אוניברסיטת תל אביב

סמסטר א' תשפ"ג

**מבני נתונים - פרויקט חקר תכנותי מספר 1 - עץ דרגות**

**ירון ראובן איטח 208935312**

**הקדמה:**

בתרגיל זה שני חלקים:

1. חלק המעשי: מימוש של List באמצעות עץ AVL. עמודים 1-2 במסמך זה מתארים את החלק הזה.
2. חלק ניסויי-תיאורטי: בהתבסס על המימוש מהחלק המעשי, נבצע מספר "ניסויים" עם ניתוח תיאורטי נלווה ושאלות חקר. עמודים 3-5 מתארים את החלק הזה.

**שימו לב:** בסוף המסמך (עמוד 5) ישנן הוראות הגשה – הקפידו לפעול לפיהן.

**תאריך הגשה:** 30/12 .

בנוסף, יש לעקוב אחר השרשור הנעוץ בפורום בו נפרסם הבהרות חשובות.

**חלק מעשי**

**דרישות**

בתרגיל זה נממש את ה ADT **רשימה** באמצעות עץ AVL. לכל איבר ברשימה יש ערך (info). המימוש יהיה **בשפת python 3.9 וצריך להיות מבוסס על קובץ השלד המופיע באתר הקורס**.   
הפעולות שיש לממש הן:

|  |  |
| --- | --- |
| **פעולה** | **תיאור** |
| empty() | הפונקציה מחזירה ערך TRUE אם ורק אם הרשימה ריקה. |
| retrieve(i) | הפונקציה מחזירה את ערך האיבר במקום ה-i אם קיים, אחרת היא מחזירה None. |
| insert(i, s) | הכנסת איבר בעל ערך s לרשימה במקום ה-i, במידה וקיימים לפחות i איברים ברשימה. הפונקציה מחזירה את מספר פעולות האיזון שנדרשו בסה"כ בשלב תיקון העץ על מנת לשמר את תכונת האיזון. |
| delete(i) | מחיקת האיבר במקום ה-i ברשימה, אם הוא קיים. הפונקציה מחזירה את מספר  פעולות האיזון שנדרשו בסך הכל בשלב תיקון העץ על מנת לשמר את תכונת האיזון. אם לא קיימים מספיק איברים ברשימה הפונקציה מחזירה . |
| first() | מחזירה את ערך האיבר הראשון ברשימה, או None ברשימה ריקה. |
| last() | מחזירה את ערך האיבר האחרון ברשימה, או None ברשימה ריקה. |
| listToArray() | הפונקציה מחזירה מערך המכיל את איברי הרשימה לפי סדר האינדקסים, או מערך ריק אם הרשימה ריקה. |
| length() | הפונקציה מחזירה את מספר האיברים ברשימה. |
| permutation() | הפונקציה מחזירה את אותם האיברים ברשימה בסדר אקראי |
| sort() | הפונקציה מסדרת את איברי הרשימה בסדר עולה. |
| concat(lst) | הפונקציה מקבלת רשימה. על הפונקציה לשרשר אותה אל סוף הרשימה הנוכחית. על הפעולה לרוץ בזמן . יש להחזיר את הערך המוחלט של הפרש הגבהים של עצי הAVL שמוזגו. |
| search(val) | החזרת האינדקס הראשון ברשימה בו מופיע הערך val, או אם לא קיים כזה. |

**בנוסף למימוש הפונקציות האלו, יש לממש את מחלקת AVLNode כפי שמתואר בקובץ**. מטעמי נוחות, נדרוש שלכל עלה יהיו 2 בנים "וירטואליים", כלומר, צמתים שלא מייצגים איברים במבנה הנתונים. באופן זה, נוח יותר לממש גלגולים מכיוון שלכל צומת יהיו 2 בנים וזה חוסך טיפול במקרי קצה.

למחלקה AVLNode יש את המתודות הבאות (המפרט המלא נמצא בקובץ השלד):

getHeight – מחזיר את הגובה של הצומת, או אם הצומת הוא וירטואלי.

getValue – מחזיר את הinfo של הצומת או None אם הצומת הוא וירטואלי.

getLeft – מחזיר את הבן השמאלי של הצומת, או None אם אין כזה.

getRight – מחזיר את הבן הימני של הצומת, או None אם אין כזה.

getParent - מחזיר את ההורה של הצומת, או None אם אין כזה.

isRealNode – מחזיר TRUE אם הצומת מייצג צומת אמיתי בעץ (קרי: צומת שאינו וירטואלי).

