AVLTree

מייצג את העץ עצמו. מכיל משתנה מסוג IAVLNode הנקרא 'root' ומייצג את שורש העץ.

```
empty()
```

return type – boolean

precondition – none

postcondition – none

סיבוכיות – O(1)

תיאור קצר – מחזיר true אם העץ ריק ('root' הוא null), אחרת false.

```
search(int k)
```

return type – String

precondition – none

postcondition – none

סיבוכיות – O(log n)

תיאור קצר – מקבלת מפתח כמספר טבעי k ומחזירה את הצומת עם המפתח k אם קיים, אחרת מחזירה null.

```
insert(int k, String i)
```

return type – int

precondition – none

postcondition – none

סיבוכיות – O(log n)

תיאור קצר – מקבלת מפתח k וערך i תוך כדי שמירה על עץ AVL תקין. הפונ' מחזירה את מספר פעולות האיזון שבוצעו כדי לשמור על תקינות העץ. אם לא התבצעו שום פעולות הפונ' תחזיר 0, אם איבר עם מפתח k כבר קיים בעץ אז תחזיר 1- ולא תכניס את איבר חדש לעץ.

הפונ' בודקת אם העץ ריק, אם כן מכניסה את האיבר החדש כשורש ומחזירה 0, בודקת אם איבר עם מפתח k כבר קיים בעץ (באמצעות search), אם כן מחזירה 1-.

לאחר מכן מתבצעת "הליכה" העץ כדי למצוא את המקום בו נכניס את האיבר החדש ומתבצעת בדיקה אם אבא של האיבר החדש היה צומת אונארי (באמצעות wasUnaryNode), ואם כן מחזירה 0. אם הצומת אינו אונארי עושים promote לאבא, ואז מפעילים את פונ' rebalance על אבא שלו.

```
rebalance(IAVLNode y)
```

return type – none

precondition – none

postcondition – none

סיבוכיות – O(log n)

תיאור קצר – מקבלת כקלט צומת מטיפוס IAVLNode, בודקת איזה תיקון נדרש לבצע בעץ ומפנה לתיקון הרלוונטי

```
balance01(IAVLNode y)
```

return type – none

precondition – none

postcondition – none

סיבוכיות - O(log n)

תיאור קצר – מקבלת כקלט צומת מטיפוס IAVLNode, מבצעת promote על הצומת ומקדמת את מונה האיזונים ב-1, אם אבא של הצומת אינו null קוראת לפונ' rebalance עם האבא כקלט.

```
balance02_12(IAVLNode y)
```

return type – none

precondition – none

postcondition – none

סיבוכיות – O(1)

תיאור קצר – מקבלת כקלט צומת מטיפוס IAVLNode, מבצעת עליו סיבוב ימינה עם בנו השמאלי בעזרת rotateR, מבצעת demote על הקלט ומגדילה ב-2 את מונה האיזונים.

```
balance02_21(IAVLNode y)
```

return type – none

precondition – none

postcondition – none

סיבוכיות – O(1)

תיאור קצר – מקבלת כקלט צומת מטיפוס IAVLNode ומבצעת עליו ועל בנו השמאלי סיבוב שמאלה בעזרת rotateL, לאחר מכן סיבוב ימינה עליו ועל בנו הימני בעזרת rotateR. מבצעת demote על צומת הקלט ואח שלו, promote על אבא שלו ומגדילה את מונה האיזונים ב-5.

```
balance20_12(IAVLNode y)
```

return type – none

precondition – none

postcondition – none

סיבוכיות – O(1)

