**Code documentation**

**AVLNode Class**

* **Is\_real\_node**

**מימוש של בדיקה האם צומת הוא אמיתי או וריטואלי על-פי האם ילדיו הם None והאם גובהו -1.**

**AVLTree Class**

* **\_init\_**

**הוספת אתחול virtual\_node בעת אתחול העץ**

* **Search**

**מתחיל מהשורש ובכל איטרציה יורד בגובה לכיוון העלים על-פי ערכי הבנים ומעדכן את כמות הקשתות שעברנו לבסוף מחזיר את הצומת הרלוונטי (או none) ואת אורך מסלול החיפוש.**

**מכייון שזהו עץ AVL הגובה יהיה מ-logn. בכל גובה ביצענו פעולות לינאריות לכן זמן הריצה של הפונקציה יהיה O(logn).**

* **Finger\_search**

**מתחיל מהמקסימום ומתקדם לכיוון השורש עד שמוצא צומת משותף עם הערך שאנחנו מחפשים.**

**לאחר מכן מבצע לוגיקה דומה ל-search רק עד נקודת התחלה שונה (במקום השורש מתחיל מהצומת המשותף).**

**בהתאם, כאשר k גודל תת העץ הרלוונטי יתקיים כי זמן הריצה של הפונקציה הוא O(logk).**

* **Insert**

**תחילה קורא לפונקציה simple insert.**

**לאחר מכן שומר את המסלול המקורי של ההכנסה על-מנת שנוכל לבדוק את כמות שינויי הגובה.**

**מעדכן את הגבהים של מסלול ההכנסה בעזרת קריאה לפונקציה Set\_heights\_from\_node\_up**

**שומר את הגבהים לאחר עדכון הגבהים ומעדכן את ערכי ה-size ו-max.**

**קורא לפונקציית האיזון – tree\_balancer.**

**ולבסוף משווה את ערכי גבהי מסלול ההכנסה לפני ואחרי הפעולה ומחזיר את ערך ה-h הנדרש בפונקציה.**

**מכיוון שזהו עץ avl אורך המסלול יהיה logn ולכן החישוב של h יבוצע ב-logn.**

**מעבר לכך, כל פעולות העזר שהפונקציה קוראת להן מבוצעות ב-logn ולכן זמן הריצה הכולל של הפונקציה יהיה O(logn).**

* **Finger\_insert**

**מימוש דומה ל-insert רק שבמקום להגיע עד השורש מבצע את כל הפעולות עד הצומת המשותף בין הערך המוכנס לבין המקסימום.**

**בהתאם, כאשר k גודל תת העץ הרלוונטי יתקיים כי זמן הריצה של הפונקציה הוא O(logk).**

* **Delete**

**הפונקציה מוחקת צומת בהינתן מצביע לצומת בעץ.**

**המימוש קורא קודם ל־simple\_delete שמבצע את הניתוק/החלפה בפועל ומחזיר את הצומת שממנו צריך להתחיל איזון (או את ההורה הרלוונטי). לאחר מכן קוראים ל־** **tree\_balancerכדי לתקן הפרות של תנאי AVL לאורך המסלול עד השורש. לבסוף מעדכנים את size של העץ, ובמידת הצורך מחשבים מחדש את max\_node.**

**simple\_delete עולה O(logn)במקרה הגרוע (חיפוש יורש/עדכון מצביעים ועדכון גבהים לאורך המסלול לשורש) ו-tree\_balancer גם O(logn) ולכן נקבל סה״כ עלות שלO(logn) .**