**הערות חשובות:**

1. **המימוש יבוצע על ידי מילוי קובץ השלד. במידת הצורך, ניתן להרחיב את המימוש** (למשל להוסיף פונקציות עזר שאינן מופיעות בשלד), אך **אסור לשנות את הגדרות הפונקציות לעיל**. על כל הפונקציות/מחלקות להופיע בקובץ יחיד.
2. **אין להשתמש באף מימוש ספרייה של מבנה נתונים.**

**סיבוכיות**

יש לתעד בקוד ובמסמך נפרד (ביותר פירוט) את סיבוכיות זמן הריצה במקרה הגרוע (האסימפטוטית, במונחי O הדוקים) של כל פונקציה **שמכילה לולאות/רקורסיה**, כתלות במספר האיברים בעץ n. עליכם להשיג סיבוכיות זמן ריצה (במקרה הגרוע ביותר) נמוכה ככל הניתן עבור כל אחת מהפונקציות.

**פלט**

אין צורך בפלט למשתמש.

**תיעוד**

בנוסף לבדיקות אוטומטיות של הקוד שיוגש, קובץ המקור ייבדק גם באופן ידני. חשוב להקפיד על תיעוד לכל פונקציה, וכמות סבירה של הערות. **הקוד צריך להיות קריא**, בפרט הקפידו על בחירת שמות משתנים ועל אורך השורות.

יש להגיש בנוסף לקוד גם מסמך תיעוד חיצוני. המסמך יכלול את תיאור המחלקה שמומשה, ואת תפקידו של כל חבר במחלקה. עבור כל פונקציה במחלקה יש לפרט מה היא עושה, כיצד היא פועלת **ומה סיבוכיות זמן הריצה שלה**. בפרט, אם פונקציה קוראת לפונקציית עזר, **יש** להתייחס גם לפונקציית העזר בניתוח. עבור פונקציות שעולות זמן קבוע יספיק להביא רק תיאור קצר ולא לפרט את ניתוח הסיבוכיות.

מחלקת :

מחלקה זו מכילה את השדות הבאים:

* - ערך הצומת.
* - מצביע לבן השמאלי, מאותחל בתור .
* - מצביע לבן הימני, מאותחל בתור .
* - מצביע לאב הצומת, מאותחל בתור .
* - גובה הצומת (גובה של עלה מוגדר להיות , של צומת וירטואלי ).
* - מספר הצמתים בתת העץ כולל הצומת הנוכחי.

מחלקה זו מכילה את הפעולות הבאות אשר **כולן בסיבוכיות** :

* - מחזירה את שדה של הצומת, או אם אין כזה.
* - מחזירה את שדה של הצומת, או אם אין כזה.
* - מחזירה את השדה של הצומת, או אם אין כזה.
* - מחזירה את שדה של הצומת או אם הצומת הוא וירטואלי.
* - מחזירה את שדה של הצומת. עבור צומת וירטואלי הגובה .
* - מחזירה את שדה של הצומת. עבור צומת וירטואלי .
* – מחזירה את הbalance factor של הצומת ע"י חישוב: גובה הבן השמאלי פחות גובה הבן הימני.
* - מגדירה את השדה להיות ה של הקלט.
* - מגדירה את השדה להיות ה של הקלט.
* - מגדירה את השדה להיות ה של הקלט.
* – מגדירה את השדה להיות ה שהוכנס ואת שדה של ה שהוכנס בקלט להיות האיבר עצמו.
* - מגדירה את השדה להיות ה שהוכנס ואת שדה של ה שהוכנס בקלט להיות האיבר עצמו.
* – מגדירה את שדה להיות ה של הקלט.
* - מגדירה את שדה של הצומת לפי הערך של הקלט.
* - מגדירה את שדה של הצומת.
* - מחזירה אם הצומת מייצג צומת אמיתי בעץ. כלומר צומת שאינו וירטואלי וזאת כאשר .
* – מעדכן את שדה של ה בחישוב ה של הבנים שלו כפי שנלמד בכיתה.
* – מעדכן את שדה של ה בחישוב ה של הבנים שלו כפי שנלמד בכיתה.

מחלקת :

מחלקה זו מכילה את השדות הבאים:

* - מחזיק את כמות הצמתים בעץ.
* - מצביע על שורש העץ, על איבר מסוג .
* – מצביע לאיבר הראשון ברשימת האיברים הממוינת בסדר עולה.
* – מצביע לאיבר האחרון ברשימת האיברים הממוינת בסדר עולה.