תיאור קצר – מקבלת כקלט צומת מטיפוס IAVLNode ומבצעת עליו ועל בנו הימני סיבוב ימינה בעזרת rotateR, לאחר מכן סיבוב שמאלה עליו ועל בנו השמאלי בעזרת rotateL. מבצעת demote על צומת הקלט ואח שלו, promote על אבא שלו ומגדילה את מונה האיזונים ב-5.

```
balance20_21(IAVLNode y)
```

return type – none

precondition – none

postcondition – none

תיאור קצר – מקבלת כקלט צומת מטיפוס IAVLNode, מבצעת עליו סיבוב שמאלה עם בנו הימני בעזרת rotateL, מבצעת demote על הקלט ומגדילה ב-2 את מונה האיזונים.

```
rotateR(IAVLNode x)
```

return type – none

precondition – none

postcondition – none

סיבוכיות – O(1)

תיאור קצר – מקבלת כקלט צומת מטיפוס IAVLNode ומבצעת סיבוב ימינה עליו ועל בנו השמאלי בעזרת שינוי מצביעים, וכן בודקת מקרי קצה כגון ביצוע סיבוב על שורש. לבסוף מעדכנת את הגודל והגובה של צומת הקלט ובנו הימני.

```
rotateL(IAVLNode x)
```

return type – none

precondition – none

postcondition – none

סיבוכיות – O(1)

תיאור קצר – מקבלת כקלט צומת מטיפוס IAVLNode ומבצעת סיבוב שמאלי עליו ועל בנו הימני בעזרת שינוי מצביעים, וכן בודקת מקרי קצה כגון ביצוע סיבוב על שורש. לבסוף מעדכנת את הגודל והגובה של צומת הקלט ובנו השמאלי

```
wasUnaryNode(IAVLNode y)
```

return type – boolean

precondition – none

postcondition – none

סיבוכיות – O(1)

תיאור קצר – מחזירה true אם צומת הוא אונארי (יש לו רק בן אחד) ו-false אחרת.

```
delete(int k)
```

return type – int

precondition – none

postcondition – the tree must remain valid (keep all its invariants)

סיבוכיות – O(log n)

תיאור קצר – מחיקה של צומת בעלת מפתח k במידה וקיים בעץ. אם אכן מתקיים, נמחק את הצומת ונאזן מחדש את העץ באמצעות rebalanceDel. נמחק בהתאם למקרים השונים: האם הצומת שנמצאה היא אונארית, בינארית, או עלה. נחזיר את מספר הפעולות שבוצעו לאיזון העץ. במידה והצומת לא נמצאה, נחזיר 1-.

`rebalanceDel(IAVLNode node)`

return type – int

precondition – the tree is balanced 'below' node.

postcondition – the tree must be a valid tree

סיבוכיות - O(log n)

תיאור קצר – הפונקציה מופעלת על עץ ומקבלת את הצומת ממנה יש לאזן את העץ. הפונקציה מאזנת את העץ באמצעות חלוקה למקרים בדיוק כפי שמתואר במצגת, ולפי המקרה המתאים, תבצע promote, demote או סיבובים לימין או שמאל. לאחר מכן, לפי המקרה המתאים, נבצע איזון על האב של node או שנסיים את הפעולה. בסופה נחזיר את מספר הפעולות שנדרשו לאיזון העץ כאשר demote, promote או סיבוב בודד נחשבים כל אחד לפעולה אחת. במקרה הגרוע ביותר נתחיל בעלה העמוק ביותר, ונעלה באיזונים עד השורש. לכן הסיבוכיות כגובה העץ, log n.

`Successor (IAVLNode x)`

return type – IAVLNode

precondition – none

postcondition – the tree remains unchanged

סיבוכיות - O(log n)

תיאור קצר – נחפש את הצומת בעלת המפתח הקטן ביותר, שגדול יותר מהמפתח של x. במידה ול-x יש בן ימני, נרד אליו ונמשיך עד הסוף שמאלה (חיפוש של מינימום בתת העץ של המפתחות הגדולים מ-x). במידה ואין לו בן ימני, נעלה במעלה העץ עד שנגיע לצומת הראשונה שמהווה בן שמאלי. אם x הוא המפתח הגדול ביותר, נחזיר את x. במקרה הגרוע ביותר, נעלה את כל העץ, ונרד אותו בחזרה מצידו השני. לכן, הסיבוכיות הינה O(logn).

