מסמך תיעוד

מימוש עץ AVL מתבסס על שני מחלקות: AVLNode ו-AVLTree.

מחלקת **AVLNode** מייצגת צומת בעץ AVL. להלן תיאור הפונקציות במחלקה:

Init:

בפונקציה זו אנחנו משתמשים בבנאי ונותנים לו מפתח וערך שייצגו את הצומת, מאתחלים משתנים כמו ילד ימני, ילד שמאלי, הורה וגובה.

get_left/right:

פונקציות אלו מקבלות מצביע לצומת ומחזירות מיהו הבן ימני/שמאלי של אותו צומת, במידה וערך הצומת הוא None מייצר צומת וירטואלי.

get_parent:

פונקציה זו מקבלת מצביע לצומת ומחזירה מצביע אל ההורה במידה וקיים אחרת מחזיר None.

get_key/value:

פונקציות אלו מקבלות מצביע לצומת ומחזירה את המפתח/ערך שלו, במידה והצומת הוא None או צומת וירטואלי מחזיר None.

get_height:

פונקציה זו מקבלת מצביע לצומת ומחזירה מהו הגובה שלו, במידה והצומת הוא וירטואלי מחזיר 1-.

Is_real_node:

פונקציה זו מקבלת מצביע לצומת ומחזירה האם הצומת הוא וירטואלי או לא.

Delete_Node_Only:

פונקציה זו מקבלת מצביע לצומת ועץ AVL. פונקציה זו מוחקת את הצומת עליו היא מצביעה. פונקציה זו קוראת לפונקציית עזר Successor שבמחלקת AVLTree,הפונקציה מקבלת גם מצביע לצומת ועץ AVL ומחזירה את היורש של הצומת עליו מצביעה. פונקציה זו קוראת לפונקציית עזר Min אשר במחלקת AVLTree, הפונקציה מקבלת מצביע לצומת ועץ AVL ומוצאת את האיבר המינימלי של תת העץ אשר מכיל את הצומת. הפונקציה Min יורדת לילד השמאלי עד אשר מגיעה לעלה. הסיבוכיות היא O(logn) שכן המקרה הגרוע הוא להתחיל מהשורש ולרדת עד לעלה. הפונקציה Succeseeor מחזירה את המינימום של בן ימני במקרה ויש לו בן שמאלי, אחרת עולה במעלה העץ עד שמגיע לצומת שיש לו בן שמאלי. פעולה זו לוקחת גם במקרה הגרוע O(logn) כאשר מחפשים מינימום של בן ימני או שיכול להיות שנתחיל מעלה ונעלה עד לשורש על מנת למצוא יורש. לכן הסיבוכיות של מחיקת צומת היא O(logn), ישנם המקרים הפשוטים שבהם משנים את הפוינטרים בהתאם למקרה וזה לוקח O(1) אך במקרים בהם לצומת יש 2 ילדים אנחנו מחפשים את היורש שזה יכול לקחת O(logn) ולאחר מכן משנים את הפוינטרים של היורש, ההורה של הצומת וההורה של היורש, וזה לוקח עוד O(1) ולכן הסיבוכיות היא O(logn).

Balance_Factor:

פונקציה זו מקבלת מצביע אל הצומת ומחזירה את balance factor של הצומת כפי שהגדרנו בכיתה. אנחנו לוקחים את שדה הגובה של הילד השמאלי והימני ומחזירים את ההפרש ביניהם. סיבוכיות של הפונקציה היא O(1) שכן גישה אל השדה של צומת לוקחת זמן קבוע.

update_height:

פונקציה זו מקבלת מצביע אל הצומת, בודקת אם אינו צומת וירטואלי. במידה ולא היא נגשת אל שדה הגובה של הבן הימני והשמאלי ומחשבת את הגובה לפי הגדרה של גובה של צומת בעץ. לאחר מכן ניגשים לשדה הגובה של הצומת ובודקים האם הוא שונה מערך הגובה שחישבנו, במידה וכן מעדכנים את גובה הצומת, פונקציה זו נועדה לעדכן את הגובה של הצומת לאחר שינויים כמו גלגול או מחיקה בעץ. הסיבוכיות היא O(1) שכן לגשת אל שדה של הצומת וילדיו לוקח זמן קבוע.

מחלקת AVLTree:

מחלקה זו מייצגת עץ AVL. להלן הפונקציות במחלקה זו:

Init:

בפונקציה זו אנחנו בונים את העץ, מאתחלים שדה של שורש וגודל. לוקח O(1).

search:

בפונקציה זו אנחנו מקבלים עץ ומפתחת ומבצעים חיפוש האם קיים צומת עם אותו מפתח. החיפוש מתחיל מהשורש ובכל פעם הולכים לבן ימני או שמאלי בהתאם למפתח. הסיבוכיות היא O(logn) שכן במקרה הגרוע הצומת בעל אותו מפתח נמצא בעלה או שלא נמצא כלל בעץ ולכן מבצע פעולות כגובה העץ שהוא logn.

