מסמך תיעוד ומדידות

המחלקה Item

שדות

שם	פירוט
private int key	מכיל את מפתח ה-Item
private String info	מכיל את המידע (הערך) של ה-Item

פונקציות

חתימה	פירוט	סיבוכיות
public Item (int key, String info)	בנאי - מאתחל Item חדש ומציב בשדות את הארגומנטים	אתחול והצבה - O(1)
public int getKey()	מחזירה את המפתח של האיבר	גישה לשדות - O(1)
public String getInfo()	מחזירה את הערך של האיבר	גישה לשדות - O(1)

המחלקה CircularList

שדות

שם	פירוט
protected Item[] array;	המערך בו יוחזקו ערכי הרשימה
protected int maxLen;	אורך הרשימה המירבי האפשרי
protected int length = 0;	אורך הרשימה
protected int start = 0;	אינקדס התחלת הרשימה בתוך המערך

פונקציות

חתימה	פירוט	סיבוכיות
public CircularList (int maxLen)	איתחול מבנה נתונים מסוג רשימה מעגלית. מוקצה בזיכרון מערך באורך maxLen המאותחל ל-null.	O(maxLen)
public Item retrieve(int i)	הפונקציה מחזירה את האיבר במיקום ה-i עבור אינדקס חוקי בין 0 (כולל) ל-n-1 (לא כולל). אם i לא אינדקס חוקי אז יוחזר null.	O(1)
public int insert(int i, int k, String s)	הכנסת איבר בעל מפתח k וערך s לאינדקס i ברשימה. אם ההכנסה אפשרית יוחזר הערך 0, אם לא (i<0 או i>n או n=maxLen) יוחזר הערך -1.	$O(\min\{i+1, n-i+1\})$
public int delete(int i)	מחיקת איבר באינדקס ה i ברשימה. הפונקציה מחזירה את הערך 0 אם האינדקס חוקי, אחרת תחזיר -1.	$O(\min\{i+1, n-i+1\})$

המחלקה TreeList

שדות

public AVLTree tree; - עץ המכיל את איברי הרשימה.

פונקציות

חתימה	פירוט	סיבוכיות
public TreeList()	בנאי, מאתחל את השדה עץ עם עץ חדש ריק	השמה לשדות - O(1)
public Item retrieve(int i)	מחזיר את האיבר ה i-ברשימה, אם האיבר אינו קיים יוחזר null	O(logn) קריאה לפונקציה select הפועלת בסיבוכיות לוגריתמית ולאחר מכן גישה לשדות של הצומת
public int insert(int i, int k, String s)	מכניסה איבר בעל מפתח k וערך s לרשימה במקום ה i-ומחזירה 0 כל עוד i אינדקס חוקי בין 0 לאורך הרשימה (כולל). מחזירה 1- אם האינדקס שלילי או גדול מאורך הרשימה.	O(logn) קוראת לפונקציה listInsert המבצעת פעולת select לוגריתמית, ומכניסה את האיבר החדש בתור בן להורה המתאים. לאחר מכן מתבצע עדכון של גובה וגודל הצמתים במסלול מהאיבר המוכנס ועד השורש (logn) ולבסוף מתבצע לכל היותר גלגול אחד במקום שבו נוצר עבריין.
public int delete(int i)	i-מוחקת מהרשימה את האיבר במקום ה ומחזירה 0. מחזירה 1- אם האינדקס שלילי או גדול מגודל הרשימה	O(logn) קוראת לפונקציה listDelete שזו מבצעת פעולת select לוגריתמית, ומבצעת מחיקה מהעץ (באופן לוגריתמי בדומה למחיקה מעץ AVL) ולאחר מכן מעדכנת את מסלול הצמתים עד השורש (logn) ומבצעת גלגולים בהתאם לצורך (לכל היותר logn).

