מחלקת AVLNode

|  |  |  |
| --- | --- | --- |
| שדה (חבר) | ערך דיפולטי | תפקיד |
| Key | none | מפתח של הצומת |
| Value | none | ערך של הצומת |
| left | none | צומת שמאלי (none אם לא קיים כזה) |
| Right | none | צומת ימני (none אם לא קיים כזה) |
| Parent | none | צומת הורה (none אם שורש ולא קיים הורה) |
| height | -1 | גובה (מספר הקשתות מהעלה הכי קרוב). -1 אם עץ וירטואלי |
| size | 0 | גודל תת העץ של הצומת (0 אם עץ וירטואלי) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| שם הפונקציה | תפקיד | ערך החזרה | תיאור הלוגיקה | סיבוכיות |
| Get\_key(self) | מחזיר את המפתח של הצומת self (none אם וירטואלי) | Self.key |  |  |
| Get\_value(self) | מחזיר את הערך של הצומת (none אם וירטואלי) | Self.value |  |  |
| Get\_left(self) | מחזיר את הצומת השמאלי (none אם לא קיים כזה) | Self.left |  |  |
| Get\_right(self) | מחזיר את הצומת הימני (none אם לא קיים כזה) | Self.right |  |  |
| Get\_parent(self) | מחזיר את ההורה (none אם לא קיים כזה) | Self.parent |  |  |
| Get\_height(self) | מחזיר את הגובה של תת העץ של הצומת (-1 אם וירטואלי) | Self.height |  |  |
| Get\_size(self) | מחזיר את הגובה של תת העץ של הצומת (0 אם וירטואלי) | Self.size |  |  |
| Set\_key(self, key) | מגדיר מפתח של צומת | self |  |  |
| Set\_value(self, value) | מגדיר ערך של צומת | Self |  |  |
| Set\_left(self, node) | מגדיר צומת כבן שמאלי | Self |  |  |
| Set\_right(self, node) | מגדיר צומת כבן ימני | Self |  |  |
| Set\_parent(self, node) | מגדיר צומת כהורה | Self |  |  |
| Set\_height(self, height) | מגדיר גובה לתת העץ של הצומת | Self |  |  |
| Set\_size(self, size) | מגדיר גודל לתת העץ של הצומת | Self |  |  |
| Is\_real\_node(self) | מחזיר האם הצומת אמיתי או וירטואלי | Self.key!=none |  |  |
|  |  |  |  |  |

מחלקת AVLTree

|  |  |  |
| --- | --- | --- |
| שם שדה (חבר) | ערך דיפולטי | תפקיד |
| root | none | שורש העץ |
|  |  |  |
|  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| שם הפונקציה | תפקיד | ערך החזרה | לוגיקה | סיבוכיות |
| Search(self,key) | מחזיר מצביע לצומת המתאים למפתח | Node (none אם לא קיים כזה) | 1. אם העץ ריק נחזיר none 2. נבדוק האם שורש העץ הוא הערך הנדרש    1. אם כן נחזיר אותו    2. אם הערך הנדרש קטן יותר מהשורש, נחזור לסעיף 1 עם הבן השמאלי    3. אם הערך הנדרש גדול יותר נחזור לסעיף 1 עם הבן הימני | כאשר n גודל העץ. זו הסיבוכיות במקרה הגרוע בגלל שגובה העץ חסום על ידי  *כפי שראינו בהרצאה* |
| Insert\_node\_bst(self, key, val) | הכנסה לעץ חיפוש בינארי רגיל  כולל שינוי הsize, height של הצמתים המושפעים מהתהליך | מצביע לצומת החדש שנוצר | 1. אם העץ ריק – מכניס את הצומת כשורש והגדל את גודל הצומת הנוכחי ב1 2. אם העץ צומת יחיד- הכנס שמאלה אם קטן יותר או ימינה אם גדול יותר והגדל את גודל הצומת הנוכחי ב1 3. אחרת    1. אם הצומת בעל מפתח קטן יותר, חזור לסעיף 2 עם תת העץ השמאלי    2. אם הצומת בעל מפתח גדול יותר, חזור לסעיף 2 עם תת העץ הימני 4. אם הצומת נכנס כבן יחיד להורה שלו, הגדל ב1 את גובה הצמתים במסלול שלו לשורש שגובהם השתנה | כאשר n גודל העץ. זו הסיבוכיות במקרה הגרוע בגלל שגובה העץ חסום על ידי  *כפי שראינו בהרצאה* |
| Calculate\_BF(self, node | מחשב את הBF של הצומת | BF | גובה תת העץ השמאלי – גובה תת העץ הימני |  |
| Insert(self, key, val) | הכנסה לעץ AVL (כולל הכנסה תוך שמירה על דרישות העץ בעזרת גלגולים) |  |  |  |