מחלקה זו מכילה את הפעולות הבאות:

* - פעולה אשר מחזירה אם שורש העץ אינו צומת אמיתי, כלומר העץ ריק, ו- אחרת. פעולה זו בסיבוכיות שכן היא פונה לשדה של העץ ועובדת כמו פונקציית שעובדת בסיבוכיות זו.
* – מחזירה את ערכו של הצומת במיקום ה בגודלו ברשימה. פונקציה זו משתמשת במתודת המפורט תחת פונקציות העזר, אשר בסיבוכיות ועל כן זו הסיבוכיות של מתודה זו.
* – מכניסה את הערך בקלט למיקום ברשימה במידה שקיימים לפחות איברים ברשימה. המתודה מחזירה את מספר פעולות האיזון שנדרשו בסה"כ לטובת תיקון העץ. במתודה זו אנו משתמשים במתודות עזר אשר מחלקות למקרים כל סוג הכנסה: בדיקה של מקרי קצה – הכנסה לאיבר במיקום . במקרה זה נבדוק אם העץ הוא ריק ואם כן פשוט נאתחל את האיבר להיות השורש. אחרת, נמצא את המינימלי אם אינו קיים ואז נכניס את האיבר להיות משמאל למינימלי בתור עלה.

הכנסה לאיבר במיקום הגדול ביותר בעץ הנוכחי. נמצא את המקסימלי אם אינו קיים ואז נכניס את האיבר להיות מימין למקסימלי בתור עלה.

כל מקרה אחר: נמצא את המיקום של באמצעות השימוש ב. אם אין לאיבר בן שמאלי אז נכניס את הצומת בתור עלה. אחרת, נכניס את הצומת בתור עלה עבור ה של האיבר ה שמצאנו.  
הכנסה של עלה מתבצעת באמצעות מתודת עזר

כיסינו את כל המקרים, כעת נשתמש במתודת לתיקון העץ בהכנסה אשר מפורטת במתודות העזר. לבסוף נשתמש במתודת אשר מעדכנת את שדה של כל האיברים שהם הורים לאיבר אשר הוכנס, עד השורש. תוך כדי הפעולה נשמור את מספר הקידומים/סיבובים שבוצעו ונחזיר ערך זה. נבחין כי כל הפעולות הינן בלתי תלויות ולא מקוננות ולכן בסך הכל מדובר בסיבוכיות של .

