<u>חלק תיאורטי-שמות המגישים:</u>

rudnitzky -(20662787820) תומר רודניצקי

idorosiner -(209617000) עידו רוזינר

:AVLNode להלן תיעוד

פונקציה	תיאור	חישוב
		סיבוכיו
4.0		ת זמן
getLeft()	הפונקציה מחזירה את הבן השמאלי של הצומת. מוגדר	0(1)
	להיות None במקרה והצומת וירטואלית. – עידו רוזינר	
getRight()	הפונקציה מחזירה את הבן הימני של הצומת. מוגדר	0(1)
	להיות None במקרה והצומת וירטואלית. – עידו רוזינר	
getParent()	הפונקציה מחזירה את ההורה של הצומת. מוגדר להיות	0(1)
	None במקרה והצומת וירטואלית– עידו רוזינר	
getValue()	הפונקציה מחזירה את הערך של הצומת. מוגדר להיות	0(1)
	None במקרה והצומת וירטואלית. – עידו רוזינר	
getHeight()	הפונקציה מחזירה את הגובה של הצומת. מוגדר להיות	0(1)
	1- במקרה והצומת וירטואלית. – עידו רוזינר	
getSize()	הפונקציה מחזירה את הגודל של הצומת. מוגדר להיות 0	0(1)
	במקרה והצומת וירטואלית. – עידו רוזינר	
getBF()	– הפונקציה מחזירה את ה-balance factor של הצומת.	0(1)
	עידו רוזינר	
setLeft(node)	הפונקציה מקבעת את הבן השמאלי של הצומת להיות	0(1)
	node. – עידו רוזינר	
setRight(node)		
	node. – עידו רוזינר	0(1)
setParent(node	– node הפונקציה מקבעת את ההורה של הצומת להיות	
)	עידו רוזינר	
setValue(value)	– .value הפונקציה מקבעת את הערך של הצומת להיות	0(1)
	עידו רוזינר	
setHeight(h)	הפונקציה מקבעת את הגובה של הצומת להיות h. – עידו	0(1)
	רוזינר	
setSize(s)	הפונקציה מקבעת את הגודל של הצומת להיות s. – עידו	0(1)
	רוזינר	
setBF(bf)	הפונקציה מקבעת את ה-balance factor של הצומת	0(1)
	להיות bf. – עידו רוזינר	
isRealNode()	הפונקציה מחזירה True אם הצומת היא לא צומת	0(1)
	וירטואלית (הגובה שונה מ-(1-)) ואחרת False. – עידו	
	רוזינר	
computeBF()	של balance factor-הפונקציה מחשבת ומחזירה את	0(1)
	הצומת. כלומר, מחשבת ומחזירה	
	1 + height(leftSubtree) - (1 + height(rightSubtree))	
	עידו רוזינר –	
computeHeight	הפונקציה מחשבת ומחזירה את הגובה של הצומת.	O(1)
()	כלומר, מחשבת ומחזירה	

	1 + max{height(leftSubtree), height(rightSubtree) - עידו רוזינר	
computeSize()	הפונקציה מחשבת ומחזירה את הגודל של הצומת. כלומר, מחשבת ומחזירה את מספר הצמתים שקטנים שווים לה – 1 + size(leftSubtree) + size(rightSubtree) – עידו רוזינר	0(1)

