**protected** **void** setValue(T value) – ה setter של השדה value.

**public** **static** **class** DoublyLinkedList<T>

**------------------------------------------------------------------------------------------------------  
רשימה מקושרת דו-כיוונית, כאשר הערכים שבתוך התאים של הרשימה הם מהמחלקה הגנרית T  
------------------------------------------------------------------------------------------------------**

**private** DoublyLinkedCell tail - המצביע לזנב של הרשימה

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**private** DoublyLinkedCell head – המצביע לתחילת הרשימה

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**private** **int** length = 0 – מספר האיברים ברשימה שמאותחל להיות 0

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**public** **int** length() – ה getter של מספר האיברים ברשימה.

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**private** **boolean** isEmpty()

בודק אם הרשימה ריקה, כלומר אם אין איברים רשימה.   
נשתמש במתודה זו במתודת הוספת איבר לרשימה, שכן כאשר הרשימה ריקה, במידה ואנו רוצים להוסיף את האיבר item לרשימה, יש לאתחל גם את זנב הרשימה להיות item, וגם את ראש הרשימה (תחילת הרשימה) להיות item, בעוד שבמידה והרשימה אינה ריקה, אנו מאתחלים רק את הזנב להיות item (שכן אנחנו מבצעים רק הכנסות לסוף הרשימה במימוש זה).

**סיבוכיות המתודה:** O(1)  
שכן אנו מבצעים מספר קבוע של השוואות וגישות לשדה length.

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**public** **void** addItem(T item)

מתודה זו נועדה להוספת איבר לסוף הרשימה הנתונה.   
היא עושה זאת באמצעות יצירת תא חדש של רשימה. תא זה ממומש על ידי תת-המחלקה DoublyLinkedCell, שמתחזק את הדרישות של תא בתוך רשימה מקושרת דו-כיוונית.  
מתודה זו יוצרת מופע חדש של תא DoublyLinkedCell, שהערך שלו יהיה item.  
שלבי פעולות המתודה:  
- העלאה של length באחד  
- במידה והרשימה ריקה:  
1. אתחול של הזנב והראש של הרשימה להיות התא החדש  
2. אתחול השדות next, prev של התא החדש להיות התא עצמו  
- במידה והרשימה אינה ריקה:  
1. נהפוך את התא החדש להיות הזנב החדש של הרשימה  
2. אתחול השדה prev של התא החדש להיות הזנב הקודם שהיה לרשימה, והשדה next של הזנב הקודם שהיה לרשימה להיות התא החדש  
3. אתחול השדה next של התא החדש להיות הראש של הרשימה, והשדה prev של ראש הרשימה להיות התא החדש

**סיבוכיות המתודה:** O(1)  
אנו מבצעים מספר קבוע של החלפות מצביעים ויצירת מופעים של המחלקה DoublyLinkedCell (שלוקח בסה"כ O(1)).

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**public** **void** deleteCell(DoublyLinkedCell item)

מתודה זו נועדה למחוק תא מהרשימה.  
היא מקבלת כארגומנט את התא הספציפי שברצוננו למחוק, ומבצעת פעולות החלפת מצביעים כפי שראינו בכיתה, בכדי למחוק את התא מן הרשימה.  
בנוסף לזאת, היא מורידה את הערך length באחד.  
יש התייחסות פרטית למקרה בו התא הוא ראש הרשימה, זנב הרשימה, והמקרה בו הוא לא הראש ולא הזנב של הרשימה.

**סיבוכיות המתודה:** O(1)  
ובכן, כל מה שביצענו הוא מספר קבוע של פעולות של החלפות מצביעים בשביל למחוק את התא מן הרשימה, ועל כן, סיבוכיות המתודה היא O(1).

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**private** **class** DoublyLinkedCell

**----------------------------------------------------------------------------------------------------------מחלקה המממשת תא ברשימה מקושרת דו-כיוונית, כאשר הערך בתא יהיה מופע של המחלקה T  
----------------------------------------------------------------------------------------------------------**

**private** DoublyLinkedCell next - התא שבא אחרי התא של המופע.

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**private** DoublyLinkedCell prev - התא שבא לפני התא של המופע.

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**private** DoublyLinkedCell parallel - התא המקביל של התא של המופע. לתא המקביל אין הגדרה ספציפית, אלא הוא לשימוש הלקוח בלבד.

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**private** **final** T item - הערך שאותו התא עוטף.