* *–* תחילה נבדוק מקרי קצה: איבר שרוצים למחוק אינו בתחום, נחזיר . גודל העץ הוא כלומר יש לו רק איבר בודד, נאפס את העץ ונחזיר . בכל מקרה אחר, נבצע ב שחסום על-ידי כדי למצוא את האיבר שנמצא במיקום ה, נכנו . נבדוק אם אנו מוחקים את האיבר המינימלי/המקסימלי ואם כן נמצא את האיבר המתאים להיות המינימלי/מקסימלי הבא (עוקב או קודם). אם הינו עלה, נשתמש בפונקציית ב. אם לצומת שאנו מוחקים יש בן אחד בלבד נשתמש בפונקציית ב. אחרת, לצומת שאנו מוחקים יש שני ילדים ונפעל בהתאם: נמצא את העוקב של ונשמור אותו במשתנה . אם העוקב הוא עלה אז נמחק את העוקב, אחרת העוקב הוא בעל בן ישיר אחד ולכן נשתמש ב. לבסוף נעדכן את ההורה של , אם קיים, שהבן שלו הינו כעת העוקב (שנמחק מהעץ עצמו). אם אין הורה ל הרי שמדובר בשורש ולכן נעדכן את השורש להיות העוקב. לבסוף נעדכן את העוקב כי מעתה ההורה שלו הוא ההורה של וכך נעשה גם עבור הילד הימני והשמאלי של , גם אם הם וירטואלים. נבצע ו החל מהעוקב כדי לתקן את שדות אלו בכל העץ. שתי פונקציות אלה פועלות ב. לבסוף ננתק את הקשר של מההורה שלו ומילדיו (דה פקטו זה מוחק אותו) ונחזיר את מספיק הגלגולים ששמרנו לאורך כל תהליך המחיקה. נבחין כי השימוש בפונקציות חיצוניות הן בלתי תלויות ואינן גוברות על סיבוכיות זו, כלומר כולן לכל היותר בסיבוכיות .
* *–* אם ריק מחזירה , אחרת את ערכו של . וגישה למצביע ב.
* *–* אם ריק מחזירה , אחרת את ערכו של . וגישה למצביע ב.
* *–* נממש פונקציה פנימית של באופן רקורסיבי אשר כל עוד הצומת אמיתי, פותחת קריאה לבן השמאלי, מכניסה את הערך של הצומת למערך שהגיע בקלט ב, ופותחת קריאה לבן ימני. בפונקציה עצמה נבדוק אם מדובר בעץ ריק, אם כן נחזיר מערך ריק. אחרת, נקרא לפונקציה הרקורסיבית ולבסוף נחזיר את העץ. כפי שלמדנו, מתבצע ב וכן כל פונקציה אחרת הינה ב כלומר סך הסיבוכיות הינו .
* - מחזירה את השדה של השורש, אלא אם מדובר בצומת וירטואלי עבורו היא מחזירה את הערך . גישה לשדה בסיבוכיות .
* *–* פונקציה אשר מייצרת עץ חדש (ריק), בוחנת אם העץ הנוכחי ריק ואם כן מחזירה עץ ריק. אחרת, קוראת ל כדי לייצר מערך של הערכים, ואז משתמשת ב כדי למיינם. לבסוף, מכניסה את ערכי המערך לעץ החדש באופן הממוין. פעולת וכן גם סדרת פעולות ה לוקחות ומכיוון שכל הפעולות הינן בלתי תלויות זאת היא סיבוכיות הפונקציה.
* *–* פונקציה אשר בוחנת תחילה אם מדובר בעץ ריק, אם כן נחזיר את העץ הריק. אחרת, נבצע ב למשתנה , נקרא ל ב, נשתמש ב ב לשינוי של הערכים במערך שקיבלנו ולבסוף נשתמש בפונקציה רקורסיבית פנימית אשר מבצעת ובכל פעם משנה את ה ב לצומת ואז מבצעת ב מהמערך לאותו ערך שהוזן. נחזיר את . נבחין כי יצרנו עץ חדש שזהה במבנה שלו לעץ המקורי ולכן אינו דורש פעולות איזון (העץ המקורי הינו מאוזן מלכתחילה), וכן כל הפונקציות שהשתמשנו בהם הם ב וזאת סיבוכיות הפונקציה.
* *–* תחילה נחשב מה הוא הפרש הגבהים בין 2 הרשימות שכן זהו הערך שעלינו להחזיר. לאחר מכן נבדוק מקרי קצה: אם שני העצים ריקים נחזיר רק את ההפרש, אם רק הנוכחי ריק נעדכן את השורש ושאר המצביעים להיות כמו של ונחזיר את הפרש הגבהים. אחרת העץ הנוכחי אינו ריק, ניקח את האיבר המקסימלי בעץ הנוכחי באמצעות שימוש במצביע ב, נמחק אותו בשימוש ב מהעץ הנוכחי ונשתמש בו בתור האיבר המחבר בין שני העצים בשימוש בפונקציית . לבסוף נחזיר את הפרשי הגבהים. נבחין כי אנו משתמשים במתודות עזר בלתי תלויות אשר כל אחת מהן היא לכל היותר בסיבוכיות וזאת הסיבוכיות של פונקציה זו.
* - פונקציה אשר מחפשת ערך אשר התקבל בקלט. נתחיל בגישה לאיבר הראשון ברשימה באמצעות מצביע ולכן ב ואז נבצע פעמים פעולות כדי לבחון אם האיבר הבא הוא במקרה הערך שהתקבל בקלט. אם מצאנו נעצור ונחזיר את מיקומו, אחרת נחזיר . נבחין כי מדובר במקרה הגרוע בסדרה של פעולות עוקב ולכן מדובר בסיבוכיות .
* - מחזירה את השדה של העץ אם אינו ריק. גישה לשדה בסיבוכיות .