:AVLTreeList להלן תיעוד

פונקציה	תיאור	חישוב סיבוכיות זמן
getRoot()	הפונקציה מחזירה את	0(1)
	השורש של העץ. מוחזר	
	AVLNode(None)	
	_ במקרה והעץ וירטואלי.	
	עידו רוזינר	
getMax()	הפונקציה מחזירה את	0(1)
3 "	י הצומת המקסימלית בעץ.	
	מוחזר None במקרה	
	י והעץ וירטואלי– עידו	
	רוזינר	
getMin()	הפונקציה מחזירה את	0(1)
3 (/	י הצומת המינימלית בעץ.	
	מוחזר None במקרה	
	והעץ וירטואלי. – עידו	
	רוזינר	
getSize()	הפונקציה מחזירה את	0(1)
95.5.25()	גודל העץ - מספר	
	הצמתים הלא וירטואליים	
	שנמצאים בעץ. מוחזר 0	
	במקרה והעץ וירטואלי. –	
	עידו רוזינר	
setRoot(node)	הפונקציה מקבעת את	0(1)
	השורש של העץ להיות	
	יוען זוז וונ o o node. – עידו רוזינר	
setMax(node)	הפונקציה מקבעת את	0(1)
octiviax(nede)	הצומת המקסימלית בעץ	
	יובונות node. – עידו	
	רוזינר	
setMin(node)	הפונקציה מקבעת את	0(1)
	הצומת המינימלית בעץ	
	יובונות node. – עידו	
	רוזינר	
setSize(s)	רוז נו הפונקציה מקבעת את	0(1)
30(0)20(3)	הגודל של העץ להיות s.	
	רוגורז פל חען לדרווניט. – עידו רוזינר	
empty()	True הפונקציה מחזירה	0(1)
Citipty()	וופונקב זו מוודדוד שחדר אם הרשימה (העץ) ריקה	
	אם דוו שימוד (דועץ) דיקוד (עץ וירטואלי) ואחרת	
	ען דרטואי) אווויזנ False – עידו רוזינר	
retrieve(i)	ו. – ע דודוו נו הפונקציה מחזירה את	- הפונקציה קוראת ל
1611676(1)	וופונקציון מווזין וו אונ ערך האיבר במקום ה-i	יופונון אוני - treeSelect(self, i+1), שרצה ב-
	ערן וואיבו בנקום וו-ו אם קיים, אחרת היא	ובין, וופפטפופטנו(seii, ווידו), וופטפופטנו(seii, ווידו $O(\log i)$
	אם קיים, אחדות חיא מחזירה None. – תומר	$(\log t)$ ט, וכל שאו הפעלוונ קורות ב- $O(1)$. לכן הפונקציה
incort(i vol)	רודניצקי	רצה ב-0(log i).
insert(i, val)	הפונקציה מכניסה איבר	תחילה הפונקציה קוראת ל- (trooSoloct/i) אונגעה בסובונות
	בעל ערך val לרשימה	treeSelect(i) שרצה בסיבוכיות
		$.0(\log i)$

	במקום ה-i במידה וקיימים	לאחר מכן הפונקציה קוראת ל-
	לפחותו איברים ברשימה.	שרצה rebalanceTree(node)
	הפונקציה מחזירה את	בסיבוכיות $O(\log n)$. מכיוון שכל
	מספר פעולות האיזון	O(1)-שאר הפעולות קורות ב
	שנדרשו בשלב תיקון העץ	הפונקציה רצה בסיבוכיות
	על מנת לשמר את תכונת	$O(\log n)$
	האיזון. – תומר רודניצקי	
delete(i)	הפונקציה מוחקת את	תחילה הפונקציה קוראת ל-
.,	האיבר במקום ה-i	יית treeSelect(i) שרצה בסיבוכיות
	ברשימה, אם הוא קיים.	או לפונקציה $O(\log i)$
	הפונקציה מחזירה את	שרצה בסיבוכיות successor(x)
	מספר פעולות האיזון	$O(\log n)$
	שנדרשו בשלב תיקון העץ	לאחר מכן הפונקציה קוראת ל-
	על מנת לשמר את תכונת	delete_node(node_to_del)
	האיזון.	,rebalanceTree(node)
	יהי הן. אם לא קיימים מספיק	שרצות בסיבוכיות $O(\log n)$, אחד
	איברים ברשימה איברים ברשימה	אחרי השני. מכיוון שכל שאר
	הפונקציה מחזירה 1 –	אווו רוסני. מל וון סלי טאו הפעולות קורות ב- $0(1)$,
	עידו רוזינר	הפונקציה רצה בסיבוכיות
		$O(\log n)$
doloto nodo(nodo to dol)	בפונדעוב מוסדת עת	. <i>ס</i> (<u>log <i>it)</i></u> אם node_to_del בעל שני ילדים
delete_node(node_to_del)	הפונקציה מוחקת את	
	node_to_del הצומת	אמיתיים הפונקציה קוראת ל-
	מהעץ ומחזירה את	successor(node), שרצה
	הצומת שממנה נתחיל את	בסיבוכיות $O(\log n)$, לכל היותר
	פעולת האיזון של העץ. – בידיר	פעמים. מכיוון שכל שאר $ hinspace 0(1)$
	עידו רוזינר	הפעולות בפונקציה קורות ב-
		הפונקציה רצה בסיבוכיות $O(1)$
		$O(\log n)$
first()	הפונקציה מחזירה את	0(1)
	ערך האיבר הראשון 	
	ברשימה, או None	
	ברשימה ריקה. – עידו	
	רוזינר	
last()	הפונקציה מחזירה את	0(1)
	ערך האיבר האחרון	
	ברשימה, או None	
	ברשימה ריקה. – עידו	
	רוזינר	
listToArray()	הפונקציה מחזירה מערך	הפונקציה קוראת לפונקציה
- "	המכיל את הערכים של ·	שרצה inorder(node, L)
	איברי הרשימה (העץ) לפי	בסיבוכיות $O(n)$. מכיוון שכל שאר
	יי. סדר האינדקסים (-in	הפעולות קורות ב- $\dot{O}(1)$,
	או מערך ריק אם (order	הפונקציה רצה בסיבוכיות $O(n)$.
	הרשימה ריקה. – עידו	= ::= : :: = /=:= ::
	I -	
	11111	
inorder(node 1.)	רוזינר הפונקציה מכניסה	ההכנסה מבוצעת כמו באלגוריתם
inorder(node, L)	הפונקציה מכניסה	ההכנסה מבוצעת כמו באלגוריתם in-order שנלמד בשיעור (במקוח
inorder(node, L)	הפונקציה מכניסה לרשימה L את הערכים	שנלמד בשיעור (במקום in-order
inorder(node, L)	הפונקציה מכניסה לרשימה L את הערכים של איברי הרשימה (העץ)	in-order שנלמד בשיעור (במקום הדפסה נוסיף את הערך של
inorder(node, L)	הפונקציה מכניסה לרשימה L את הערכים	שנלמד בשיעור (במקום in-order