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**public** DoublyLinkedCell(T item)

הבנאי של המחלקה. הבנאי מקבל מופע כלשהו של המחלקה T (T מוגדר באופן גנרי ביצירת מופע הרשימה), ועוטף אותו באמצעות המחלקה.  
כלומר, הבנאי מאתחל **this**.item = item.

**סיבוכיות המתודה:** O(1)

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**public** **void** setParallel(DoublyLinkedCell parallel) - ה setter של השדה parallel.

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**public** DoublyLinkedCell getParallel() - ה getter של השדה parallel.

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**public** DoublyLinkedList<T> getRepresentativeList() - ה getter של הרשימה שמכילה את התא.

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**public** T getItem() - ה getter של הערך אותו התא עוטף

**public** **static** **class** hashMap<V>

**----------------------------------------------------------------------------------------------------------  
מחלקה המממשת טבלת האש Hash-Map, בצורת מימוש של chaining, עם universal hashing עם משפחת פונקציות המודולו, כאשר הראשוני הרלוונטי הוא , כאשר הערכים אותם מחזיק כל תא בטבלת ה Hash-Map, הם מופעים של המחלקה הגנרית V  
----------------------------------------------------------------------------------------------------------**

**private** **final** **int** p – המספר הראשוני שאיתו אנחנו נייצר פונקציית מודולו שלו: hash

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**private** **final** hashCell<V>[] table – המערך שבו נתחזק את ה chaining של הטבלה

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**private** **final** **int** a – הa במשוואה של יצירת פונקציית המודולו שלנו: hash

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**private** **final** **int** b – הb במשוואה של יצירת פונקציית המודולו שלנו: hash

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**private** **final** **int** m – מספר התאים בטבלה table שלנו שבא נחזיק את ה chaining שלנו. מספר התאים בטבלה הוא כמספר האיברים שהכנסנו בבנאי של המחלקה לתוך טבלת ההאש כפול   
כאשר הוא פקטור העומס שקיבלנו בבנאי של המחלקה. בתרגיל זה, יהיה בהכרח , ועל כן מספר התאים בטבלה, יהיה תמיד 2 כפול מספר האיברים שהכנסנו בבנאי של המחלקה.  
כיוון שאנחנו לא יכולים להכניס איברים חדשים לטבלת ההאש לאחר אתחול המשתנה nodesHash שבבנאי המחלקה Graph, אין טעם להגדיל/להקטין את m (כלומר את גודל המערך table) לאורך המימוש.

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**private** **int** size = 0 – מספר האיברים שנמצאים בטבלת ההאש. שדה זה מאותחל להיות 0.

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**public** hashMap(**int** m, **float** loadFactor)

בנאי המחלקה של טבלת ההאש.  
אנו מקבלים את m שהוא (לכל היותר) מספר האיברים שאנחנו הולכים להכניס לטבלה בסך הכל.  
אנחנו מכפילים את m ב , והתוצאה תהיה הערך שנאתחל את השדה **this**.m להיות.  
כעת, נאתחל את   
זה בעצם יתן לנו פקטור עומס כפי שניתן לנו כארגומנט loadFactor.  
נאתחל את p הראשוני שבעצם נשתמש בו בשביל להשלים את הדרישה לקיום משפחת פונקצייות מודולו של universal hashing, להיות כפי שנכתב בהנחיות התרגיל: .  
נוסף על כך, נאתחל באופן ראנדומלי את הערכים a ו b, שאנו נשתמש בהם בשביל לייצר את פונקציית המודולו hash שלנו באמצעות universal hashing.   
ובכן, לפי הדרישות של פונקציית מודולו ע"פ universal hashing, נייצר a ו b ראנדומלים העונים על הדרישות: .

**סיבוכיות המתודה:** O(m)  
ובכן, הפעולה היחידה במתודה זו, שאינה בעלת סיבוכיות קבועה, היא יצירת המערך שאנו מתחזקים שינהל את ה chaining. מערך זה מאותחל להיות באורך 2m=O(m), ועל כן סיבוכיות יצירתו היא O(m), וכך גם סיבוכיות המתודה היא O(m).

**private** **int** hash(**int** key)

מתודה זו היא פונקציית ה hash של הטבלת האש. זו פונקציית מודולו ממשפחת הפונקציות שראינו בכיתה, שמתאימה לuniversal hashing. ובכן, לפי מה שראינו בכיתה, כיוון שכבר איתחלנו a,b ראנדומליים, אז פונקציית המודולו שלנו נבחרה באופן ראנדומלי כרצוי, ועבור מפתח כלשהו key, היא תחזיר: .

