עומר יצחק, ת"ז : 205656986, omeritzhak

סער ג'רסי ת"ז : 313221210, saargerassi

תיעוד פרוייקט:

המחלקה שמומשה היא AVLTree. למחלקה יש מספר שדות. וביניהם :

- שורש העץ root
- ערך המינימום - min
- ערך המקסימום -max
- מספר הצמתים בעץ - size

במחלקה מספר פונקציות שנדרשנו לממש והן :

1.()empty- הפונקציה מחזירה true אם העץ הוא עץ ריק. ופועלת בסיבוכיות זמן O(1).

2.(search(int k - הפונקציה מחפשת איבר בעל מפתח k .אם קיים הפונקציה מחזירה את ערך הצומת. אחרת מחזירה null. הפונקציה פועלת בסיבוכיות זמן של O(n).

3.(insert (int k, String i- הפונקציה מכניסה איבר בעל ערך i ומפתח k לעץ, אם המפתח לא קיים. הפונקציה מחזירה int כמספר פעולות האיזון שבוצעו. אם המפתח קיים הפונקציה תחזיר 1- .

הפונקציה פועלת בסיבוכיות זמן(O(logn וקוראת למספר פונקציות נוספות :

- ()empty
- search
- search_node_insert- פונקציה זו פועלת בסיבוכיות זמן (O(logn ומחזירה את האבא של הצומת שבה נרצה להכניס את הצומת החדש.
- get_min_key- מעדכן את ערך המינימום בעץ. בסיבוכיות זמן (O(logn.
- get_max_key- מעדכן את ערך המקסימום בעץ, בסיבוכיות זמן (O(logn.
- compute_BF - מחשב את ערך ה BF של צומת. פועל בסיבוכיות זמן (O(1.
- rebalance- מחזיר את סוג האיזון שיש לבצע. פועל בסיבוכיות זמן (O(1.
- rotation- מבצעת את פעולת האיזון בפועל. וקוראת לפונקציית העזר left_rotationאו right_rotation לפי סוג האיזון, וכולן פועלות בסיבוכיות זמן (O(1.

4. (delete (int k- הפונקציה מוחקת איבר בעל מפתח k, אם המפתח קיים. הפונקציה מחזירה int כמספר פעולות האיזון שבוצעו. אם המפתח לא קיים הפונקציה תחזיר 1- .

הפונקציה פועלת בסיבוכיות זמן O(logn) וקוראת למספר פונקציות נוספות:

•empty()

•search

•search_node_delete- פונקציה זו פועלת בסיבוכיות זמן O(logn) ומחזירה את הצומת שנרה למחוק.

•Delete_root- אם הצומת שנרצה למחוק הוא השורש. ואין לו שני בנים, אז נשתמש בפונקציה זו, כדי למחוק את השורש. פועלת בסיבוכיות זמן O(1).

•Delete_node- מוחק את הצומת. במקרה שלצומת יש שני ילדים. הפונקציה תקרא לפונקצית עזר שנקראת successor שמוצאת את האיבר העוקב, בסיבוכיות זמן O(logn).

•get_min_key- מעדכן את ערך המינימום בעץ. בסיבוכיות זמן O(logn).

•get_max_key- מעדכן את ערך המקסימום בעץ, בסיבוכיות זמן O(logn).

•compute_BF - מחשב את ערך ה BF של צומת. פועל בסיבוכיות זמן O(1).

•rebalance- מחזיר את סוג האיזון שיש לבצע. פועל בסיבוכיות זמן O(1).

•rotation- מבצעת את פעולת האיזון בפועל. וקוראת לפונקציית העזר left_rotationאו right_rotation לפי סוג האיזון, וכולן פועלות בסיבוכיות זמן O(1).

5. Min() – הפונקציה מחזירה את הערך של הצומת בעל המפתח המינימלי ב O(1).

6.Max() – הפונקציה מחזירה את הערך של הצומת בעל המפתח המקסימלי ב O(1).

7.keysToArray - הפונקציה מחזירה מערך ממוין המכיל את כל המפתחות בעץ. אם העץ ריק הפונקציה מחזירה מערך ריק. הפונקציה פועלת בסיבוכיות זמן O(n). הפונקציה קוראת לפונקציה keysToArray_rec, פונקציה רקורסיבית שמכניסה את האיברים למערך באופן ממוין.