פונקציות עזר

חתימה	פירוט	סיבוכיות
private void shiftR(int i, String mode)	פונקציית עזר להזזת אינדקסים ימינה (העלאת ערכם באחד) במערך.	בהכנסה O(i) ובמחיקה O(n-i)
private void shiftL(int i, String mode)	פונקציית עזר להזזת אינדקסים שמאלה (הורדת ערכם באחד) במערך.	בהכנסה O(n-i) ובמחיקה O(i)

המחלקה AVLTree

שדות

שם	פירוט
private IAVLNode root	שורש העץ, מאותחל ל-null
private int size	גודל העץ, מאותחל ל-0
private int index	שדה עזר לחישובים, מאותחל ל-0
private IAVLNode minimum;	צומת מינימלי בעץ, מאותחל ל-null
private IAVLNode maximum;	צומת מקסימלי בעץ, מאותחל ל-null

פונקציות

חתימה	פירוט	סיבוכיות
public boolean empty()	מחזירה TRUE אם העץ ריק ואחרת FALSE	O(1) בדיקה האם השורש של העץ הוא null
public String search(int k)	מחזירה את ערכו של צומת בעל מפתח k אם הוא נמצא בעץ, אחרת, מחזירהnull	O(logn) קריאה לפונקציית העזר searchNode שרצה בסיבוכיות לוגריתמית, ולאחר מכן החזרת ערך הצומת ב(O(1
public int insert(int k, String i)	מכניסה לעץ צומת חדש בעל מפתח k וערך i במקום המתאים, מתחזקת גובה, גודל, מקסימום ומינימום. מחזירה את מספר האיזונים שהתבצעו.	O(logn) מבצעת פעולת search לוגריתמית, מכניסה את האיבר החדש בתור בן להורה המתאים. לאחר מכן מתבצע עדכון של גובה וגודל הצמתים במסלול מהאיבר המוכנס ועד השורש (logn) ולבסוף מתבצע לכל היותר גלגול אחד במקום שבו נוצר עבריין
public int delete(int k)	מוחקת מהעץ את הצומת בעל מפתח k ומחזירה את מספר האיזונים שהתבצעו. מתחזקת גובה, גודל, מקסימום ומינימום.	O(logn) המחיקה מתחלקת ל3 מקרים (עלה, בן יחיד, שני בנים) כאשר בשני המקרים הראשונים מתבצעת השמה של מצביעים ב-O(1), אך במקרה השלישי יש למצוא את העוקב בסיבוכיות לוגריתמית. לאחר מכן מתבצעת לולאת עדכון של גדלי וגובהי הצמתים שהשתנו, לכל היותר logn ולבסוף לולאת גלגולים בהתאם לצורך. סה"כ O(logn)
public String min()	מחזירה את ערכו של השדה minimum שהוא האיבר בעץ בעל המפתח המינימלי	גישה לשדה והחזרת ערך - O(1)

public String max()	מחזירה את ערכו של השדה maximum שהוא האיבר בעץ בעל המפתח המקסימלי	גישה לשדה והחזרת ערך - O(1)
public int[] keysToArray()	מחזירה מערך ממוין המכיל את כל המפתחות בעץ, או מערך ריק אם העץ ריק.	O(n) מבצעים הילוך In-Order בעץ ומוסיפים בביקור של כל צומת את המתפתח שלה למערך.
public String[] infoToArray()	מחזירה מערך מחרוזות המכיל את כל המחרוזות בעץ, ממוינות על פי סדר המפתחות. אם העץ ריק, יוחזר מערך ריק.	O(n) מבצעים הילוך In-Order בעץ ומוסיפים בביקור של כל צומת את ערך המחרוזת שלה למערך.
public int size()	מחזירה מספר (שלם) של הצמתים בעץ. אם העץ ריק, יוחזר הערך 0.	O(1) החזרת השדה size.
public IAVLNode getRoot()	מחזירה מצביע לשורש (אם קיים), אשר מממש את המנשקIAVLNode. אם לא קיים שורש יוחזר null.	O(1) החזרת השדה root.