**סיבוכיות המתודה:** O(1)  
סיבוכיות המתודה היא פעולות אריתמטיות בלבד, ולכן כיוון שלא צוין שאנו צריכים להתחשב בהם, הסיבוכיות היא קבועה, כלומר O(1).

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**public** V get(**int** key)

מתודה זו מקבלת מפתח כלשהו key, וצריכה להחזיר את הערך שבתא בטבלת ההאש, שמחזיק את המפתח key. במידה ואין כזה תא בטבלה שמחזיק את המפתח, תחזיר null.  
אותו ערך הוא מופע של המחלקה V שהגדרנו גנרית ביצירת המופע של המחלקה hashMap<V>.  
ובכן, אנחנו מוצאים את ה hash של המפתח key, וניגשים לתא ה hash(key) שבמערך table של טבלת ההאש שלנו.  
בתא הזה, קיימת רשימה מקושרת חד-כיוונית של כל האיברים שנכנסו עם מפתחות ש hash של המפתחות שלהם היה hash(key). אזי, נעשה חיפוש נורמטיבי ברשימה המקושרת עד שנגיע לתא ברשימה המקושרת שמחזיק את המפתח המדוייק key שלנו שאנו מחפשים (קיים רק אחד כזה, שכן המפתח המדוייק הזה זה גם המזהה של הצומת Node, וידוע לנו שכל הצמתים בעלי מזההים שונים).  
ברגע שנמצא תא שמחזיק במפתח הרצוי, נחזיר את הערך שנמצא בתוך התא.

**סיבוכיות המתודה:** בתוחלת  
ידוע לנו שחיפוש בטבלת hash עם chaining, כאשר פקטור העומס הוא , במידה והחיפוש לא מוצלח (מקרה זה גרוע יותר מחיפוש מוצלח), הוא בתוחלת:

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**public** hashCell<V> addItem(**int** key, V item)

מתודה זו מוסיפה לטבלת ההאש את האיבר item יחד עם המפתח key. נעלה את המשתנה size ב1.  
נסביר את התהליך בקצרה:  
ניצור מופע חדש של תת-המחלקה hashCell<V> שיעטוף את item.  
נקרא למופע זה itemCell.  
לבנאי של itemCell, יישלח המפתח key, הערך item, וnull שיהיה מופע של hashCell<V>.  
key יאותחל להיות המפתח של itemCell, item יאותחל להיות הערך של itemCell, וnull יאותחל להיות הnext של itemCell (כלומר התא שבא אחרי itemCell, ברשימה המקושרת שtable[hash(key)] מכיל).  
מחשבים את hash(key), ופונים לתא ה hash(key) במערך table של טבלת ההאש.  
אם התא ה hash(key) הזה, הוא ריק, אז אנחנו מאתחלים את table[hash(key)] להיות itemCell, אחרת, אנו פונים לאיבר שנמצא ב table[hash(key)], ובאופן איטראטיבי ממשיכים עד לסוף הרשימה המקושרת של תאים hashCell<V> שנמצאת ב table[hash(key)].  
כשנגיע לסוף הרשימה המקושרת, נגדיר את השדה next של האיבר האחרון ברשימה להיות itemCell.  
המתודה מחזירה את המופע של האיבר שהכנסנו, לשם אתחול השדה hashForm של צומת Node.

**סיבוכיות המתודה:** O(1) בתוחלת  
ובכן, סיבוכיות הכנסה של איבר חדש לטבלת האש הוא כסיבוכיות חיפוש לא מוצלח של איבר כלשהו בטבלת האש, ועל כן, הסיבוכיות לפי מה שראינו בכיתה היא בתוחלת:

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**public** **int** removeNode(**int** key)

מתודה זו מחפשת איבר בטבלת ההאש שבעל המפתח key, ובמידה וקיים כזה, מוחקת אותו, אחרת, מחזירה -1.  
ובכן, התהליך דומה לחיפוש של איבר בטבלת האש.  
אנחנו מחשבים את hash(key), ופונים לתא ה hash(key) במערך table.  
נאתחל משתנה hashCell<V> curr להיות table[hash(key)], ונאתחל משתנה hashCell<V> prev שתמיד יהיה שווה לאיבר שבא לפני curr ברשימה המקושרת (גם אם נשנה את curr).  
ובכן, נתחיל לולאה על האיברים ברשימה המקושרת table[hash(key)], עד שנגיע לאיבר כלשהו ברשימה שמחזיק את המפתח key. במידה ועברנו על כל הרשימה ולא מצאנו איבר כזה, נחזיר -1.  
במידה ועברנו על הרשימה ואכן מצאנו איבר כזה, נמחק אותו מהרשימה על ידי שינויי מצביעים ברשימה מקושרת כמו שראינו בכיתה.  
בנוסף, במידה ואכן מצאנו איבר כזה, נוריד את המשתנה size באחד.