מתודות עזר:

* – פונקציה אשר מחזירה את הצומת הראשון אשר גודלו הינו לכל הפחות עבור אי-שלילי שקטן ממספר האיברים במבנה. מכיוון שיש ברשות המבנה שלנו מצביע למינימום (גישה אליו מיידית), מדובר בטיול מהמינימום כלפי מעלה בלבד עד אשר מוצאים את הצומת המתאים, שהוא במקרה הכי גרוע השורש ולכן הסיבוכיות תוך התקדמות בשדרת האבות מהמינימום לשורש העץ הינה כפי שלמדנו.
* – מבצעת את פעולת כפי שנלמד בכיתה. משתמש בפונקציית ולכן סיבוכיותה הינה כדי למצוא את האיבר ה בדרגתו.
* – פונקציה אשר מקבלת צומת. אם יש לו בן ימני היא תחזיר את הצומת הקטן ביותר של תת עץ ימני זה. אחרת, היא תעלה עד השורש עד שנעלה בפעם הראשונה מצד ימין ונחזיר צומת זה. אם מדובר באיבר הכי גדול נחזיר . סיבוכיותה כעומק העץ, שכן במקרה הגרוע אנו נטייל מעלה כלשהו ועד שורש העץ.
* – בדומה לעוקב, מציאת האיבר הקודם לצומת נוכחי בסיבוכיות .
* - פונקציה איטרטיבית אשר מטיילת משורש העץ ובכל פעם מתקדמת ימינה עד שהיא מגיעה לעלה שהוא הצומת בעל המיקום המקסימלי. סיבוכיותה כעומק העץ .
* - פונקציה איטרטיבית אשר מטיילת משורש העץ ובכל פעם מתקדמת שמאלה עד שהיא מגיעה לעלה שהוא הצומת בעל המיקום המינימלי. סיבוכיותה כעומק העץ .
* – פונקציה אשר מבצעת סיבוב ימינה כפי שנלמד בהרצאה בהינתן שני צמתים, אב ובן. מכיוון שמדובר במספר סופי של מצביעים הרי שהסיבוכיות הינה .
* –פונקציה אשר מבצעת סיבוב שמאלה כפי שנלמד בהרצאה בהינתן שני צמתים, אב ובן. מכיוון שמדובר במספר סופי של מצביעים הרי שהסיבוכיות הינה .
* – פונקציה אשר מתקנת את ה של צומת בהתאם לנלמד בכיתה, בחלוקה למקרים בהתאם לנלמד. משתמשת במתודות במידת הצורך. מכיוון שמדובר במספר סופי של מצביעים הרי שהסיבוכיות הינה .
* – פונקציה אשר מטיילת מהצומת שהוכנסה לעץ ועד השורש ומעדכנת במהלך הטיול כלפי מעלה את שדות . תעצור כאשר תמצא שאין הפרה של תנאי ה לעץ . מכיוון שיש פוטנציאל לטיול מהעלה ועד השורש המקרה הגרוע הינו . התיקון של המקרים המפרים את ה מתוקנים על-ידי שימוש במתודת .
* – בהתאם לכיוון הרצוי אנו נכניס את הצומת בתור עלה ונעדכן את השדות המתאימים. מכיוון שמדובר במספר סופי של מצביעים הרי שהסיבוכיות הינה . נבחין כי איננו מעדכנים את שדות במתודה זו, אלו מעודכנים במתודות אחרות.
* – עדכון שדה לכל האיברים מצומת מסוים ועד השורש. טיול שכזה עלול לקחת וסיבוכיות זאת בהתאם.
* – מתודה אשר מחזירה את גובה העץ, בהתאם לשדה של השורש, ו-1 אם מדובר בעץ ריק. גישה לשדה הינה ב.
* – מתודה אשר מבצעת את פעולת לערך שהוזן כקלט בתור האיבר האחרון ברשימה. בהתאם לסיבוכיות , .
* – מתודה אשר מטיילת מעלה מצומת שנמחק או הצומת המחבר בין שני עצים ומבצעת תיקוני AVL (כלומר סיבובים) לתתי עצים שאינם תקינים במעלה שדרת העץ. לכל צומת במעלה הדרך מתקנת את שדות במידת הצורך ומחזירה את מספר פעולות האיזון שנתבצעו. הסיבוכיות היא שזה סכום האורכים במידה ומדובר בפעולת ופשוט גודל העץ אם מדובר ב.
* – מאחדת שני עצים כפי שנלמדת בכיתה באמצעות צומת מקשר. תחילה נבדוק אם אחד העצים ריק, אם כן פשוט נכניס את האיבר המחבר. לאחר מכן נבדוק לאיזה עץ יש גובה גבוה יותר ונבצע טיול בעץ הגבוה יותר כדי להגיע למצב שיש תת עץ בגובה לפחות של העץ השני, ואז נבצע את האיחוד עם הצומת המחבר. נעדכן את האיברים הרלוונטיים ונשתמש ב לביצוע תיקונים במידת הצורך (כולל המקרה המיוחד שתואר בפורום).   
  סיבוכיות מתודה זו הינה כפי שנלמד בכיתה.
* – מתודה אשר מוחקת עלה מעץ ולאחר מכן משתמשת בפונקציית כדי ולעדכן את שדות של האיברים מהעלה ועד שורש העץ ולאזן במידת הצורך. שימוש במתודה גורר סיבוכיות . מחזירה את מספר פעולות האיזון שנתבצעו במתודה.
* – מתודה אשר מבצעת מחיקה של צומת עם בן יחיד. בודקת איזה בן יש לצומת ומבצעת "מעקף" כפי שנלמד בכיתה לבן של הצומת. לאחר מכן משתמשת בפונקציית החל מהאב של הצומת שנמחק כדי ולעדכן את שדות של האיברים מהעלה ועד שורש העץ ולאזן במידת הצורך. שימוש במתודה גורר סיבוכיות . מחזירה את מספר פעולות האיזון שנתבצעו במתודה.