* **Join**הפונקציה מאחדת שני עצי AVL בעזרת צומת מפריד (key, val) כך שכל המפתחות בעץ אחד קטנים ממנו וכל המפתחות בעץ השני גדולים ממנו. אם הפרש הגבהים קטן (≤1) מחברים מיד והצומת החדש נהיה השורש. אחרת יורדים בעץ הגבוה לאורך השדרה המתאימה עד נקודה בגובה מתאים, משבצים שם את הצומת המפריד, ואז מעדכנים גבהים ומאזנים עם tree\_balancer. נקבל שזמן הריצה הוא כזמן הירידה לאורך הפרש הגבהים כולל איזון עד השורש, כלומר O(logn).
* **Split**הפונקציה מפצלת את העץ לשני עצים: אחד עם כל המפתחות הקטנים מ-node.key ואחד עם כל הגדולים ממנו. מתחילים מ-node.left כעץ שמאלי ומ-node.right כעץ ימני, ואז עולים מהצומת node עד השורש. בכל עלייה מוסיפים את ההורה ואת תת־העץ “האח” לצד המתאים בעזרת קריאה לפונקציה join לשמאלי או לימני לפי האם עלינו מבן שמאל/ימין.

**מבחינת סיבוכיות זמן ריצה נשים לב כי ראשית כל מטפסים לאורך גובה העץ – O(logn). בכל צעד מבצעים join/insert על עצים שגובהם** קטן/יורד **והעלות הכוללת של כל פעולות הjoin לאורך המסלול חסומה על ידי O(logn). לכן נקבל סה״כ עלות זמן ריצה של O(logn).**

* **Avl\_to\_array  
  הפונקציה** מחזירה רשימה ממוינת לפי מפתחות של זוגות (key, value). המימוש מבצע מעבר in-order בעזרת פונקציית עזר רקורסיבית \_create\_in\_order\_list נכנסים שמאלה, מוסיפים את הצומת הנוכחי לרשימה, ואז נכנסים ימינה. בגלל ש־AVL הוא עץ חיפוש בינארי, סדר זה מחזיר מפתחות בסדר עולה. **מכיוון ש**מבקרים בכל אחד מ-n הצמתים פעם אחת ומבצעים פעולה O(1) לכל צומת נקבל סה״כ עלות ריצה של O(n).
* **Max\_node**

**מחזיר את הצומת המקסימלי בעץ. מבוצע ב-O(1).**

* **Size**

**מחזיר את גודל העץ. מבוצע ב-O(1).**

* **Get\_root**

**מחזיר את שורש העץ. מבוצע ב-O(1).**

**פונקציות עזר שמימשנו במחלקה –**

* **Tree\_balancer**

**מקבל עץ ומוודא שהוא עומד בתנאי האיזון של עץ AVL.**

**משמש גם בהכנסה וגם במחיקה.**

**מתחיל מהצומת שהוכנס ומתקדם ממנו עד לשורש לבדוק האם יש צומת עם balance factor לא תקין. במידה ויש כזה, קורה לאחת מארבע הרוטציות המתאימות. בהתאם למקרה.**

**מכיוון שמבוצעת לכל היותר איטרציה אחת בכל צומת ולכל היותר קריאה אחת לפונקציית סיבוב נסיק מסיבוכיות פונקציות הסיבוב ומכך שבעץ AVL הגובה יהיה מ-logn שסיבוכיות זמן הריצה של הפונקציה תהיה O(logn).**

* **Set\_heights\_from\_node\_up**

**מקבל צומת ומתקף ממנו עד השורש את גבהי הצמתים.**

**הבדיקה של הגובה בכל צומת מבוצעת ב-O(1) ולכל היותר נעבור בצומת אחד בכל גובה עד ההגעה לשורש. מכיוון שזה עץ AVL הגובה יהיה מ-logn. בהתאם, סה״כ הסיבוכיות של הפונקציה תהיה O(logn).**

* **Simple\_insert**

**מקבל צומת התחלתי ובודק ממנו לכיוון העלים עד שמוצא את הנקודה הנכונה להכניס את הערך, מחזיר את הצומת החדש.**

**לכל היותר עובר בצומת אחד בכל גובה של העץ. ובכל צומת מבצע פעולות ב-O(1).**

**בהתאם לכך ושזהו עץ שהך תקין הסיבוכיות תהיה O(logn).**

* **Simple\_delete**

**הפונקציה simple\_delete מבצעת “מחיקה מבנית” בלי לאזן את העץ (האיזון נעשה ב־delete). היא מטפלת ב־4 מקרים:**

1. **עלה - מנתקים מההורה ומחברים במקומו צומת וירטואלי (או אם זה השורש העץ נהיה ריק).**
2. **בן יחיד ימני – מחברים את הבן הימני במקום הצומת ומעדכנים הורה/שורש.**
3. **בן יחיד שמאלי – מחברים את הבן השמאלי במקום הצומת ומעדכנים הורה/שורש.**
4. **שני בנים – מוצאים את ה־successor (הצומת השמאלי ביותר בתת־העץ הימני), “מעלים” אותו למקום הצומת שנמחק, ומחברים אליו את שני תתי־העצים בהתאם, תוך עדכון מצביעי הורים.**