**סיבוכיות המתודה:** O(1) בתוחלת  
סיבוכיות מתודה זו היא כסיבוכיות חיפוש מוצלח/לא מוצלח בטבלת האש, ועל כן, סיבוכיות המתודה היא בתוחלת: .

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**private** **class** hashCell<T> **extends** Cell<T>

**----------------------------------------------------------------------------------------------------------  
מחלקה המממשת את תא בתוך טבלת Hash-Map, העוטפת ערכים מהמופע של המחלקה T.  
מחלקה זו מממשת את המחלקה האבסטרקטית Cell<T>  
----------------------------------------------------------------------------------------------------------**

**private** hashCell<T> next – שדה זה מחזיק מצביע לאיבר שבא אחרי האיבר שלנו ברשימה המקושרת.

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**public** hashCell(**int** key, T value, hashCell<T> next)

הבנאי של המחלקה.  
הבנאי מקבל את המפתח key, ואת הערך value.  
הבנאי קורא לבנאי ההורה של המחלקה Cell<T> , יחד עם הערכים key,value: **super**(key, value)  
הבנאי מקבל גם מקבל ארגומנט next שאותו אנחנו מיישמים לשדה next שלנו. בדרך כלל next יהיה null פה, אך הוספנו תכונה זו לשם שלמות המימוש.

**סיבוכיות המתודה:** O(1)

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**public** **void** setNext(hashCell<T> next)

מטרת מתודה זו, היא לשנות את השדה next של איזשהו hashCell<T>.  
היא מקבלת כארגומנט את האיבר שאנחנו צריכים ליישם ל next.

**סיבוכיות המתודה:** O(1)

**public static** **class** maxheap<T>

**----------------------------------------------------------------------------------------------------------  
מחלקה המממשת ערימת מקסימום, שהערכים הצמודים למפתחות בערימה הם מופע של המחלקה T  
----------------------------------------------------------------------------------------------------------**

**private** **final** heapNode<T>[] Heap – מערך בו התאים הם צמתי הערימה, כלומר מופעים של תת-המחלקה heapNode<T>, שעוטפים מופעים של המחלקה הגנרית T.  
המערך לא בהכרח מלא, כלומר קיימים בו ערכי null.  
גודל המערך מאותחל להיות כמות האיברים המקסימלית שאנחנו מחליטים שיהיו בערימה בכל רגע נתון. הכמות הזאת מתקבלת בבנאי המחלקה בתור ארגומנט. לא ניתן לחרוג מכמות הצמתים המקסימלית שהגדרנו.  
מערך זה זה ה Queue שראינו בכיתה של ערימה. כלומר ההורה של הצומת באינדקס ה-i, נמצא באינדקס ה , הילד השמאלי של הצומת באינדקס ה-i נמצא באינדקס , והילד הימני של הצומת באינדקס ה-i נמצא באינדקס .

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**private** **int** size – מספר הצמתים שקיימים בערימה (מספר הצמתים ששונים מnull בHeap)

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**public** maxHeap(**int** maxSize)

בנאי המחלקה.  
נאתחל את size להיות 0.  
אנחנו מקבלים כארגומנט maxSize, את מספר הצמתים המקסימלי שיכולים להיות קיימים בערימה בו זמנית. ועל כן, אנו מאתחלים את Heap להיות מערך של heapNode<T>[maxSize], כלומר מערך של maxSize צמתים (כולם ריקים ברגע האתחול).

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**private** **int** parent(**int** pos)

עבור צומת שנמצא באינדקס pos במערך הערימה, נחזיר את אינדקס ההורה שלו שהוא (תיאורטית קיים לו אב. למשל במידה והוא השורש ואין לו הורה, המתודה פשוט תחזיר -1. בדברים אלה נטפל בקפידה בהמשך המימוש).

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**private** **int** leftChild(**int** pos)

עבור צומת שנמצא באינדקס pos במערך הערימה, נחזיר את אינדקס הילד השמאלי שלו שהוא (תיאורטית קיים לו ילד שמאלי. למשל במידה ואין לו ילד שמאלי, המתודה פשוט תחזיר אינדקס של צומת שהוא null, ויכול להיות שהמתודה תחזיר גם אינדקס שמחוץ ל maxSize. בדברים אלה נטפל בקפידה בהמשך המימוש).