מתודות עזר אשר אינן תחת מחלקת

* – מימוש אלגוריתם אשר מבצע סידור רנדומי למערך באופן שהוא . אנו רצים בלולאה במשך מספר האיברים ובכל פעם בוחרים רנדומית אינדקס (בטווח שהולך ויורד בכל איטרציה), ואז מבצעים החלפה לזוג איברים. נבחין כי אנו מניחים כי השימוש בפונקציית מספריית הינו לכל היותר קבוע ולכן מדובר בסיבוכיות במקרה הגרוע.
* – מימוש של כפי שנלמד בכיתה, כולל רקורסיבי. כמו כן למדנו בכיתה כי מיון זה הינו במקרה הגרוע .

**בדיקות**

התרגילים ייבדקו באמצעות תוכנת טסטר שקוראת לפונקציות המפורטות מעלה בתרחישים שונים, ומוודאת את נכונות התוצאות. קובץ הטסטר שלנו **לא יפורסם** לפני הבדיקות.

**מומלץ מאוד לממש אוסף בדיקות עבור המימוש**, לא בשביל ההגשה, אלא כדי לבדוק שהקוד לא רק רץ, אלא גם נכון!

בקובץ שתגישו **לא תהיה פונקציית main ולא יהיו הרצות קוד/הדפסות**, דבר זה יפגע בטסטר שיבדוק לכם את התרגילים.

**חלק ניסויי/תיאורטי**

**שאלה 1:**

בשאלה זאת נדון בכמות צעדי האיזון הממוצעת הנדרשת בסדרת הכנסות, בסדרת מחיקות ובסדרה מעורבבת של הכנסות ומחיקות.

* לצורך הניתוח, נבנה עצי AVL בגדלים שונים. מספר איברים שנכניס לעץ יהיה כאשר i=1,…,10. כלומר, עבור i=1 העץ בגודל 3000, ועבור i=10 העץ בגודל כמיליון וחצי.
* לכל גודל של עץ, נבצע 3 ניסויים נפרדים:
  + **נכניס** איברים בסדר אקראי.
  + נכניס איברים בסדר אקראי (לא נספר), ולאחר מכן **נמחק** אותם בסדר אקראי.
  + נכניס n/2 איברים בסדר אקראי, לאחר מכן נבצע n/4 **הכנסות ומחיקות אקראיות** **לסירוגין**.

1. עבור כל ניסוי, יש לציין את מספר פעולות האיזון שנדרשו כדי לתקן את העץ עבור הפעולות **שהודגשו**.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| מספר סידורי i | ניסוי 1 - הכנסות | ניסוי 2 - מחיקות | ניסוי 3 - הכנסות בלבד | ניסוי 3 – הכנסות ומחיקות לסירגון |
| 1 | 8969 | 2351 | 4471 | 2644 |
| 2 | 17967 | 4695 | 8969 | 5252 |
| 3 | 35965 | 9439 | 17967 | 10748 |
| 4 | 71963 | 18803 | 35965 | 21310 |
| 5 | 143961 | 37811 | 71963 | 42395 |
| 6 | 287959 | 75462 | 143961 | 85029 |
| 7 | 575957 | 151018 | 287959 | 171400 |
| 8 | 1151955 | 302131 | 575957 | 340637 |
| 9 | 2303953 | 602720 | 1151955 | 682431 |
| 10 | 4607951 | 1207772 | 2303953 | 1360956 |

1. היעזרו באקסל, איזה ביטוי אסימפטוטי תואם כל עמודה? (במונחים של n)  
   **הדרכה**: במקרה שמצפים לביטוי , ניתן לבדוק את מידת ההתאמה האמפירית של הנתונים על-ידי חילוק הנתונים ב . בתוכנת אקסל, למשל, ניתן לחשב קו-מגמה (trendline) ומדד ה- מעיד על איכות הקירוב.

