פרויקט מעשי AVLTREE:

מחלקת העל של AVLTREE	
בנאי ראשון:	קלט: אין פלט: יוצר עץ ריק חדש, מאתחל את המינימום ומקסימום להיות null. סיבוכיות: O(1).
בנאי שני:	קלט: מפתח מטיפוס int וערך מטיפוס String פלט: יוצר עץ חדש עם שורש בעל המפתח שקיבלנו והערך, מעדכן את המינימום ומקסימום להיות השורש. סיבוכיות: O(1).
empty	קלט: אין פלט: מחזיר true אם העץ ריק וfalse אחרת. סיבוכיות: O(1).
Search	קלט: מפתח מטיפוס int פלט: הערך(מטיפוס String) השמור בעץ תחת אותו המפתח שקלטנו. הפונקציה משתמשת בפונקציית עזר פנימית SearchRec סיבוכיות: O(logn) **SearchRec** קלט: מפתח מטיפוס int וצומת מטיפוס IAVLNode פלט: חיפוש אחר הצומת עם המפתח הנתון באמצעות חיפוש בינארי רקורסיבי, אם המפתח שקיבלנו גדול משורש העץ תישלח אל הפונקציה תת העץ הימני, ואם קטן יישלח אל הפונקציה תת העץ השמאלי וכך הלאה בצורה רקורסיבית עד למציאת המפתח המתאים. סיבוכיות:O(logn)
insert	קלט: מפתח מטיפוס int וערך מטיפוס String פלט: ערך מספרי מטיפוס int. הפונקציה מבצעת הכנסה לעץ הקיים ופעולות גלגול על מנת לדאוג שהעץ יהיה עץ AVL תקין בסיום ההכנסה. הפונקציה מחזירה את כמות הגלגולים שנדרשו ו (1-) אם הצומת כבר היה קיים קודם לכן (שימוש בפונקציה search שמופיעה שורה אחת למעלה בסיבוכיות O(logn)). הפונקציה משתמשת במספר פעולות עזר- insertPlace, updateHeight, updateSize, promote, demote, checkBalance, LLrotation, RRrotation, LRrotation, RLrotation. סיבוכיות: O(logn)- הפונקציה מתבצעת כפי שנלמד בכיתה, ונראה כי גם פעולות העזר שבהן השתמשנו סיבוכיותן לכל היותר O(logn).

insertPlace:
קלט: צומת מטיפוס IAVLNode ומפתח מטיפוס int
פלט: צומת מטיפוס IAVLNode .
נקרא לפונקציה על מנת למצוא את נקודת ההכנסה המיועדת עבור הצומת אותו נרצה להכניס לעץ. הפונקציה מחזירה את הצומת אשר עתידה להיות האבא של הצומת החדשה אותה אנחנו רוצים להכניס לעץ. הפונקציה מחזיקה תחילה מצביע אל השורש ורצה לאורך העץ מטה, בכל איטרציה בודקת האם המפתח קטן ממנו ממפתח עליו שמור המצביע, במידה וכן תשנה את המצביע להיות הבן השמאלי שלו, במידה וגדול ממנו, תשנה את המצביע להיות הבן הימני שלו. באופן זה נמשיך עד למציאת המקום הראשון הפנוי המתאים.
סיבוכיות: O(logn)

updateHeight

קלט: צומת מטיפוס IAVLNode
פלט: אין. הפונקציה מעדכנת את גובה העץ לאחר ההכנסה החל מהצומת שהתקבל עד למעלה השורש – לאורך המסלול שעברנו בעת ההכנסה.
סיבוכיות: O(logn) – לכל היותר נעבור על אורך גובה העץ.

updateSize

קלט: צומת מטיפוס IAVLNode
פלט: אין. הפונקציה מעדכנת את מספר האיברים בעץ החל מהצומת שהתקבל עד לשורש – לאורך המסלול שעברנו בעת ההכנסה.
סיבוכיות: O(logn)- לכל היותר נעבור על אורך גובה העץ.