פונקציות עזר

חתימה	פירוט	סיבוכיות
private int getBalance(IAVLNode x)	מחזירה את ערך המאזן של צומת נתון.	גישה לשדות וחישוב - O(1) אריתמטי
private void leftRotate(IAVLNode x)	מבצעת רוטציה שמאלה בעץ עבור צומת נתון.	שינוי מצביעים - O(1)
private void rightRotate(IAVLNode x)	מבצעת רוטציה ימינה בעץ עבור צומת נתון.	שינוי מצביעים - O(1)
private void keysOrderWalk(IAVLNode x, int[] arr)		בעץ In-Order הליכת - O(n)
private void infoOrderWalk(IAVLNode x, String[] arr)		בעץ In-Order הליכת - O(n)
private IAVLNode searchNode (int k)	מחזירה את הצומת שמפתחו הוא k אם הוא נמצא בעץ, אחרת, מחזירה null	מעבר על תת עץ - O(logn) בודד בכל איטרציה, כמות האיטרציות המקסימלית זהה לגובה העץ
private void bypass(IAVLNode deletedNode, IAVLNode child)	בודקת האם הצומת שמיועד למחיקה היא בן ימני או שמאלי של ההורה שלו, ומבצעת עיקוף מההורה אל הבן בהתאמה	שינוי מצביעים - O(1)
public static IAVLNode findSuccessor(IAVLNode x)	עבור צומת מסויים, מחזירה את הצומת ה"עוקב" - בעל המפתח הכי קטן שגדול ממנו. מחזירה null אם אין כזה	כתלות באורך העץ - O(logn)
public static IAVLNode minInSubTree(IAVLNode root)	מחזירה את הצומת המינימלי בתוך תת עץ מסוים	כתלות באורך העץ - O(logn)

public void listInsert(int i, int k, String s)	פונקציה המיועדת רשימה עצית, הכנסת איבר חדש לעץ המבוססת על אינדקס נתון ולא על המפתח	O(logn) מבצעת פעולת select לוגריתמית, ומכניסה את האיבר החדש בתור בן להורה המתאים. לאחר מכן מתבצע עדכון של גובה וגודל הצמתים במסלול מהאיבר המוכנס ועד השורש (logn) ולבסוף מתבצע לכל היותר גלגול אחד במקום שבו נוצר עבריין
public IAVLNode treeSelect(int k)	מקבלת דרגה k ומחזירה nullאם העץ ריק או ש k לא מהווה דרגה בעץ, אחרת מחזיקה את הצומת בעל הדרגהk	O(logn) קוראת לפונקציה select שזו רצה בסיבוכיות לוגריתמית על העץ (חיפוש איטרטיבי בתתי העצים)
public static IAVLNode select(IAVLNode root, int k)	מקבלת צומת של עץ ואינדקס k ובאופן רקורסיבי מחפשת את הצומת בדרגה זו בעץ	O(logn) חיפוש בעץ AVL בסיבוכיות לוגריתמית
public static int getRank(IAVLNode x)	מקבלת צומת x ומחזירה את דרגתו בעץ	O(logn) חיפוש צומת בעץ AVL בסיבוכיות לוגריתמית
private int rotationsManager(IAVLNode x, int BF)	בהינתן צומת x ו BFשל אותו צומת, בודקת איזה גלגול יש לעשות, מבצעת את הגלגול ומחזירה את מספר הגלגולים שבוצעו	O(1) בדיקת שדות ושינוי מצביעים (כל גלגול מהווה החלפת מצביעים מסוימים)
public void listDelete(int k)	פונקציה המיועדת רשימה עצית, מחיקת איבר בעל אינדקס מהעץ בלי התייחסות למפתחות הצמתים	O(logn) מבצעת פעולת select לוגריתמית, מוחקת את האיבר הרצוי בדומה למחיקה מעץ AVL. לאחר מכן מתבצע עדכון של גובה וגודל הצמתים במסלול מהאיבר המוכנס ועד השורש (logn) ולבסוף מתבצעים גלגולים במקומות שבהם נוצרו עבריינים
public static IAVLNode findPredecessor(IAVLNode x)	עבור צומת מסויים, מחזירה את הצומת ה"קודם" - בעל המפתח הכי גדול שקטן ממנו. מחזירה null אם אין כזה	כתלות באורך העץ - O(logn)
private static int updateHeight(IAVLNode x)	מקבלת צומת x ומעדכנת את גובהו	O(1) גישה לשדות וחישוב אריתמטי