`min()`

return type – String

precondition – none

postcondition – the tree remains unchanged

סיבוכיות - O(log n)

תיאור קצר – נחפש את הערך של המפתח הכי קטן בעץ. לכן, נרד לצומת הכי שמאלית בעץ, ונחזיר את ערכה. נחזיר null אם העץ ריק. לכל היותר נרד כגובה העץ, ולכן הסיבוכיות הינה O(log n) .

`max()`

return type – String

precondition – none

postcondition – the tree remains unchanged

סיבוכיות - O(log n)

תיאור קצר – נחפש את הערך של המפתח הכי גדול בעץ. לכן, נרד לצומת הכי ימנית בעץ, ונחזיר את ערכה. נחזיר null אם העץ ריק. לכל היותר נרד כגובה העץ, ולכן הסיבוכיות הינה O(log n) .

```
keysToArray()
```

return type – int[]

precondition – none

postcondition – none

סיבוכיות – O(n)

תיאור קצר – אם העץ ריק מחזירה מערך ריק, אחרת מחזירה מערך ממוין של כל המפתחות בעץ באמצעות המתודה inOrder.

```
inOrder(IAVLNode node,int[] st)
```

return type – none

precondition – none

postcondition – none

סיבוכיות – O(n)

תיאור קצר – מקבלת כקלט מערך של int וצומת מטיפוס IAVLNode, אם הצומת הוא אמיתי, באמצעות רקורסיה נעבור על כל הצמתים בעץ מהקטן לגדול ונכניס אותם לפי הסדר באמצעות משתנה int סטטי.

```
infoToArray()
```

return type – String[]

precondition – none

postcondition – none

סיבוכיות – O(n)

תיאור קצר – אם העץ ריק מחזירה מערך ריק, אחרת מחזירה מערך ממוין של כל הערכים בעץ באמצעות המתודה inOrderString.

```
inOrderString(IAVLNode node,String[] st)
```

return type – none

precondition – none

postcondition – none

סיבוכיות – O(n)

תיאור קצר – מקבלת כקלט מערך של string וצומת מטיפוס IAVLNode, אם הצומת הוא אמיתי, באמצעות רקורסיה נעבור על כל הצמתים בעץ מהקטן לגדול ונכניס אותם לפי הסדר באמצעות משתנה int סטטי.

```
size()
```

return type – int

precondition – none

postcondition – none

סיבוכיות – O(1)

תיאור קצר – מחזיר את גודל העץ אשר מוגדר להיות גודל השורש. הגודל מוגדר באופן רקרורסיבי כך שגודל צומת הוא גודל תת העץ הימני שלו + גודל תת העץ השמאלי שלו + 1. גודל עלה מוגדר להיות 1. מוחזר 0 אם העץ ריק (לא קיים שורש).

```
getRoot()
```

return type – IAVLNode

precondition – none

postcondition – none

סיבוכיות – O(1)

תיאור קצר – מחזיר את השורש של העץ, null אם לא קיים.

```
split(int x)
```

return type – AVLTree []

precondition – search(x)!=null

postcondition – none

תיאור קצר – מקבלת כקלט מספר טבעי x, בודקת את מיקומו בעץ, לאחר מכן שומרת את בניו ואם אינם וירטואליים שומרת אותם כשורשים של עצים (אחרת עצים ריקים). הפונ' עולה במסלול הצומת עד השורש וכל פעם בודקת אם הצומת עם המפתח x היה תת עץ ימני או שמאלי של צומת זה ועושה עליו join עם העץ הרלוונטי, לבסוף מחזירה מערך של AVLTree מגודל 2.

```
join(IAVLNode x, AVLTree t)
```

return type – int

precondition – all the keys in t are bigger than the key of x, who is bigger than all the keys in this(), or, all the keys in t are smaller than the key of x, who is smaller than all the keys in this()

postcondition – this is now a valid AVL tree containing previous this, t, and x.