Promote

קלט: צומת מטיפוס IAVLNode
פלט: אין. הפונקציה מעדכנת את דרגת הצומת הנקלט להיות 1=+.
סיבוכיות: O(1)

Demote

קלט: צומת מטיפוס IAVLNode
פלט: אין. הפונקציה מעדכנת את דרגת הצומת הנקלט להיות 1=-.
סיבוכיות: O(1)

checkBalance

קלט: צומת מטיפוס IAVLNode
פלט: הפונקציה בודקת האם העץ מאוזן לאחר ההכנסה, במידה ולא מאזנת אותו ומחזירה את מספר פעולות הגלגול שנדרשו לשם כך.
סיבוכיות: O(logn)

BalancrAfterInsert

קלט: צומת מטיפוס IAVLNode את הדרגה של הבן השמאלי מטיפוס int ואת הדרגה של הבן הימני מטיפוס int.
פלט: נקרא לפעולה זו כאשר העץ אינו מאוזן לאחר ההכנסה, נבדוק אילו גלגולים עלינו לעשות כדי לאזן אותו, נבצע אותם ונחזיר את מספר פעולות אלו.
סיבוכיות: O(1)

LLrotation

קלט: צומת מסוג IAVLNode
פלט: אין. הפונקציה מבצעת פעולת גלגול לעץ לכיוון ימין ומעדכנת את הגובה(דרגה) ואת הsize בהתאם לאחר הפעולה.
סיבוכיות: O(logn) – מאחר שמבצעים פעולת עדכון לגובה וגודל העץ פעולות הגלגול עצמן עולות 0(1).

RRrotation

קלט: צומת מסוג IAVLNode
פלט: אין. הפונקציה מבצעת פעולת גלגול לעץ לכיוון שמאל ומעדכנת את הגובה(דרגה) ואת הsize בהתאם לאחר הפעולה.
סיבוכיות: O(logn) - מאחר שמבצעים פעולת עדכון לגובה וגודל העץ פעולות הגלגול עצמן עולות 0(1).

LRrotation

קלט: צומת מסוג IAVLNode
פלט: אין. הפונקציה מבצעת פעולת גלגול לעץ לכיוון שמאל (קוראת לפונקציה RRrotation- שולחת אליה את הבן השמאלי של הצומת הנקלט) ואז פעולת גלגול נוספת לכיוון ימין (קוראת לפונקציה LLrotation שולחת אליה את הצומת הנקלט) ומעדכנת את הגובה(דרגה) ואת הsize בהתאם לאחר הפעולה.
סיבוכיות: O(logn) - מאחר שמבצעים פעולת עדכון לגובה וגודל העץ פעולות הגלגול עצמן עולות 0(1).

RLrotation

קלט: צומת מסוג IAVLNode
פלט: אין. הפונקציה מבצעת פעולת גלגול לעץ לכיוון ימין (קוראת לפונקציה LLrotation שולחת אליה את הבן הימני של הצומת הנקלט) ואז פעולת גלגול נוספת לכיוון שמאל (קוראת לפונקציה RRrotation- שולחת אליה את הצומת הנקלט) ומעדכנת את הגובה(דרגה) ואת הsize בהתאם לאחר הפעולה.
סיבוכיות: O(logn) - מאחר שמבצעים פעולת עדכון לגובה וגודל העץ פעולות הגלגול עצמן עולות 0(1).

Delete	קלט: מפתח מטיפוס int פלט: ערך מספרי מטיפוס int. הפונקציה מוחקת מהעץ את האיבר ששמור עליו המפתח הנקלט ומוודאת כי לאחר המחיקה העץ עדיין AVL תקין. במידה ולא, תבצע פעולות איזון על מנת שהעץ יהיה תקין. ובסופן תחזיר את מספר פעולות האיזון שנאצלה לעשות על מנת לדאוג שהעץ יישאר תקין לאחר המחיקה. הפונקציה משתמשת במספר פונקציות עזר: BSTdelete, deleteLeaf, replace, passBy, successor, predecessor. סיבוכיות: O(logn)- כפי שנלמד בכיתה, ובנוסף נראה כי גם פונקציות העזר סיבוכיותן לכל היותר O(logn). **BSTdelete** קלט: מפתח מטיפוס int פלט: צומת מטיפוס IAVLNode הפונקציה מבצעת מחיקה רגילה מעץ חיפוש בינארי, כפי שנלמד בכיתה ומחזירה את האבא של הצומת שנמחק בפועל. קודם כל בודקת את מקרי הקצה- במידה והמפתח אותו נרצה למחוק שווה למינימום או מקסימום אז מעדכנים אותם להיות ה-successor או הpredecessor בהתאמה. לאחר מכן הפונקציה מאתרת את הצומת של המפתח הנקלט אותו נרצה למחוק ומפרידה למקרים שונים: כאשר הצומת הנמחק הוא עלה, צומת אונרי או צומת בעל שני ילדים. סיבוכיות: O(logn) – הפונקציה פועלת לפי האלגוריתם שהוצג בכיתה למחיקת צומת. **deleteLeaf** קלט: צומת מטיפוס IAVLNode פלט: אין. הפונקציה מטפלת במקרה בו הצומת אותו נרצה למחוק הוא עלה. הפונקציה מחלקת למקרים שונים- כאשר העלה הוא הבן הימני וכאשר העלה הוא הבן השמאלי, בשני המקרים הפונקציה מוחקת את העלה ומעדכנת את הבן המתאים של ההורה שלו להיות null. סיבוכיות: O(1) **replace** קלט: שני צמתים מטיפוס IAVLNode פלט: אין. הפונקציה מקבלת את הצומת אותה נרצה למחוק ואת האיבר העוקב שלה ה- successor במקרה בו יש לצומת אותו נרצה למחוק שני בנים. במקרה זה נשלח את המשתנים הללו אל הפונקציה ונרצה להחליף ביניהם ולכן הפונקציה מחליפה את המפתח והערך Value בין השניים. סיבוכיות: O(1)