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**private** **int** rightChild(**int** pos)

עבור צומת שנמצא באינדקס pos במערך הערימה, נחזיר את אינדקס הילד השמאלי שלו שהוא (תיאורטית קיים לו ילד ימני. למשל במידה ואין לו ילד ימני, המתודה פשוט תחזיר אינדקס של צומת שהוא null, ויכול להיות שהמתודה תחזיר גם אינדקס שמחוץ ל maxSize. בדברים אלה נטפל בקפידה בהמשך המימוש).

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**private** **int** biggerChild(**int** pos)

מתודה זו מחזירה את עבור צומת שנמצא באינדקס pos, את הבן של אותו צומת, שבעל המפתח הגדול יותר. היא עושה זאת על ידי השוואות פשוטות בין ילדיו של הצומת (אם קיימים לו).  
במידה ולא קיים לצומת בנים, היא מחזירה -1.  
מטרת מתודה זו היא לשמש בעת צורך ב Heapify-Down, בו לפי המימוש שראינו בכיתה, לשם היעילות של המבנה, נחליף צמתים ב Heapify-Down עם הבן הקטן מבין השניים במידת האפשר (למשל אם קיים רק אחד אין ברירה אלה להחליף איתו – למרות שאני בספק שדבר זה קורה).

**סיבוכיות המתודה:** O(1)  
ובכן, ביצענו מספר קבוע של פעולות השוואה

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**private** **boolean** isLeaf(**int** pos)

מתודה זו מקבלת אינדקס כלשהו של צומת במערך הערימה Heap, ומחזירה true במידה והצומת הוא עלה בערימה, אחרת false.  
היא עושה זאת על ידי בדיקה פשוט של האם שני הבנים שלו הם null. זה נבדק על ידי גישה לבן השמאל. במידה והבן השמאלי בעל אינדקס שגדול מהאינדקס המקסימלי של צומת כלשהו בערימה, אז בהכרח הבן השמאלי הוא null, **ובהכרח** הבן הימני הוא null (שכן הבן הימני בעל אינדקס אפילו יותר גדול), ועל כן הצומת הינו עלה, ונחזיר true, **במידה ו:**  
במידה ו pos הוא אינדקס שיותר גדול מצומת בעל האינדקס המקסימלי, נחזיר false.  
גישה לצומת בעל האינדקס המקסימלי תתבצע באמצעות המתודה getMaxIndex() שנתעד בהמשך.

**סיבוכיות המתודה:** O(1)  
ובכן, ביצענו מספר קבוע של פעולות השוואה

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**private** **void** swap(**int** pos1, **int** pos2)

מתודה זו מחליפה בין הצמתים שנמצאים באינדקסים pos1, pos2 במערך הערימה.  
החלפות מצביעים פשוטים של מערך.  
נוסף על כך, כל צומת בערימה, ממומש על ידי heapNode<T>, שמתחזק את השדה pos, שמאחסן את האינדקס של הצומת במערך של הערימה. ועל כן, נחדש את השדה הזה באמצעות מתודות עזר של heapNode.

**סיבוכיות המתודה:** O(1)  
ובכן, ביצענו מספר קבוע של פעולות החלפות מצביעים

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**public** heapNode<T> addNode(T node, **int** key)

מתודה זו מוסיפה את האיבר node שהוא מופע של המחלקה T, יחד עם המפתח key, ועם האינדקס שלו במערך הערימה – שהוא size.  
ראשית, היא יוצרת בשבילו איבר heapNode<T> שיעטוף אותו ואת המפתח, ומוסיפה את איבר זה לסוף מערך הערימה Heap (סוף המערך הכוונה היא שהוא יהיה האיבר האחרון במערך שלא יהיה null, כלומר תכניס אותו לאינדקס ה size), ונעלה את size באחד.  
המתודה מחזירה את האיבר שהוספנו (את האיבר העוטף, ולא את node) – שכן במידה ואנחנו רוצים לשמור את המצביע לצומת (כפי שאנחנו רוצים לעשות לכל צומת בגרף שאנחנו מכניסים), נרצה שהמתודה הזו תחזיר את המצביע.  
ניקח את האינדקס המקסימלי של צומת במערך של הערימה וניישם אותו למשתנה כלשהו curr.  
במידה והאינדקס הזה הוא 0 (כלומר הערימה הייתה ריקה לפני ההוספה), לא נעשה כלום ונחזיר את האיבר כפי שצריך.  
אחרת, נבצע לולאה, בה תנאי העצירה הוא כאשר האיבר שהכנסנו בהכרח בעל מפתח הקטן או שווה מהמפתח של ההורה שלו.   
בכל איטראציה של הלולאה, מידה ומתקיימת איטראציה זה אומר שהאיבר שהכנסנו בעל מפתח שגדול מהמפתח של ההורה שלו, ועל כן יש לבצע פעולת Heapify-Up (שכן האיבר נוסף כעלה לסוף מערך הערימה). ובכן, נחליף את האיבר עם ההורה שלו באמצעות הפעולה swap (כל איבר מחזיק בתוכו את ה pos שלו), וניישם את curr להיות ההורה שלו באמצעות parent(curr), ונמשיך לאיטראציה הבאה.  
איטראציה תעצור את ריצת הלולאה במידה ו curr הוא האינדקס של השורש בערימה/הערימה תקינה