8.infoToArray- הפונקציה מחזירה מערך ממוין המכיל את כל הערכים בעץ. אם העץ ריק הפונקציה מחזירה מערך ריק. הפונקציה פועלת בסיבוכיות זמן O(n). הפונקציה קוראת לפונקציה infoToArray_rec, פונקציה רקורסיבית שמכניסה את האיברים למערך באופן ממוין.

9. Size- הפונקציה מחזירה את מספר האיברים בעץ ב O(1).

10.GetRoot – הפונקציה מחזירה את השורש של העץ בסיבוכיות O(1).

11. Join() – הפונקציה מקבלת צומת, ותת עץ כך שתת העץ מכיל מפתחות גדולים יותר מהצומת, ומחברת את תת העץ לעץ הנוכחי, עם הצומת x. הפונקציה פועלת בסיבוכיות o(logn). בנוסף לפעולת האיחוד, הפונקציה מחזירה את סיבוכיות הפעולה.

הפונקציה משתמשת במספר פונקציות עזר, אשר יפורטו להלן:

- WhereToJoin- הפונקציה מקבלת שורש, וצומת ובודקת היכן יתבצע איחוד העצים ביחס לצומת הנתונה.
- SizeAfterJoin- הפונקציה מקבלת צומת ואת המספר 1, ומעדכנת את שדה size של כל הצמתים הנמצאים מעל x בעץ החדש.

- FixJoinRotation- הפונקציה מקבלת שורש של עץ ומבצעת עליו את כל פעולות האיזון לאחר איחוד העצים, על מנת להפוך אותו לעץ תקין.

- whichCase- הפונקציה מקבלת עץ AVL והפרש גבהים בין העץ הנוכחי לבין העץ החדש אותו אנו רוצים לאחד, ומחזירה מחרוזת אשר מייצגת את סוג האיחוד אותו נרצה לבצע, ביחס לאיזה עץ גבוהה יותר, והאם האיחוד יתבצע מימין או משמאל, כתלות בערכי המפתחות ביחס ל x.

- equalJoin- הפונקציה מאחדת את העצים במידה והם בעלי גובה זהה.

- leftJoin- הפונקציה מצרפת את העץ הקטן יותר ובעל המפתחות הקטנים יותר מצד שמאל.

- hightJoin- הפונקציה מצרפת את העץ הקטן יותר ובעל המפתחות הגדולים יותר מצד ימין.

- changePointers- הפונקציה מקבלת ארבע צמתים, ומשנה את המצביעים בין הצמתים ואת המצביעים ביניהם. כלומר, הפונקציה משנה את ההירכיה בין הצמתים ע"י שינוי של המצבעים בלבד.

- getRankl- הפונקציה מקבלת שני צמתים ודרגה, ומחזירה את הצומת הראשונה בשדרה השמאלית של תת העץ, אשר הגובה שלה קטן או שווה מהדרגה שקיבלנו כקלט. (בנוסף הפונקציה משנה מצביע לצומת השניה שקיבלנו כקלט)

- getRankr- הפונקציה מקבלת שני צמתים ודרגה, ומחזירה את הצומת הראשונה בשדרה הימנית של תת העץ, אשר הגובה שלה קטן או שווה מהדרגה שקיבלנו כקלט. (בנוסף הפונקציה משנה מצביע לצומת השניה שקיבלנו כקלט)

12.Split- הפונקציה מקבלת מפתח של צומת עבורו נרצה לפצל את העץ הקיים. ומחזיקה מערך בגודל 2 שמכיל את 2 תתי העצים שקיבלנו לאחר הפיצול. העץ הראשון בעל מפתחות קטנים ממש מ k והעץ השני בעל מפתחות גדולים ממש מ k. הפונקציה קוראת למספר פונקציות עזר:

- WhereToJoin- הפונקציה מקבלת שורש, וצומת ובודקת היכן יתבצע איחוד העצים ביחס לצומת הנתונה.

- Join- כפי שמפורט מעל.

- getTempMin- הפונקציה מקבלת צומת, ומחזירה את המינימום בתת העץ שהצומת הינה השורש שלו.

- getTempMax- הפונקציה מקבלת צומת, ומחזירה את המקסימום בתת העץ שהצומת הינה השורש שלו.