	passBy קלט: צומת מטיפוס IAVLNode פלט: אין. הפונקציה מקבלת צומת עליה נרצה "לדלג" כפי שלמדנו בשיעור, נחבר בין הבן של הצומת (שידוע שקיים- תנאי לקריאה לפונקציה) לבין האבא שלו. סיבוכיות: O(1) **Successor** קלט: צומת מטיפוס IAVLNode פלט: מחזירה צומת מטיפוס IAVLNode שהוא האיבר העוקב של הצומת הנקלט (לפי מפתח). סיבוכיות: O(logn) **Predecessor** קלט: צומת מטיפוס IAVLNode פלט: מחזיר צומת מטיפוס IAVLNode שהוא האיבר הקודם של הצומת הנקלט (לפי מפתח). סיבוכיות: O(logn)
max	קלט: אין פלט: מחזיר את הערך String של הצומת השמור בתור max. סיבוכיות: O(1).
min	קלט: אין פלט: מחזיר את הערך String של הצומת השמור בתור min. סיבוכיות: O(1).
keysToArray	קלט: אין פלט: מערך מספרים מכיל את כל המפתחות הקיימים בעץ. הפונקציה מעדכנת מערך keys_arr שאותחל מראש להיות מערך חדש בגודל העץ. לאחר מכן מתבצעת קריאה לפונקציה רקורסיבית orderKeysToArray עם השורש. במידה והעץ ריק הפונקציה תחזיר מערך ריק. סיבוכיות:O(n) **orderKeysToArray** קלט: צומת מטיפוס IAVLNode פלט: אין. הפונקציה מבצעת סיור in-order על העץ ומעדכנת את המערך שאתחלנו מראש עם המפתחות המתאימים. משתמשת באינדקס סטטי index_of_keys_arr ומערך סטטי. סיבוכיות: O(n)

infoToArray	קלט: אין. פלט: מערך מחרוזות המכיל את כל המידע של המפתחות הקיימים בעץ. הפונקציה מעדכנת את המערך info_arr שאותחל מראש להיות מערך חדש בגודל העץ. לאחר מכן מתבצעת קריאה לפונקציה רקורסיבית orderInfoToArray עם השורש. במידה והעץ ריק הפונקציה תחזיר מערך ריק. סיבוכיות: O(n). **orderInfoToArray** קלט: צומת מטיפוס IAVLNode פלט: אין. הפונקציה מבצעת סיור in-order של העץ ומעדכנת את המערך שאתחלנו מראש עם הערכים של המפתחות. משתמשת באינדקס סטטי index_of_info_arr ומערך סטטי. סיבוכיות: O(n)
size	קלט: אין פלט: מחזיר את כמות הצמתים בעץ – טיפוס מסוג Int. סיבוכיות: O(1)
getRoot	קלט: אין פלט: מחזיר את השורש של העץ מטיפוס IAVLNode או null אם העץ ריק. סיבוכיות: O(1)
Split	קלט: מפתח מטיפוס int פלט: מערך מטיפוס IAVLNode שמכיל את שני עצי הAVL. הפונקציה מפרידה את העץ לשני עצי AVL כאשר המפתחות של האחד גדולים מx ושל השני קטנים מx. תחילה הפונקציה בודקת האם המפתח שהתקבל הוא המינימום או המקסימום במידה וכן, מוחקת אותו ומחזירה את המערך המתאים. בשלב הבא, מחפשת את המפתח בעץ, כלומר את המקום ממנו נרצה לבצע את ה-split. הפונקציה נכתבה לפי האלגוריתם שהוצג בכיתה- עולים במעלה העץ ובכל איטרציה מעדכנים את העץ שמכיל את הערכים שגדולים מהמפתח הנקלט ואת העץ של הערכים שקטנים ממנו. משתמשת בפונקציות findMin,findMax,find סיבוכיות:O(logn) **findMax** קלט: צומת מטיפוס IAVLNode- שורש העץ פלט: צומת מטיפוס IAVLNode - מחזירה את הצומת המקסימלית בעץ. סיבוכיות: O(logn)