	ידי ריצה in-order על	
1	איברי העץ. – עידו רוזינר	0(1)
length()	הפונקציה מחזירה את	0(1)
	מספר האיברים ברשימה.	
	עידו רוזינר –	L
sort()	הפונקציה מסדרת את	תחילה הפונקציה קוראת ל-
	איברי הרשימה בסדר	listToArray(self), שרצה
	עולה ומחזירה רשימה	בסיבוכיות $O(n)$.
	(עץ) חדשה. – תומר	לאחר מכן הפונקציה מפרידה את כל האיברים שהם None
	רודניצקי	כל האיבו ים שהם שחטוו מהאיברים שהם לא None
		בסיבוכיות $O(n)$. לאחר מכן הפונקציה קוראת ל-
		יאווו מכן וופונקציו קוו אוניי- merge_sort(lst) שרצה
		בסיבוכיות $O(n \cdot \log n)$.
		בס בול וול (אונים און ס. אחר כך הפונקציה קוראת ל-
		buildTreeFromList(lst, first,
		O(n) שרצה בסיבוכיות last)
		ושומרת את הצומת שהתקבלה
		כשורש של AVLTreeList חדש.
		לבסוף הפונקציה מעדכנת את
		השדות של ה-AVLTreeList
		החדש. לשם כך הפונקציה קוראת
		ל- (max_node(node)
		אשר רצות min_node(node)
		בסיבוכיות $O(\log n)$. סה"כ נקבל
		כי הפונקציה (sort(self רצה
		בסיבוכיות $O(n \cdot \log n)$.
permutation()	הפונקציה מחזירה רשימה	תחילה הפונקציה קוראת ל-
	עץ) חדשה המכילה את	שרצה listToArray(self)
	אותם האיברים של	בסיבוכיות $O(n)$.
	הרשימה (העץ) הנוכחי	לאחר מכן הפונקציה קוראת
	– בסדר אקראי. (self)	לפונקציה (shuffle(lst, שרצה
	עידו רוזינר	בסיבוכיות $O(n)$.
		אחר כך הפונקציה קוראת ל-
		buildTreeFromList(lst, first,
		O(n) שרצה בסיבוכיות last)
		ושומרת את הצומת שהתקבלה
		כשורש של AVLTreeList חדש.
		לבסוף הפונקציה מעדכנת את השדות של ה-AVLTreeList
		השו ות של ה-AVLTreeList החדש. לשם כך הפונקציה קוראת
		ל- (max_node(node ו-
		אשר רצות min_node(node)
		בסיבוכיות $O(\log n)$. סה"כ נקבל
		בסיבוכיות (10g 10) ט. טוד כ מובר כי הפונקציה (permutation(self
		רצה בסיבוכיות $O(n)$.
concat(lst)	הפונקציה מקבלת רשימה	רצוז בסיבוליות (אי) ט. במקרה שבו lst או self ריקים
	וופונקציון נמןביוניו שימוד ומשרשר אותה אל סוף	בניון דו שבו זכו או וושכדי קים הפונקציה מחברת את העצים
	ומשו שו אוומדאז טוף הרשימה הנוכחית.	וופונוןב זו נווובו זו אונ וועב ם (אם יש מה לחבר) ומסיימת לרוץ
<u> </u>	ווו ס בווו וובוכוו ונ.	1.11.3113 0131 (12111 1113 0 111)