כל העמודות בעלות ביטוי אסימפטוטי של .

**שאלה 2:**

בניסוי זה נשווה את ביצועי עץ AVL לעומת רשימה מקושרת ומערך.

ממשו רשימה באמצעות רשימה מקושרת ובאמצעות מערך (אין צורך להגיש את המימוש הזה).

עבור כל חזרו על התהליך הבא:

* הכניסו איברים לרשימה הממומשת בעזרת עץ AVL (לפי שלושת המקרים המפורטים בהמשך).
* הכניסו את אותם האיברים לרשימה הממומשת בעזרת רשימה מקושרת ומערך.
* עבור כל אחד מהעצים, מדדו את זמן הריצה הממוצע (הממוצע על פני ההכנסות).

חזרו על התהליך כאשר ההכנסות הן לפי:

1. **הכנסות להתחלה** – סדרת הכנסות לראש הרשימה.
2. **סדרה אקראית** - מכניסים במקומות אקראיים מתוך גודל הרשימה הנוכחי.
3. הכנסות לסוף – סדרת הכנסות לסוף הרשימה

רשמו את התוצאות בטבלאות הבאות:

|  |  |  |  |
| --- | --- | --- | --- |
| זמן ריצה בממוצע  מספר סידורי | עץ AVL  הכנסות להתחלה | רשימה מקושרת  הכנסות להתחלה | מערך הכנסות להתחלה |
| 1 | 0.0719302000943 | 0.00150119990576 | 0.00461480009835 |
| 2 | 0.1420611999928 | 0.00455469998996 | 0.01937870006076 |
| 3 | 0.1690334000159 | 0.01341659994795 | 0.04646209999918 |
| 4 | 0.1752182999625 | 0.01495720003731 | 0.07353569997940 |
| 5 | 0.2431256999261 | 0.01338869996834 | 0.10449330008123 |
| 6 | 0.2321030999301 | 0.02425469993613 | 0.12620890000835 |
| 7 | 0.4628970000194 | 0.03011670005973 | 0.17885989998467 |
| 8 | 0.4601383000845 | 0.02347999997437 | 0.24970769998617 |
| 9 | 0.4247344000032 | 0.01865750004071 | 0.31897650007158 |
| 10 | 0.4737175999907 | 0.03779059997759 | 0.37538049998693 |

|  |  |  |  |
| --- | --- | --- | --- |
| זמן ריצה בממוצע  מספר סידורי | עץ AVL  הכנסות אקראיות | רשימה מקושרת הכנסות אקראיות | מערך הכנסות אקראיות |
| 1 | 0.088270199950 | 0.12506780005060 | 0.08668860001489 |
| 2 | 0.182253699982 | 0.43650770001113 | 0.36609270004555 |
| 3 | 0.197465099976 | 1.13095019990578 | 0.78395069995895 |
| 4 | 0.270113600068 | 1.22214540001004 | 1.84673170000314 |
| 5 | 0.376800099969 | 1.87511729996185 | 2.22196080000139 |
| 6 | 0.381999799981 | 3.27135920000728 | 2.94231159996707 |
| 7 | 0.434939900063 | 3.87575119989924 | 3.61792579991742 |
| 8 | 0.550053799990 | 5.86719140002969 | 5.85327860002871 |
| 9 | 0.571138200000 | 6.95263429998885 | 7.94336779997684 |
| 10 | 0.612739400006 | 11.6008293000049 | 10.4729150999337 |

|  |  |  |  |
| --- | --- | --- | --- |
| זמן ריצה בממוצע  מספר סידורי | עץ AVL  הכנסות בסוף | רשימה מקושרת הכנסות בסוף | מערך הכנסות בסוף |
| 1 | 0.04397960007 | 0.00098370003979 | 0.00649770000018 |
| 2 | 0.10533829999 | 0.00467669998761 | 0.01918740000110 |
| 3 | 0.14144800009 | 0.00630010000895 | 0.05228589999023 |
| 4 | 0.22471300000 | 0.00769720005337 | 0.13386649999301 |
| 5 | 0.55970950005 | 0.01382789993658 | 0.14232189999893 |
| 6 | 0.50825630000 | 0.00993960001505 | 0.22519739996641 |
| 7 | 0.47159900004 | 0.04356629995163 | 0.28805390000343 |
| 8 | 0.51200190000 | 0.02177240001037 | 0.37499919999390 |
| 9 | 0.51144400006 | 0.03695460001472 | 0.57516700006090 |
| 10 | 0.55193129996 | 0.02005190006457 | 0.54191740008536 |

מה הייתם מצפים שתהיינה התוצאות, והאם התוצאות האמיתיות מסתדרות עם ציפייה זו? הסבירו.