**בסיום קוראים ל־set\_heights\_from\_node\_up החל מצומת מתאים כדי לתקן גבהים עד השורש. מציאת successor היא ירידה לאורך גובה העץ, ועדכון הגבהים הוא עליה לשורש במקרה הגרוע ולכן סה״כ נקבל סיבוכיות O(logn).**

* **Left\_rotation**

**מקבל צומת ומבצע עליו סיבוב ״שמאל״. מבוצע ב-O(1).**

**לאחר ביצוע הסיבוב מתקף את הגבהים מהצומת מעלה בעזרת set\_heights\_from\_node\_up.**

**בהתאם, סה״כ נקבל סיבוכיות O(logn).**

* **Right\_then\_left\_rotation**

**מקבל צומת ומבצע עליו סיבוב ״ימין ואז שמאל״. מבוצע ב-O(1).**

**לאחר ביצוע הסיבוב מתקף את הגבהים מהצומת מעלה בעזרת set\_heights\_from\_node\_up.**

**בהתאם, סה״כ נקבל סיבוכיות O(logn).**

* **Left\_then\_right\_rotation**

**מקבל צומת ומבצע עליו סיבוב ״שמאל ואז ימין״. מבוצע ב-O(1).**

**לאחר ביצוע הסיבוב מתקף את הגבהים מהצומת מעלה בעזרת set\_heights\_from\_node\_up.**

**בהתאם, סה״כ נקבל סיבוכיות O(logn).**

* **Right\_rotation**

**מקבל צומת ומבצע עליו סיבוב ״ימין״. מבוצע ב-O(1).**

**לאחר ביצוע הסיבוב מתקף את הגבהים מהצומת מעלה בעזרת set\_heights\_from\_node\_up.**

**בהתאם, סה״כ נקבל סיבוכיות O(logn).**

חלק ניסויי/תאורטי

בשאלה זו נדון ב insertion-sort באמצעות AVL Finger Tree. המיון מתבצע באופן הבא: מכניסים את האיברים לפי הסדר (הלא ממוין) אל העץ, כאשר החיפוש בהכנסת כל איבר חדש מתחיל מהמקסימום הנוכחי, ובסיום מבצעים סריקת in-order לקבלת הסדר הממוין. עבור עלות בניית העץ, ננתח בנפרד את עלות החיפושים ואת מספר פעולות האיזון.

* לצורך הניתוח, נמיין מערכים בגדלים שונים. גודל המערך שנמיין יהיה כאשר , ואיבריו יהיו הטבעיים עד . למשל, עבור המערך בגודל , ועבור המערך בגודל .
* לכל גודל של מערך, נבצע 4 ניסויים נפרדים:
  + בניסוי הראשון נמיין מערך **ממוין**, מקטן לגדול.
  + בניסוי השני נמיין מערך **ממוין הפוך**, מגדול לקטן.
  + בניסוי השלישי סדר האיברים במערך יהיה **אקראי**.
  + בניסוי הרביעי ניקח מערך ממוין ועבור כל אינדקס פרט לאחרון , נבצע החלפה עם האיבר הבא בסיכוי חצי (שימו לב שייתכן שאיבר יוחלף מספר פעמים).

הערה: בסעיפים הבאים, עבור ניסויים אקראיים, יש לקחת את הממוצע על פני 20 ניסויים.