private static int updateSize(IAVLNode x)	מקבלת צומת x ומעדכנת את גודלו	O(1) גישה לשדות וחישוב אריתמטי
public static IAVLNode minInSubTree(IAVLNode root)	מקבלת שורש ומחזירה את הצומת בעל המפתח המינימלי (הצאצא השמאלי ביותר), או null אם זהו תת עץ ריק	הליכה שמאלה על עץ - O(logn)
public static IAVLNode maxInSubTree(IAVLNode root)	מקבלת שורש ומחזירה את הצומת בעל המפתח המקסימלי (הצאצא הימני ביותר), או null אם זהו תת עץ ריק	הליכה ימינה על עץ - O(logn)

מדידות

1. הכנסת איברים המדגימה את היתרון של רשימה מעגלית על-פני רשימה עצית

מספר סידורי	**מספר פעולות**	**זמן הכנסה ממוצע עבור רשימה מעגלית**	**זמן הכנסה ממוצע עבור רשימה עצית**	**כמות גלגולים ימינה ממוצעת עבור רשימה עצית**	**כמות גלגולים שמאלה ממוצעת עבור רשימה עצית**
1	10000	116.2	1787	0.999	0
2	20000	43.69	627.1	0.999	0
3	30000	41.23	430.5	1	0
4	40000	36.6	567.2	1	0
5	50000	44.66	399.1	1	0
6	60000	23.73	190.8	1	0
7	70000	18.34	179.4	1	0
8	80000	18.54	196	1	0
9	90000	18.38	205.7	1	0
10	100000	18.2	209.8	1	0

בחרנו לבצע הכנסה בתחילת הרשימה (באינדקס 0), משום שהכנסה זו תיקח O(1) פעולות ברשימה המעגלית והן להכניס איבר לאינדס (start-1)%maxLen ולעדכן את שדה ה-start לאינדקס זה. עבור רשימה עצית, יהיה צורך ב-O(logn) פעולות. קונקרטית במימוש שלנו, הגישה לאיבר המינימום (או המקסימום) היא באמצעות מצביע ב-O(1) אך עדיין קיים הצורך לעדכן פרמטרים במעלה העץ ופעולה זו לוקחת בבירור O(logn) שכן מתחילים מצומת שהוא עלה. לכן, ניתן לצפות כי זמן ההכנסה הממוצע עבור רשימה מעגלית לא ישתנה עם הכנסת איברים נוספים, ואילו עבור הרשימה העצית הזמן יעלה באופן לוגריתמי.

במדידות שביצענו ניתן לראות כי זמן ההכנסה הממוצע של איבר ברשימה מעגלית אכן קבוע יחסית בעוד שברשימה העצית הוא עולה באיטיות כתלות במספר הצמתים שהכנסנו (וכמעט כל הכנסה דורשת ביצוע גלגול ימינה). נציין כי המדידות שביצענו מושפעות מפעולות חישוב נוספות שהמחשב מבצע ואלו מהווים הסבר אפשרי לקפיצות במדידת הזמן בין הניסויים.

משמעות המדידה היא שעבור מבנה נתונים הדורש הכנסת איברים בהתחלת הרשימה (או בסופה), נעדיף רשימה מעגלית על-פני רשימה עצית.

2. הכנסת איברים המדגימה את היתרון של רשימה עצית על-פני רשימה מעגלית

מספר סידורי	מספר פעולות	זמן הכנסה ממוצע עבור רשימה מעגלית	זמן הכנסה ממוצע עבור רשימה עצית	כמות גלגולים ימינה ממוצעת עבור רשימה עצית	כמות גלגולים שמאלה ממוצעת עבור רשימה עצית
1	10000	13239	1238	0.81	0.812
2	20000	24244	655.1	0.812	0.812
3	30000	34735	594.8	0.812	0.812
4	40000	45645	609	0.812	0.813
5	50000	57846	327.6	0.812	0.812
6	60000	68907	384.5	0.812	0.812
7	70000	83726	252.7	0.812	0.812
8	80000	95319	254.9	0.812	0.812
9	90000	106993	246.9	0.812	0.812
10	100000	120096	239.9	0.812	0.812