עבור הכנסה סדורה (כלומר להכנסה בלבד או לסוף בלבד) נצפה שAVL יהיה הכי פחות טוב שכן אנו מכניסים תמיד איברים לתת עץ אחד (ובאופן ספציפי את האיבר המינימלי והמקסימלי בהתאמה) ולכן עלינו לבצע מספר פעולות איזון לעץ בעוד שגם עבור מערך יש לנו יכולת שרשור פשוטה אשר לעיתים תיצור העתקות של המערך שכן עליו להישאר באופן רציף, אך אנו מניחים כי פייתון עושים את מלא האופטמיזציות לתהליך זה. כמו כן עבור רשימה מקושרת עם מצביע לסוף מדובר במספר סופי של שינוי מצביעים, ללא "טיולים" ברשימה ולכן נצפה שתהליך זה יהיה מהיר יותר מאשר של עץ ואף משל מערך.

עבור הכנסות רנדומיות נצפה כי AVL יהיה הטוב ביותר שעליו רק לאזן את העץ בכל הכנסה ועל כן נעריך כי מכיוון שההכנסות תהיינה באופן רנדומי, סדר ההכנסות יהיה כזה אשר לעיתים יהיה לתת עץ ימין ולעיתים לתת עץ שמאל ולכן מספר פעולות האיזון הנדרשות יפחת ביחס להכנסה לתת עץ אחד. כמו כן, מכיוון שאין צורך בטיול ברשימה או בשמירה על רציפות בזיכרון אלא לביצוע במקרה הגרוע ביותר פעולות בכל הכנסה הרי שנצפה לביצועים טובים במקרה רנדומי.

נבחין כי עבור רשימה מקושרת עם מצביע לסוף נצפה להכנסות רנדומיות במקרה הגרוע ביותר כי על אף שמדובר בעת הכנסה על שינוי מספר סופי של מצביעים, עלינו לבצע "טיול" ברשימה המקושרת כדי להגיע ליעד. במקרה הגרוע ביותר מדובר ב שזה בעצם . נבחין כי עבור המערך עלינו לבצע העתקות חדשות בכל הכנסה ליצירת שרשורים מתאימים, תהליך שגם עלול לעלות ולכן נעריך כי ביצועים אלו יהיו גרועים, אך מכיוון שמדובר ביכולת מדף של פייתון היא ככל הנראה תהיה טובה יותר מאשר זו של הרשימה המקושרת.

תוצאות האמת מסתדרות עם ציפיות אלו.

**הוראות הגשה**

הגשת התרגיל תתבצע באופן אלקטרוני באתר הקורס במודל.

**הגשת התרגיל היא בזוגות בלבד!**

כל זוג יבחר **נציג/ה** ויעלה **רק** תחת שם המשתמש של הנציג/ה את קבצי התרגיל (תחת קובץ zip) למודל.

על ההגשה לכלול שלושה קבצים:

1. קובץ המקור (הרחבה של קובץ השלד שניתן) תחת השם AVLTreeList.py.
2. קובץ טקסט info.txt המכיל את פרטי הזוג: מספר ת"ז, שמות, ושמות משתמש.
3. מסמך תיעוד חיצוני, המכיל גם את תוצאות המדידות. את המסמך יש להגיש באחד הפורמטים הבאים: doc, docx או pdf.

שמות קובץ התיעוד וקובץ הzip צריכים לכלול את שמות המשתמש האוניברסיטאיים של **הזוג המגיש** לפי הפורמט AVLTreeList\_username1\_username2.pdf/doc/zip/…. בתוכן הקבצים יש לציין את שמות המשתמש, תעודות הזהות ושמות המגישים (בכותרת המסמך ובשורת הערה בקובץ המקור).

הגשת שיעורי הבית באיחור - באישור מראש בלבד. הגשה באיחור ללא אישור תגרור הורדת נקודות מהציון.

**הגשת התרגיל היא חובה לשם קבלת ציון בקורס**.

**בהצלחה!**