חלק ניסוי/תיאורטי –

שאלה 1-

מספר סידורי i	מספר חילופים במערך ממוין - הפוך	עלות החיפושים במיון AVLעבור מערך ממוין-הפוך	מספר חילופים במערך מסודר אקראית	עלות החיפושים במיון AVL עבור מערך מסודר אקראי
1	1999000	38884	972240	32885
2	7998000	85764	3978442	71704
3	31996000	187524	15955458	157653
4	127992000	407044	64239897	359652
5	511984000	878084	257160971	788593

ב. במקרה בו הרשימה ממוינת הפוך אנו נמצאים במקרה הגרוע, ולכן סיבוכיות הזמן תחושב באופן הבא:

מכיוון שהאיברים ממוינים הפוך, בכל פעם ניקח את האיבר ונשים אותו בראש הרשימה זה אומר שבאיטרציה הראשונה נבצע חילוף אחד ולאחר מכן 2 חילופים וכן הלאה עד n.
ולכן סיבוכיות הזמן היא סדרה חשבונית:

$$1 + 2 + 3 \ldots + n = \frac{(n+1)n}{2} = \frac{n^2}{2} + \frac{n}{2} = \theta(n^2)$$

בנוסף לכך, זמן הכולל של פעולות החיפוש חסום ע"י O(nlogn) ונראה זאת כעט:

אנו מצביעים לאיבר המקסימלי כדרוש בשאלה ובכל הכנסה מצבעים חיפוש למיקום ההכנסה, כל חיפוש חסום ע"י $2logn$

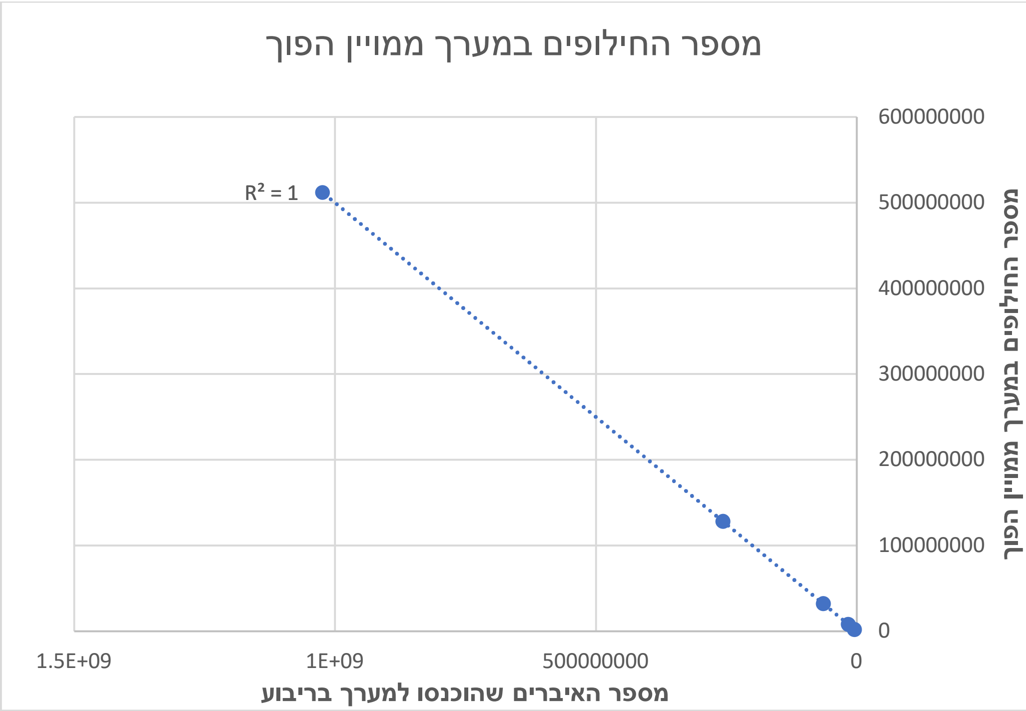
מכיוון שאנו נמצאים במקרה הגרוע ואחרי $\frac{n}{2}$ הכנסות נכניס בכל פעם את האיבר המינימלי אז נעלה עד השורש ואז נרד עד בצד השמאלי ביותר:

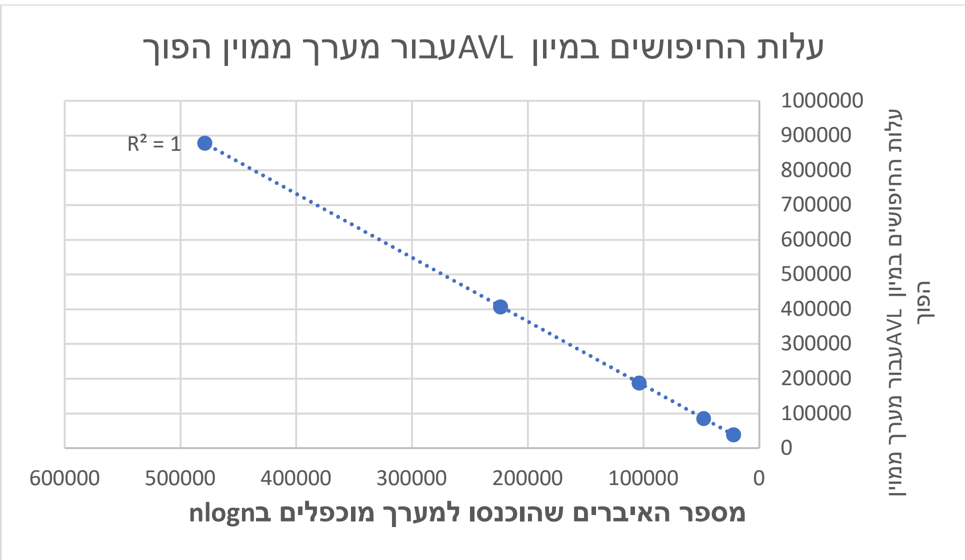
$$2\log\frac{n}{2} + 2\log\left(\frac{n}{2}+1\right) + \cdots + 2\log(n) = 2\frac{\left(log\frac{n}{2} + \log(n)\right)\frac{n}{2}}{2} = \theta(nlogn)$$

חסום מלמטה וגם מלמעלה מכיוון שמלמעלה נוכל לחסום את הסיבוכיות אם נכפול את המשוואה פי 2, ומלמטה זה חסום ע״י המשוואה שלעיל מכיוון שאנו מתחילים אותה אחרי $\frac{n}{2}$ הכנסות.

ג. אחרי התאמת ציר ה-x להיות x^2 ו xlogx אלו הגרפים שהתקבלו :

הערה: ציפינו לקבל R^2 שקרוב מאד ל 1(גדול מ 0.95), בפועל קיבלנו R^2= 1 . הדבר נובע מקירוב של תוכנת אקסל, שכן הקשר הלינארי היה הדוק מאד. (בפועל איננו 1 בידיוק).





הגרפים אכן משקפים את מה שציפינו לקבל.

עבור מיון ע"י insertion sort , אנו יודעים שהוא נעשה ב O(n^2). לכן כאשר ביצענו טרנספורמציה על ציר ה x' כך שערכיו יהיו בריבוע. קיבלנו קשר לינארי הדוק מאד. ניתן לראות שביחס לקו המגה המצופה לקשר לינארי, ערכי הגרף שקיבלנו ממש נמצאים על קו המגמה. כלומר, הסיבוכיות היא אכן $O(n^2)$

עבור מיון ע"י עץ AVL, ציפינו שהמיון יעלה O(nlogn).לאחר שביצענו נרמול של צריך ה X להציג את הערכים שהוכנסו לעץ כ nlogn , מצאנו שהדבר עומד בקנה אחד עם הממצאים בגרף. משום שביחס לקו המגמה הלינארי, הערכים מקיימים קשר לינארי הדוק, כלומר הסיבוכיות היא אכן $O(nlogn)$.

ד. במיון insertion sort המבוסס על עץ AVL כמות החילופים h היא סך החילופים שבוצעו עבור הכנסה של n איברים לעץ. כלומר מתקיים : $\sum_{i=0}^{n} hi$ כאשר hi הוא כמות החילופים שבוצעו בהכנסה של האיבר ה i. אנו יודעים שכמות ההחלפות הדרושה היא כמספר האיברים במערך עד למיקום ה i שגדולים מהערך במקום ה i. כפי שראינו בהרצאה, ע"י שימוש ב finger tree , עלות החיפוש של האיבר ה i תיהיה O(loghi) כלומר תלויה בכמות הערכים שגדולים ממנו. ולכן נקבל שסך עלות החיפושים תיהיה :

$$\sum_{i=0}^{n} loghi \leq \sum_{i=0}^{n} logh = O(nlogh)$$

ה. בסעיף ד' נתבקשנו לחשב חסם עליון בלבד ולא חסם : θ , מכיוון , שהחסם התחתון, הוא o של החסם העליון, ולכן אין חסם טטא שנוכל למצוא.

שאלה 2-

א.