סיבוכיות - O(log n)

תיאור קצר – נחפש את העץ בעל הערכים הגדולים (לשם בהירות נסמנו ב- a). אם הrank שלו גדול משל הrank של העץ בעל הערכים הקטנים (נסמנו ב – b), נרד שמאלה, עד שנגיע לצומת עם rank קטן או שווה לזה של הצומת של עץ b. נחבר את x כאב של שניהם ואת האב של הצומת אליה הגענו כאב של x. כעת נאזן את העץ באמצעות rebalanceDel. כמובן שנדאג לשמר את כל האינווריאנטות כגון height, size, rank וכו. במקרה וה-rank של a קטן משל b, נחבר את a ו – b כבנים של x ואת x נגדיר כשורש של העץ. המקרה בו a בעל הערכים הקטנים ו – b בעל הערכים הגדולים הינו סימטרי.

נשים לב שלכל היותר ירדנו מהשורש עד העלה, ולאחר מכן עשינו rebalanceDel שגם הוא לכל היותר O(log n). לכן הסיבוכיות לכל היותר O(log n).

AVLNode

מחלקה המייצגת צומת. לכל צומת נגדיר מספר משתנים. מסוג AVLNode נגדיר left, right, parent אשר ייצגו בן שמאלי, ימני ואב, בהתאמה. מסוג int נגדיר rank, key, size, height שייצגו דרגה, מפתח, גודל וגובה בהתאמה. מסוג String נגדיר value שייצג את הערך. כל אחד מהמשתנים האלו ייוצג ויישום באופן זהה לנלמד בכיתה.

```
AVLNode(int key,String value)
```

return type – none

precondition – none

postcondition – none

סיבוכיות – O(1)

תיאור קצר – בנאי של המחלקה. מגדיר את key ואת value כפי שהתקבלו. קורא לבנאי AVLNode ומגדיר את הבנים הימניים והשמאליים כצומת וירטואלית.

```
AVLNode()
```

return type – none

precondition – none

postcondition – none

סיבוכיות – O(1)

תיאור קצר – בנאי של המחלקה. מייצר עלה וירטואלי אשר מוגדר להיות בעל ערך ודרגה 1-.

```
getKey()
```

return type – int

precondition – none

postcondition – none

סיבוכיות – O(1)

תיאור קצר – מחזיר את המפתח של הצומת עליה מופעלת הפונקציה. אם הצומת היא וירטואלית מוחזר 1-.

```
getValue()
```

return type – String

precondition – none

postcondition – none

סיבוכיות – O(1)

תיאור קצר – מחזיר את הערך של הצומת עליה מופעלת הפונקציה. אם הצומת היא וירטואלית, מוחזר null.

`setLeft(IAVLNode node)`

return type – none

precondition – none

postcondition – none

סיבוכיות – O(1)

תיאור קצר – מגדיר את הבן השמאלי של הצומת עליה מופעלת הפונקציה להיות node.

`getLeft()`

return type – AVLnode

precondition – none

postcondition – none

סיבוכיות – O(1)

תיאור קצר – מחזיר את הבן השמאלי של צומת עליה מופעלת הפונקציה. Null אם לא קיים.

`setRight(IAVLNode node)`

return type – none

precondition – none

postcondition – none

סיבוכיות – O(1)

תיאור קצר – מגדיר את הבן הימני של הצומת עליה מופעלת הפונקציה להיות node.

`getRight()`

return type – AVLnode

precondition – none

postcondition – none

סיבוכיות – O(1)

תיאור קצר – מחזיר את הבן השמאלי של צומת עליה מופעלת הפונקציה. Null אם לא קיים.

```
setParent(IAVLNode node)
```

return type – none

precondition – none

postcondition – none

סיבוכיות – O(1)

תיאור קצר – נגדיר את האב של הצומת עליה מופעלת הפונקציה להיות node.

```
getParent()
```

return type – AVLNode

precondition – none

postcondition – none

סיבוכיות – O(1)