	findMin קלט: צומת מטיפוס IAVLNode -שורש העץ פלט: צומת מטיפוס IAVLNode – מחזירה את הצומת המינימלית בעץ. סיבוכיות: O(logn). **find** קלט: מפתח מטיפוס int פלט: צומת מטיפוס IAVLNode – מחזירה את הצומת עם המפתח השמור אותו קלטנו וnull במידה ולא קיים צומת עם מפתח זה. משתמשת בשיטת החיפוש- חיפוש בינארי בצורה רקורסיבית.- קוראת לפונקציית עזר findRec סיבוכיות: O(logn).
join	קלט: צומת מטיפוס IAVLNode ועץ מטיפוס AVLTree. פלט: מספר שלם מטיפוס int – הפונקציה מחזירה את העלות פעולת ה-join- הפרשי גבהי העצים 1+. מטרת הפונקציה היא לאחד את הצומת והעץ הנקלטים יחד עם העץ הנתון לעץ אחד. בתחילתה בודקת מקרי קצה- אם אחד מבין הנקלטים או העץ הנתון ריקים, לאחר מכן בודקת מי מהעצים גבוה יותר (מבחינת ערכי המפתחות). בהתאם מחליטה הפונקציה האם להוסיף מצד ימין או שמאל לעץ הנתון. משתמשת בפונקציות עזר addOnLeft, addOnRight. סיבוכיות:O(logn) – הפונקציה מבוצעת כפי שנלמד בכיתה ונראה גם כי פונקציות העזר סיבוכיותן לכל היותר O(logn). **addOnLeft** קלט: שני עצים מטיפוס AVLTree וצומת מטיפוס IAVLNode פלט: צומת מטיפוס IAVLNode– מחזירה את השורש של העץ המאוחד. הפונקציה מוסיפה את העץ הנמוך יותר (בעל הערכים הקטנים יותר)מצד שמאל של העץ הגבוה יותר(בעל הערכים הגדולים יותר). סיבוכיות: O(logn)- לאורך הפונקציה מחפשים את הנקודה בה יש לבצע את הjoin. בנוסף קוראת לפונקציה checkBalance על מנת לבדוק את האיזון לאחר ה- Join. **addOnRight** קלט: שני עצים מטיפוס AVLTree וצומת מטיפוס IAVLNode פלט: צומת מטיפוס IAVLNode– מחזירה את השורש של העץ המאוחד. הפונקציה מוסיפה את העץ הנמוך יותר(בעל הערכים הגדולים יותר) אל העץ הגדול יותר מצד ימין (בעל הערכים הקטנים יותר). סיבוכיות: O(logn)- לאורך הפונקציה מחפשים את הנקודה בה יש לבצע את הjoin. בנוסף קוראת לפונקציה checkBalance על מנת לבדוק את האיזון לאחר ה- Join.

מחלקה מקוננת של AVLTREE מממשת את המנשק IAVLTree	
בנאים	בפונקציה שני בנאים כאשר ההבדלים ביניהם הם רק בקלטים המועברים אליהם. בנאי ראשון מפתח מטיפוס int , ערך מטיפוס String. ואילו הבנאי השני מקבל מפתח מטיפוס int , ערך מטיפוס String, וצומת מטיפוס IAVLNode שהוא האבא הייעודי של הצומת החדשה אותה ניצור. סיבוכיות O(1).
get and set	לכל שדה ישנה מתודת get אשר מחזירה את ערך השדה בסיבוכיות O(1). לשדות size, height, left, right, parent, key, value יש גם set שמעדכן את הערך של השדות לפי קלט נתון בסיבוכיות O(1).
isRealNode	<u>קלט:</u> אין <u>פלט:</u> הפונקציה מחזירה ערך true אם הצומת אינו וירטואלי כלומר, אם המפתח שונה מ(-1) אחרת מחזירה false. <u>סיבוכיות:</u>O(1)