1. יש למלא בטבלה הבאה את סך עלויות האיזון **ללא גלגולים** עבור כל אחד מהניסויים. הסבירו מהו החסם העליון התאורטי על סך עלויות האיזון כולל גלגולים, והאם הערכים בטבלה מתאימים. לסיום, נמקו מדוע תוספת הגלגולים לספירה אינה משנה אסימפטוטית.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| מספר סידורי i | עלות איזון במערך ממוין | עלות איזון במערך ממוין-הפוך | עלות איזון במערך מסודר אקראית | עלות איזון במערך עם היפוכים סמוכים אקראיים |
| 1 | 1186 | 1186 | 1058 | 1165 |
| 2 | 2385 | 2385 | 2132 | 2359 |
| 3 | 4784 | 4784 | 4284 | 4723 |
| 4 | 9583 | 9583 | 8551 | 9471 |
| 5 | 19182 | 19182 | 17136 | 18941 |
| 6 | 38381 | 38381 | 34273 | 37922 |
| 7 | 76780 | 76780 | 68517 | 75926 |
| 8 | 153579 | 153579 | 137087 | 151827 |
| 9 | 307178 | 307178 | 274197 | 303627 |
| 10 | 614377 | 614377 | 548465 | 607339 |

לפי ניתוח תאורטי של עצי AVL, גובה העץ לאחר n הכנסות הוא O(logn). בכל פעולת הכנסה או מחיקה, תהליך האיזון מטפס לאורך מסלול אחד מהצומת שבו בוצעה הפעולה ועד לשורש, ולכן עלות האיזון לפעולה בודדת חסומה על ידי O(logn). מכאן נובע כי עבור סדרה של n פעולות, החסם העליון התאורטי על סך עלויות האיזון הוא:

גלגולים (rotations) בעצי AVL הם פעולות בזמן קבוע O(1) ומספר הגלגולים בכל פעולת איזון הוא לכל היותר קבוע. לכן, סך מספר הגלגולים לאורך n פעולות הוא לכל היותר O(n). לפיכך, גם כאשר כוללים את עלות הגלגולים בספירה, החסם העליון התאורטי על סך עלויות האיזון נשאר: O(nlogn).

הערכים המופיעים בטבלה תואמים לחסם זה, מאחר והם מציגים קצב גידול התואם סדר גודל של nlog(n) או נמוך ממנו. יש לזכור כי החסם הוא חסם עליון, ולכן ייתכן שבפועל העלויות הנמדדות יהיו קטנות ממנו.

1. בהינתן מערך בגודל נגדיר היפוך בתור זוג אינדקסים כך שמתקיים , ונסמן את מספר ההיפוכים הכולל במערך ב-. ניתן לשים לב כי באופן כללי וככל שיש פחות היפוכים כך המערך קרוב יותר לממוין. יש למלא בטבלה הבאה את מספר ההיפוכים במערך הקלט עבור כל אחד מהניסויים (מספיק עד ).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| מספר סידורי i | מספר היפוכים במערך ממוין | מספר היפוכים במערך ממוין-הפוך | מספר היפוכים במערך מסודר אקראית | מספר היפוכים במערך עם היפוכים סמוכים אקראיים |
| 1 | 0 | 179700 | 90019 | 300 |
| 2 | 0 | 719400 | 356936 | 601 |
| 3 | 0 | 2878800 | 1433347 | 1195 |
| 4 | 0 | 11517600 | 5736973 | 2397 |
| 5 | 0 | 46075200 | 23101475 | 4824 |

1. יש למלא בטבלה הבאה את סך עלויות החיפוש עבור כל אחד מהניסויים.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| מספר סידורי i | עלות חיפוש במערך ממוין | עלות חיפוש במערך ממוין-הפוך | עלות חיפוש במערך מסודר אקראית | עלות חיפוש במערך עם היפוכים סמוכים אקראיים |
| 1 | 599 | 9012 | 7656 | 789 |
| 2 | 1199 | 20420 | 17808 | 1578 |
| 3 | 2399 | 45636 | 40387 | 3162 |
| 4 | 4799 | 100868 | 89864 | 6311 |
| 5 | 9599 | 220932 | 197761 | 12623 |
| 6 | 19199 | 480260 | 437497 | 25159 |
| 7 | 38399 | 1037316 | 957710 | 50378 |
| 8 | 76799 | 2228228 | 2063760 | 100800 |
| 9 | 153599 | 4763652 | 4419926 | 201581 |
| 10 | 307199 | 10141700 | 9533728 | 403027 |

1. כדי לחסום מלמעלה את סך עלויות החיפוש באופן תאורטי, נסמן ב- את מספר האיברים לפני האיבר באינדקס שגדולים ממנו.
   1. הסבירו מדוע .