Insert:

בפונקציה זו אנחנו מקבלים עץ, מפתח וערך. מבצעים הכנסה של צומת בעל המפתח והערך לעץ ומחזירים את מספר הפעולות שנדרשו לאזן את העץ לאחר הכנסה . אנחנו נעזרים בפונקציות עזר כמו init של המחלקה AVLNode, Balance_Factor, update_height ושהן כולן בעלות סיבוכיות של O(1).תחילה מבצעים חיפוש לאן להכניס את הצומת, ההכנסה מתבצעת בעלה ולכן לוקחת O(logn). לאחר מכן מתחילים מאותו צומת לעלות כלפי מעלה ולבצע רוטציות ועדכון גבהים על מנת לאזן את העץ. אנחנו קוראים לפונקציות עזר LeftRotation/RightRotation אשר מקבלות עץ ומצביע לצומת ומבצעות רוטציה, פונקציות אלו לוקחות זמן O(1) שכן מבצעים בהן שינוי של פוינטרים בלבד וזה לוקח זמן קבוע. כאשר עולים מעלה אנחנו נדרש לכל היותר גלגול אחד או גלגול כפול אבל כיוון שלא ידוע היכן נדרש ואם בכלל אנחנו עלולים לעלות חזרה אל השורש במקרה הגרוע וזה לוקח O(logn). ולכן בסך הכל הסיבוכיות היא O(logn).

Delete:

בפונקציה זו אנחנו מקבלים עץ ומצביע אל צומת. אנחנו מבצעים מחיקה של אותו הצומת ומחזירים את מספר פעולות האיזון הנדרשות . פונקציה זו קוראת לפונקציית עזר Delete_Node_Only אשר מבצעת מחיקה של הצומת והיא לוקחת במקרה הגרוע O(logn). לאחר מכן מתחילים מההורה של הצומת שנמחק פיזית ועולים למעלה ומבצעים פעולות

איזון. אנחנו קוראים לפונקציות עזר LeftRotation/RightRotation אשר מקבלות עץ ומצביע לצומת ומבצעות רוטציה, פונקציות אלו לוקחות זמן O(1) שכן מבצעים בהן שינוי של פוינטרים בלבד וזה לוקח זמן קבוע. במחיקה אנחנו לא יודעים כמה גלגולים דרושים ולכן במקרה הגרוע נעלה עד לשורש וזה יכול לקחת במקרה הגרוע O(logn).

avl_to_array:

פונקציה זו מקבל עץ AVL ומחזירה רשימה של איברי העץ בצורה ממוינת. פונקציה זו משתמשת ברקורסיה וליתר דיוק באלגוריתם in order אשר עוברת על כל איברי העץ והסיבוכיות היא O(n).

size:

פונקציה זו מקבלת עץ ומחזירה מה גודלו של העץ. פונקציה זו קוראת לפונקציית עזר avl_to_array ולכן הסיבוכיות היא O(n).

split:

הפונקציה מקבלת צומת בעץ שלפיו אנו מפצלים את העץ לשני תתי עצים וללא הצומת הנ"ל. ניצור שני תתי עצים, בתת עץ אחד נכניס כשורש את הבן השמאלי של הצומת והוא יהיה התת עץ של האיברים בעלי המפתחות הקטנים ממפתח הצומת. בתת העץ השני נכניס כשורש את הבן הימני והוא יהיה התת עץ של האיברים בעלי המפתחות הגדולים ממפתח הצומת. נרוץ במעלה המסלול מהצומת אל השורש של העץ המקורי. בכל צומת שעולים נבדוק אם הבן השמאלי שלו הוא הצומת שקיבלנו בקלט. במידה וכן נבנה תת עץ נוסף ונכניס לתוכו את הבן הימני ונעשה JOIN ביחד עם התת עץ של האיברים הגדולים מהצומת שקיבלנו בקלט. בצורה סימטרית נעשה אותם סדר פעולות רק לבן השמאלי של הצומת ונעשה JOIN עם התת עץ של האיברים הקטנים מהצומת בקלט.
הפונקציה מחזירה רשימה שבה התת עץ של האיברים הקטנים והתת עץ של האיברים הגדולים.

סיבוכיות הריצה היא סכום עלויות הJOIN שאנו מפעילים. כמו שראינו בהרצאה הסכום מגיע לסיבוכיות של O(logn)

join:

הפונקציה מקבלת עץ נוסף וערכי KEY,VALUE שהם יהיו הצומת שימזג בין העצים. תחילה נבדוק מקרי קצה בהם אחד או שניים מהעצים ריק או בעל צומת בודד. במידה וכן המיזוג פשוט- אם אחד העצים ריק נכניס את הצומת עם ערכי הקלט לעץ השני ונתאים את השורש בעץ הנוכחי התאם. אם אחד העצים בעל צומת אחד נכניס לעץ השני את הצומת עם ערכי הקלט ואת הצומת של העץ עם הצומת הבודד.