תיאור קצר – מחזיר את האב של הצומת עליה מופעלת הפונקציה. Null אם לא קיים.

```
isRealNode()
```

return type – boolean

precondition – none

postcondition – none

סיבוכיות – O(1)

תיאור קצר – מחזיר false אם הצומת הינה צומת וירטואלית (המפתח והדרגה שלה 1-). אחרת יוחזר true.

```
setHeight(int height)
```

return type – none

precondition – none

postcondition – none

סיבוכיות – O(1)

תיאור קצר – נגדיר את הגובה של הצומת עליה מופעלת הפונקציה להיות height.

```
getHeight()
```

return type – int

precondition – none

postcondition – none

סיבוכיות – O(1)

תיאור קצר – נחזיר את הגובה של הצומת עליה מופעלת הפונקציה.

`promote()`

return type – none

precondition – none

postcondition – none

סיבוכיות – O(1)

תיאור קצר – נגדיל את הדרגה של הצומת עליה מופעלת הפונקציה ב - 1.

`demote()`

return type – none

precondition – none

postcondition – none

סיבוכיות – O(1)

תיאור קצר – נקטין את הדרגה של הצומת עליה מופעלת הפונקציה ב - 1.

`getRank()`

return type – int

precondition – none

postcondition – none

סיבוכיות – O(1)

תיאור קצר – נחזיר את הדרגה של הצומת עליה מופעלת הפונקציה.

`getSize()`

return type – int

precondition – none

postcondition – none

סיבוכיות – O(1)

תיאור קצר – נחזיר את הגודל של הצומת עליה מופעלת הפונקציה.

```
setSize(int size)
```

return type – none

precondition – size ≥ 0

postcondition – none

סיבוכיות – O(1)

תיאור קצר – נגדיר את הגודל של הצומת עליה מופעלת הפונקציה להיות size.

```
updetSize()
```

```
setRank(int rank)
```

return type – none

precondition – none

postcondition – none

סיבוכיות – O(1)

תיאור קצר – נגדיר את הדרגה של הצומת עליה מופעלת הפונקציה להיות rank.

```
setValue (String value)
```

return type – none

precondition – none

postcondition – none

סיבוכיות – O(1)

תיאור קצר – נגדיר את הערך של הצומת עליה מופעלת הפונקציה להיות value.

```
setKey (int key)
```

return type – none

precondition – none

postcondition – none

סיבוכיות – O(1)

תיאור קצר – נגדיר את המפתח של הצומת עליה מופעלת הפונקציה להיות key.

מדידות

ניסוי 1:

מספר סידורי	מספר פעולות	מספר פעולות האיזון הממוצע לפעולת insert	מספר פעולות האיזון הממוצע לפעולת delete	מספר פעולות האיזון המקסימלי לפעולת insert	מספר פעולות האיזון המקסימלי לפעולת delete
1	10,000	2.591267	1.875233	15	20
2	20,000	2.597600	1.880900	16	20
3	30,000	2.602856	1.874933	17	21
4	40,000	2.594833	1.878158	18	20
5	50,000	2.599060	1.877613	19	23
6	60,000	2.598578	1.878956	18	23
7	70,000	2.604829	1.879648	19	25
8	80,000	2.596504	1.880808	19	24
9	90,000	2.600111	1.880900	19	22
10	100,000	2.593017	1.879920	19	23

ניסוי 2:

מספר סידורי	עלות join ממוצע עבור split אקראי	עלות join מקסימלי עבור split אקראי	עלות join ממוצע עבור split של איבר מקס בתת העץ השמאלי	עלות join מקסימלי עבור split של איבר מקס בתת העץ השמאלי
1	2.662	5	2.83	16
2	2.3	6	2.6	17
3	2.7	4	2.62	19
4	2.57	5	2.82	18
5	2.45	6	2.73	19
6	2.69	5	2.846	19
7	2.412	6	2.788	19
8	2.8	7	2.376	20
9	2.9	7	2.3	19
10	2.927	8	2.934	20