מספר סידורי i	עלות join ממוצע עבור split אקראי	עלות join מקסימלי עבור splitאקראי	עלות join ממוצע עבור split של האיבר מקסימלי בתת העץ השמאלי	עלות join מקסימלי עבור splitשל איבר מקסימלי בתת העץ השמאלי
1	2.63	5	2.6	13
2	2.7	6	2.64	14
3	2.64	7	2.67	15
4	2.68	7	2.5	17

5	2.44	8	2.79	17
6	2.66	6	2.79	19
7	2.99	7	2.59	20
8	2.44	9	2.5	21
9	2.4	7	2.82	22
10	2.71	8	2.74	23

ב. עלות join ממוצע עבור שני המקרים :

* עבור split על האיבר המקסימלי בתת עץ השמאלי -

עבור עץ שגובהו h כאשר נבצע split על האיבר המקס' בתת העץ השמאלי – נצפה שיתבצעו כ h-פעולותjoin . ראשית נצפה לרצף פעולות שעלותן היא קטנה, 2 לכל היותר, מכיוון שההפרש ה Rank-של העצים שנחבר יהיה נמוך. לאחר רצף הפעולות הזולות תתבצע פעולה אחת יקרה שעלותה כ h-והיא תתבצע עבור ה join-שמחבר את עץ הגדול יותר המורכב מתת העץ הימני של האיבר עליו ביצענו את הsplit – ולכן עץ ריק כי הבן הימני שלו הוא עלה חיצוני, השורש ותת העץ הימני של השורש. עבור הממוצע נצפה לקבל: (נסמן כי כל פעולה זולה תעלה 1.5)

$$\frac{1.5 + 1.5 + \cdots + 1.5 + h}{h} = \frac{1.5h + h}{h} = 2.5$$

יצא בערך 2.5 שזו תוצאה שמתיישבת עם הערכים בטבלה.

עבור split אקראי –

לא נוכל לדעת בוודאות איזה צומת ומאיזה עומק הצומת יהיה, אך מבחינה הסתברותית נצפה לקבל צומת ברמה התחתונה יותר מכיוון שאם העץ מאוזן ברמה התחתונה ביותר + אחת מעליה יש יותר מחצי מכמות הצמתים (הוכחנו זאת בהרצאה) ולכן נקבל בהסתברות גבוהה יותר שנעשה את הsplit על עלה.

ולכן, נצפה שכמות הjoin שנבצע יהיה מסדר גודל של logn.

עלות ה join-הממוצע מושפעת מעומקו של הצומת הנבחר. הראנו כי בהסתברות גבוהה יבחר צומת מהרמות הנמוכות ובנוסף ראינו בהרצאה כי עלות פעולת split היא. O(logn) כמות פעולות ה join-שיבוצעו היא כעומק הצומת בעץ כלומר סדר גודל של logn ולכן נקבל שהעלות הממוצעת היא O(1)

ניתן לראות כי ניתוח זה מתיישב עם הנתונים בטבלה.

ג. join מקסימלי בתרחיש split על איבר מקסימלי בתת העץ השמאלי :

ההתחלה זה אותו הסבר מסעיף ב'.

עבור עץ שגובהו h כאשר נבצע split על האיבר המקס' בתת העץ השמאלי – נצפה שיתבצעו כ h-פעולותjoin . ראשית נצפה לרצף פעולות שעלותן היא קטנה, 2 לכל היותר, מכיוון שההפרש ה Rank-של העצים שנחבר יהיה נמוך. לאחר רצף הפעולות הזולות תתבצע פעולה אחת יקרה שעלותה כ h-והיא תתבצע עבור ה join-שמחבר

את עץ הגדול יותר המורכב מתת העץ הימני של האיבר עליו ביצענו את הsplit – ולכן עץ ריק כי הבן הימני שלו הוא עלה חיצוני, השורש ותת העץ הימני של השורש.

ולכן נצפה לקבל שהעלות המקסימלי תהיה בגודל h שh חסום מלעיל ע"י 1.44logn ניתוח זה מתיישב בצורה מדויקת עם הערכים שיצאו בטבלה.

ד. הסיכוי שייצא צומת ברמה הi הוא $\frac{2^i}{n}$ כאשר i מתחיל מ -0 עד h גובה העץ.
עלות הjoin המקסימלית תהיה העומק של הצומת שעליה מפצלים ולכן :

$$\sum_{i=0}^{logn}(i+1)*\frac{2^i}{n} \geq \sum_{\frac{logn}{2}}^{logn}(i)*\frac{2^i}{n} \geq \left(\frac{logn}{2}+\frac{logn}{2}+\cdots+\frac{logn}{2}\right)=O(logn)$$

ולכן תוחל פעולת הjoin המקסימלית תיהיה o(logn).