בחרנו לבצע הכנסה באמצע הרשימה (באינקדס $\left\lfloor \frac{n}{2} \right\rfloor$) שכן במצב זה הרשימה המעגלית תצטרך לבצע $\left\lfloor \frac{n}{2} \right\rfloor$ העתקות של איברים לאינדקס צמוד. כלומר הכנסה זו אמורה להתבצע בסיבוכיות לינארית ביחס למספר האיברים ברשימה.

עבור רשימה עצית, כל פעולה מתבצעת ב-O(logn) ולכן זמן ההכנסה צפוי לגדול לוגריתמית כתלות במספר האיברים.

במדידות שביצענו ניתן לראות שזמן ההכנסה לרשימה מעגלית אכן עולה באופן לינארי, בעוד שזמן ההכנסה לרשימה עצית קטן ככל שמספר ההכנסות בניסוי היה גבוה יותר. על תגלית זו, ועדת פרס טיורינג העמידה אותנו לקבלת הפרס השנה 😉. (נציין שככל הנראה המדידות הושפעו מפעולות אחרות וכבדות יותר שהמחשב ביצע באותו הזמן וכן שטווח הערכים שהתקבל עבור הרשימה העצית נמוך בהרבה מזה שהתקבל עבור הרשימה המעגלית). בנוסף, נשים לב כי בוצעו מספר כמעט זהה של גלגולים ימינה ושמאלה בבדיקה זו.

משמעות המדידה היא שעבור מבנה נתונים הדורש הכנסת איברים באמצע הרשימה נעדיף רשימה עצית על-פני רשימה מעגלית.

3. הכנסת איברים בהתפלגות אחידה

מספר סידורי	מספר פעולות	זמן הכנסה ממוצע עבור רשימה מעגלית	זמן הכנסה ממוצע עבור רשימה עצית	כמות גלגולים ימינה ממוצעת עבור רשימה עצית	כמות גלגולים שמאלה ממוצעת עבור רשימה עצית
1	10000	7813	1112	0.349	0.353
2	20000	12247	1026	0.347	0.343
3	30000	17376	704.2	0.35	0.35
4	40000	23221	635	0.346	0.347
5	50000	30203	612.2	0.351	0.351
6	60000	34738	374.7	0.35	0.347
7	70000	40951	391.1	0.349	0.347
8	80000	46846	406.3	0.349	0.35
9	90000	53076	377	0.351	0.351
10	100000	57995	375.5	0.35	0.35

כיוון שההכנסה מתפלגת אחיד בין האינדקסים, ברשימה מעגלית נצטרך עבור כל הכנסה לבצע בין 0 ל-$\left\lfloor \frac{n}{2} \right\rfloor$ הזזות של איברים לאינדקס צמוד. לכן, בתוחלת נצטרך להזיז $\frac{n}{4}$ איברים עבור כל הכנסה, וזו עדיין סיבוכיות לינארית (אמנם עם מקדם קטן יותר).

עבור הרשימה העצית, כל הכנסה בכל אינדקס תדרוש O(logn) פעולות ולכן נצפה לעלייה איטית בהכנסה כתלות במספר האיברים.

במדידות שביצענו, אכן ניתן לראות עלייה לינארית במשך הזמן הממוצע להכנסה ברשימה מעגלית, וזו עלייה מתונה יותר מאשר במדידה הקודמת. עבור הרשימה העצית ניתן לראות כי משך הזמן הממוצע להכנסה יחסית קבוע, פרט לקפיצות מסוימות בחלק מהניסויים (נציין כי טווח הערכים יחסית מצומצם בהשוואה לרשימה המעגלית).

משמעות המדידה היא שעבור מבנה נתונים הדורש הכנסת איברים במקומות אקראיים (בהתפלגות אחידה) נעדיף רשימה עצית על-פני רשימה מעגלית.