במקרה הכללי נשווה בין הגבהים של שורשי שני העצים, נרוץ על העץ הגבוה יותר עד שנגיע לגובה של השורש של העץ השני. נכניס את הצומת עם ערכי הקלט שקיבלנו וניתן לו כשני ילדים את השורש של העץ הנמוך יותר ואת הצומת שהגענו אליו בעץ הגבוה יותר.

לאחר מכן נבצע את איזון העץ באותה צורה בדיוק כמו שעשינו בINSERT

סיבוכיות הפונקציה היא כמו שלמדנו O(logn)

get_root:

פונקציה זו מקבלת עץ ומחזירה את המצביע אל השורש. הסיבוכיות היא O(1).

LeftRotatoin/RightRotation:

פונקציות אלו מקבלות עץ ומצביע אל צומת. הפונקציות מבצעות גלגול ימינה/שמאלה ואלו עוזרת לאזן את העץ. הפונקציה קוראת לפונקציית עזר update_height על מנת לעדכן את הגבהים של הצמתים שהשתתפו בגלגול. בפונקציה הזו אנחנו רק משנים פוינטרים וגם בעדכון הגבהים קוראים לשדות של הצמתים שהשתתפו בגלגול ולכן הסיבוכיות היא O(1).

Min:

פונקציה זו מקבלת עץ ומצביע אל צומת ומחזירה את האיבר המינימלי של תת העץ במכיל את הצומת. בפונקציה זו אנחנו מתחילים מהצומת ויורדים לבן השמאלי עד שמגיעים לעלה. במקרה הגרוע מתחילים משורש ויורדים לעלה ולכן הסיבוכיות היא O(logn).

Successor:

הפונקציה מקבלת עץ ומצביע לצומת ומחזירה את הצומת שהוא היורש שלו. במידה וילד הימני שלו יש בן שמאלי אנחנו מחפשים את הצומת המינימלי של תת העץ של הילד השמאלי של הילד הימני של הצומת. פונקציה זו קוראת לפונקציית עזר Min.במדיה ואין בן ימני עולים מעלה עד אשר מוצאים את הצומת במסלול למעלה שיש לו בן שמאלי במסלול. הסיבוכיות היא O(logn) , במקרה שבו יש לבן הימני בן שמאלי אז קוראים לפונקציית Min וזה במקרה הגרוע O(logn) אחרת עולים מעלה וזה במקרה הגרוע O(logn).

תוצאות הניסוי

מספר סידורי i	עלות join מקסימלי עבור split אקראי	עלות join מקסימלי עבור split אקראי	עלות join ממוצע עבור split של האיבר מקסימלי בתת עץ שמאלי	עלות join מקסימלי עבור split של איבר מקסימלי בתת עץ שמאלי
1	2.66	5	3.1	12
2	2.3	5	2.72	13
3	2.76	5	2.54	15
4	2.909	5	2.923	15
5	3.230	6	3.066	17
6	3.23	6	3.071	18
7	3.2	7	2.705	19
8	3.133	8	2.578	20
9	3.166	5	3.052	22
10	2.9	7	2.894	23

תשובה לשאלה 2:

את פעולת הsplit מבצעים בעזרת join וכידוע העלות של join זה הפרש הגבהים פלוס 1. מכיוון שאנחנו עושים זאת על עץ AVL שידוע שהפרש הגבהים בין תתי העץ של צומת מסוים הוא לכל היותר 1. לכן בממוצע פעולות הjoin לא יהיו יקרות כל כך ולכן העלות של join אקראי הוא נע בין 2 ל3.

תשובה לשאלה 3:

כאשר מבצעים פעולת split על איבר מקסימלי של תת העץ השמאלי אנחנו מבצעים פעולת join החל מאיבר זה, מכיוון שהוא איבר האמצע של העץ אנחנו בעצם נתחיל מצומת זה ונעלה עד לשורש ובכל עלייה מבצעים join ואז כאשר מגיעים לשורש קיבלנו את הפיצול הנדרש של עץ אחד המכיל מפתחות קטנים מהצומת והעץ השני שהוא תת עץ ימני של העץ מקורי נכיל את כל המפתחות הגדולים מהצומת.

אישור הארכה של שבוע להגשת הפרוייקט

פרויקט מעשי מבני נתונים דואר נכנס

ממני 19 בפבר׳
אל Dani

שלום דני,
שמי יונתן פנוב ת.ז 206580102.
אני מילואימניק שמשוייך לקבוצה 1 ולצערי אני נאלץ לבקש הארכה של כמה ימים בהגשת המטלה המעשית.
נתקלנו בכמה באגים ולצערי הזמינות שלי נמוכה אז מתעכב לנו סיום המטלה.
תודה רבה,
יונתן פנוב

Dani Dorfman 19 בפבר׳
אל אני, Dani

היי יונתן,

מאשר הארכה של שבוע. תצרפו צילום מסך של המייל הזה לpdf של התרגיל כדי שלא יורידו בבדיקה.

תודה על השירות,
דני