**סיבוכיות המתודה:** O(log n)  
סיבוכיות המתודה היא כסיבוכיות עומק הערימה, שכן עלינו לעשות פעולת Heapify פשוטה, מלבד לפעולות הזמן הקבוע שביצענו קודם לכן כמו השוואות וכו'.  
על כן, סיבוכיות המתודה היא O(log n), כנדרש.

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**public** **void** deleteNode(heapNode<T> node)

מתודה זו מקבלת כארגומנט איזשהו מצביע לצומת בערימת המינימום, ורוצה למחוק את אותה הצומת מן הערימה.  
ובכן, כפי שראינו ובכיתה, התהליך מתבצע בדרך הבאה:  
1. החלפה של האיבר הנתון עם האיבר האחרון שנוסף לערימה: swap(node.getPos, getMaxIndex())  
2. מחיקה של האיבר האחרון בערימה (שזה שבעצם אחרי ההחלפה זה האיבר שאנו רוצים למחוק), ע"י פשוט שינוי האיבר האחרון במערך הערימה ל null (כלומר Heap[size-1]=null), והורדה של כמות הצמתים בערימה באחד (כלומר size -= 1).  
3. לבצע פעולת Heapify על הצומת שהחלפנו, כלומר על הצומת שכרגע נמצאת באינדקס המקורי של האיבר אותו רצינו למחוק. את זה נעשה באמצעות מתודת העזר Heapify(Heap[node.getPos()]), אותה נתעד עוד רגע.  
  
כמובן שיש צורך להתייחסות למקרים הפרטיים של המתודה, למשל אם יש בערימה רק איבר אחד, אז אין צורך בביצוע פעולת Heapify, או שלמשל האיבר שאנו רוצים למחוק הוא האיבר שנמצא באינדקס המקסימלי של מערך הערימה (node.getPos() == getMaxIndex()), גם אין צורך בפעולת Heapify.

**סיבוכיות המתודה:** O(log n)  
ובכן, יש בחובתנו לבצע מספר קבוע של פעולות השוואה, ופעולת Heapify אותה נתעד בהמשך, על כן, סיבוכיות המתודה היא O(log n), כנדרש.

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**public** **void** Heapify(heapNode<T> node)

מתודה זו מבצעת Heapify **כלשהו** (למעלה או למטה) לצומת node שאנו מקבלים כארגומנט.  
ובכן, נוציא את אינדקס הצומת אל תוך משתנה: int pos = node.getPos()  
כעת, נפעיל לולאת while, עם תנאי עצירה של תקינות האינדקס בלבד (שכן בתוך הלולאה נבצע הרבה בדיקות שברובן נסיים בקריאת break ו continue, ובסופו של דבר במידה ולא ביצענו שום continue, אנחנו נבצע break. הוספנו את מכניזם זה בשביל לדמות פעילות רקורסיבית, כלומר הלולאה עצמה מתופעלת בעיקר מתוך הקוד שלה, ולא על פי תנאי העצירה כל כך – החלטנו להעביר את הרקורסיה לצורה איטראטיבית, שכן הדבר פגע לנו רבות במדידות של התרגיל בית הקודם.)  
ובכן, תנאי העצירה של הלולאה או שהאינדקס pos הוא אי שלילי (parent(pos) תחזיר -1, במידה וpos=0), ושהאינדקס pos קטן או שווה מהאינדקס המקסימלי של צומת במערך הערימה.  
כעת, נסביר שלב שלב מה כל איטראציה עושה:  
  