יהי מערך A. נסמן .

מהגדרת היפוך נסיק כי ההגדרה נכונה גם בכיוון השני ולכן ניתן להגדיר אותה כך –  
לכל זוג אינדקסים המקיימים A[i] < A[j] נספור היפוך אחד.

מכך נסיק כי במעבר על כל האינדקסים. (מההגדרה נסיק כי

וספירת כל האיברים לפניהם הגדולים מהם נקבל את כמות ההיפוכים הנדרשת.  
**בהתאם על פי הגדרת נקבל כי**  .

* 1. חסמו את עלות החיפוש בעת הכנסת האיבר ה- כפונקציה של , והסיקו כי סך עלויות החיפוש לאורך סדרת ההכנסות הוא .  
     הערה: כאשר הסיבוכיות היא עבור , ניתן להמיר לביטוי כדי לעבור לביטוי אחד פשוט וחוקי שחוסם את שניהם מלמעלה.

יהי מערך A, נסמן .

נעבור על כל האינדקסים . (מההגדרה נסיק כי

מכיוון שאנו משתמשים ב-finger insert, בכל חיפוש נתחיל מהמקסימום, נעלה למעלה עד שנגיע לצומת מתאים לדרישה ואז נתחיל לרדת עד שנגיע לצומת מתאים להכנסה.

נחלק למקרים –

* . מכך נסיק כי הצומת החדש יוכנס כבן למקסימום. בהתאם, סך עלות החיפוש כלפי מעלה תהיה 1.
* . מכך נסיק כי נטפס למעלה לפחות צומת אחד ואז נתחיל לרדת.  
  כפי שנלמד על finger insert, הטיפוס למעלה חסום על ידי לוג 2 של כמות האיברים הגדולים מ-A[i] שכבר נמצאים בעץ. בהתאם, עלות החיפוש במקרה זה תהיה .

על-פי ההערה נסיק כי עלות החיפוש כלפי מעלה היא

בהתאם לאחר מעבר על כל האינדקסים נקבל כי עלות החיפוש כלפי מעלה תהיה . ועל-פי חוקי לוגריתמים נקבל .

ממבנה עץ AVL נסיק כי עלות החיפוש כלפי מטה חסומה על-ידי ביטוי זה +1.

בהתאם, **נקבל כי עלות החיפוש חסומה על-ידי .**

* 1. השתמשו באי-שוויון הממוצעים כדי לחסום מלמעלה את סך עלויות החיפוש כפונקציה של . הסבירו מדוע זוהי העלות הדומיננטית מבחינה אסימפטוטית.

תחילה נשים לב כי על-פי חוקי לוגריתמים מתקיים: (\*)

בנוסף, על-פי אי-שיוויון הממוצעים מתקיים   
מהגדרת I וסעיף א׳ נסיק כי.   
נציב זאת חזרה בביטוי ב-(\*) ונקבל

**סה״כ קיבלנו כי עלות החיפוש חסומה אסימפטוטית על-ידי .**

* 1. השוו בין החסם שהתקבל ותוצאות הניסויים.  
       
     ההשוואה מראה לנו כי אכן זהו חסם לעלות החיפוש.  
     ניתן לשים לב כי בניסוי הממוין, המקרה בעל עלות החיפוש הכי קטנה, עלות החיפוש חסומה ב- . (מכיוון שמתקיים I=0)  
     וכי בניסוי הממוין הפוך, המקרה בעל עלות החיפוש הכי גבוהה, עלות החיפוש חסומה ב-.  
     אלו הם מקרי הקצה, בשאר המקרים עלות החיפוש נמצאת בין שני המקרים ובהתאם גם כן חסומה על-ידי החסם של סעיף ג׳ – .