1. נבדוק ראשית אם יש לבצע פעולת Heapify-Up. ובכן, מקרה זה יכול להתקיים אך ורק כאשר pos הוא לא האינדקס של השורש **ו** כאשר המפתח של הצומת ב pos גדול ממש מהמפתח של הצומת ב parent(pos) (כלומר של ההורה שלו).  
**במידה ואכן צריך**: נבצע החלפה בין הצומת להורה שלו ע"י swap(pos, parent(pos)). כעת אנו צריכים להמשיך בלולאה עם הצומת שהחלפנו איתו, לכן נחליף את pos ב parent(pos), כלומר  
pos = parent(pos), ונמשיך לאיטראציה הבאה ע"י continue, בשביל לבצע עוד Heapify במקרה הצורך.

2. יש צורך לבצע פעולת Heapify-Down. ובכן, מקרה זה יכול להתקיים רק כאשר אחד מהבנים של הצומת באינדקס pos בעל מפתח גדול ממש מהמפתח של הצומת באינדקס pos.  
ובכן, במידה ואכן יש צורך בכך, ניקח את האינדקס של הבן עם המפתח **הגדול יותר**. נעשה זאת באמצעות biggerchild = biggerChild(pos).  
במידה ו biggerchild גדול מ0 (המתודה biggerChild(pos) תחזיר -1 במידה ואין לצומת בנים), נמשיך בפעולת ה Heapify, אחרת, נשבור את הלולאה ע"י break.  
ובכן, במקרה שאכן יש לצומת בן כלשהו, וכאשר biggerchild הוא האינדקס של הבן עם המפתח הגדול יותר, אם הצומת שנמצאת באינדקס biggerchild בעלת מפתח הגדול ממש מהמפתח של הצומת, נחליף בין הצמתים, ע"י swap(pos, biggerchild), ונשנה את pos להיות biggerchild, ונמשיך לאיטראציה הבאה בכדי לבדוק אם יש צורך בעוד פעולות Heapify, ע"י continue.

**סיבוכיות המתודה:** O(log n)  
סיבוכיות המתודה היא כסיבוכיות ביצוע פעולת Heapify שראינו בכיתה – O(log n).  
ובכן, זה אכן סיבוכיות O(log n), שכן אנחנו אף פעם לא נבצע בקריאה אחת של Heapify, **שני כיוונים של Heapify,** שכן צורת המימוש של הערימה (Heapify-Down רק עם ה biggerChild) מונעת מדבר זה לקרות, ולכן, מספר האיטראציות שהתבצעו בלולאה, יהיה לכל היותר כעומק הערימה (שכן אנחנו **רק עולים** או **רק יורדים**), ולכן יש O(log n) איטראציות כאשר כל אחת בעלת סיבוכיות זמן קבועה, ועל כן סיבוכיות המתודה היא O(log n)).

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**public** heapNode<T> getMax()

מתודה זו מחזירה את הצומת בעל המפתח המקסימלי של הערימה. ובכן, לפי דרך המימוש של ערימת מקסימום, כל שעלינו לעשות זה לגשת לצומת באינדקס ה-0 במערך הערימה Heap, ולהחזיר את הצומת שנמצאת שם. במידה והערימה ריקה, המתודה תחזיר null באופן טבעי (שכן Heap[0]=null).

**סיבוכיות המתודה:** O(1)

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**public** **int** getSize()

מתודה זו מחזירה את מספר הצמתים שנמצאים בערימה. זה ה getter של השדה size.

**סיבוכיות המתודה:** O(1)

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**public** **int** getMaxIndex()

מתודה זו מחזירה את האינדקס האחרון במערך הערימה Heap, שהינו בעל ערך שונה מ null. כלומר, מתודה זו בעצם מחזירה את הצומת הכי ימני ב'רמה' האחרונה של ה'עץ'. במידה והערימה ריקה, נחזיר -1.   
מתודה זו עושה זאת על ידי החזרה של size-1.

**private** **class** heapNode<V> **extends** Cell<V>

**----------------------------------------------------------------------------------------------------------  
מחלקה המממשת צומת בערימת מקסימום. הצומת עוטף ערכים שהם מופעים של המחלקה V  
מחלקה זו מממשת את המחלקה האבסטרקטית Cell<V>  
----------------------------------------------------------------------------------------------------------**

**private** **int** pos – שדה המחזיק את האינדקס של הצומת במערך הערימה.

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**public** heapNode(**int** key, V node, **int** pos)

בנאי המחלקה.   
אנו קוראים ראשית לבנאי המחלקה המורישה Cell<V> ע"י super(key, value), ואחר כך מאתחלים את השדה this.pos להיות הארגומנט שקיבלנו pos.

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**public** **int** getPos()

ה getter של השדה pos.

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**public** **void** setPos(**int** pos)

ה setter של השדה pos.

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**public** **void** changeKey(**int** key)

מתודה זו משנה את המפתח של הצומת.  
מכיוון שהמפתח השתנה, יש חשש לצורך בפעולת Heapify. ועל כן, אנו קוראים למתודה Heapify של המחלקה ההורית maxHeap, ושולחים אליה את הצומת ע"י maxheap.this.Heapify(this).

**סיבוכיות המתודה:** O(log n)  
סיבוכיות המתודה היא כסיבוכיות המתודה Heapify.

**מדידות**

|  |  |  |
| --- | --- | --- |
| מספר סידורי |  | דרגה מקסימלית בגרף |
| 6 | 64 | 5 |
| 7 | 128 | 7 |
| 8 | 256 | 10 |
| 9 | 512 | 7 |
| 10 | 1024 | 8 |
| 11 | 2048 | 8 |
| 12 | 4096 | 8 |
| 13 | 8192 | 8 |
| 14 | 16384 | 11 |
| 15 | 32768 | 9 |
| 16 | 65536 | 10 |
| 17 | 131072 | 11 |
| 18 | 262144 | 11 |
| 19 | 524288 | 12 |
| 20 | 1048576 | 12 |
| 21 | 2097152 | 12 |

**מסקנות המדידות**

* הדרגה המקסימלית גדלה ביחס ל n בקצב יותר איטי מקצב לוגריתמי.
* דרך הבנייה של הגרף הראנדומי שבחרנו:  
  - תפעלנו לולאה בת איטראציות, כאשר בכל איטראציה:  
  1. בחרנו באופן ראנדומלי לחלוטין שני צמתים. ההסתברות שהצמתים הם אותה צומת ושעלינו לדלג לאיטראציה הבאה היא: .  
  2. תחזקנו טבלת האש שמחזיקה קבוצות HashSet שבעצם מייצגות את הקשתות שהכנסנו עד כה לגרף (עבור קשת , הטבלה ייצגה אותה בטבלת ההאש שלנו).  
  על כן, בכל איטראציה, במידה והקשת שאנחנו בודקים **אינה** לולאה (כלומר ששלב 1 עבר בהצלחה ללא עצירת האיטראציה), אז התוחלת שיש התנגשות ושהקשת כבר נוספה לגרף היא: . ולכן, בכל איטראציה, ההסתברות שהקשת שייצרנו קיימת בגרף ושיש לעבור לאיטראציה הבאה היא: .  
  סך הכל, בכל איטראציה, ההסתברות שהקשת שייצרנו לא תקינה, היא , *ועל כן מימשנו זאת בזמן סביר מבלי להסתכן בלהכניס את אותה הקשת פעמיים.*
* לאורך כתיבת המדידות, צירפנו כל צומת לגרף עם משקל דיפולטי של 1.  
  כיוון שהדרגה של כל צומת תהיה המשקל הסביבתי של הצומת פחות 1. ובכן:  
  בהינתן צומת כלשהו, המשקל הסביבתי שלו הוא:

כאשר היא הדרגה של (עבור צומת נסמן ב- את *משקל הצומת וב- את שכניו של . נגדיר את משקל הסביבה של להיות ונסמנו ב-).*

*ולכן, כאשר נרצה לדעת מה הדרגה של הצומת בעל המשקל הסביבתי המקסימלי ביותר, כל שעלינו לעשות הוא לגשת למשקל הסביבתי שלו ולהחסיר ב 1.*

* ***תשובה:*** *בהתייחס לבעיית הכדורים שצורפה למטה, ניתן להתייחס לצמתים בתור הסלים ולקשתות בתור הכדורים, כלומר ישנם* n כדים ואנו מכניסים m=n כדורים, על *כן, אנו מצפים שעומס הכד המקסימלי (הצומת בעל הדרגה המקסימלית) יהיה בהסתברות* ***מאוד גבוהה*** *() מסדר גודל של: (או כפי שכתוב בנוסחא המפורטת, )*.  
  **מסקנה:**  
  *התצפיות שלנו מאששות זאת ויתר על כן, נראה כי הקבוע הרלוונטי הוא:   
  כלומר, דרגתו של הצומת עם המשקל הסביבתי המקסימלי, היא בקירוב טוב*