	הפונקציה מחזירה את	– פעולה זו קוראת בסיבוכיות
	הערך המוחלט של הפרש	.0(1)
	הגבהים של עצי ה-AVL	אם האורך של אחד העצים הוא 1
	שמוזגו. – עידו רוזינר	לאיבר insert לאיבר
		של העץ בעל האיבר היחיד אל
		תוך העץ השני ותסיים את הריצה
		$O(\log n)$ יבוכיות.
		אחרת, הפונקציה שומרת את
		אווורג, וופונון ביו סונוו זג אונ האיבר המקסימלי של self
		delete(self.length() וקוראת ל
		_ :
		$O(\log n)$ טיבוכיות – 0).
		כעת, הפונקציה משווה בין הגובה
		של העצים (לאחר המחיקה)
		ומחברת אותם בהתאם:
		אם הגובה של self שווה לגובה
		של lst הפונקציה קוראת ל-
		joinSelfSame(self, x, lst)
		שרצה בסיבוכיות $0(1)$.
		אם הגובה של self גדול מהגובה
		של Ist הפונקציה קוראת ל-
		joinSelfBigger(self, x, lst)
		שרצה בסיבוכיות $O(\log n)$. אם
		פו בול בול וול (נוקסו) אם self הגובה של
		ווגובוז פל וופג קופן מוזגובוז פל Ist הפונקציה קוראת ל-
		joinSelfSmaller(self, x, lst)
		שרצה בסיבוכיות $O(\log n)$.
		לבסוף הפונקציה תעדכן את
		השדות של העץ (סיבוכיות $0(1)$
		ותקרא ל יי.
		כאשר rebalanceTree(node)
		node הוא הצומת המקסימלית
		שמחקנו ושמרנו מקודם –
		$O(\log n)$ סיבוכיות
		סה"כ הפונקציה רצה בסיבוכיות
		$O(\log n)$
joinSelfBigger(x, lst)	הפונקציה מחברת את	הפונקציה מניחה כי גובה העץ
,	העץ ל-self דרך הצומת	self גדול מגובה העץ self
	יוען די ווסס דרך דובוניונ. X – עידו רוזינר	את העצים דרך הצומת x כנלמד
	א. עוווונו	אול וועב בי דרך דובונות א פנדנוו בשיעור (הפונקציה Join
		I
inin O - KO U - / - U - ($O(\log n)$ מהשיעור – סיבוכיות
joinSelfSmaller(x, lst)	הפונקציה מחברת את	הפונקציה מניחה כי גובה העץ
	העץ ל-self דרך הצומת	קטן מגובה העץ lst קטן מגובה self
	x. – עידו רוזינר	את העציִם דרך הצומת x כנלמד
		בשיעור (הפונקציה Join
		$O(\log n)$ מהשיעור – סיבוכיות
joinSelfSame(x, lst)	הפונקציה מחברת את	הפונקציה מניחה כי גובה העץ
, , - ,	י העץ ל-self דרך הצומת	self שווה לגובה העץ st
	אידן רוזינר	ומחברת את העצים דרך הצומת
	א. עוווונו	ונווזברונ אונ וועב ם דרך וזבונות x כנלמד בשיעור (הפונקציה Join
		א כנ <i>ו</i> נוו בשיעוו (וופונקציו וווטט מהשיעור). נשים לב כי הפעם אין
		[אן ושיעוו א. נשים זב כ וופעם אן

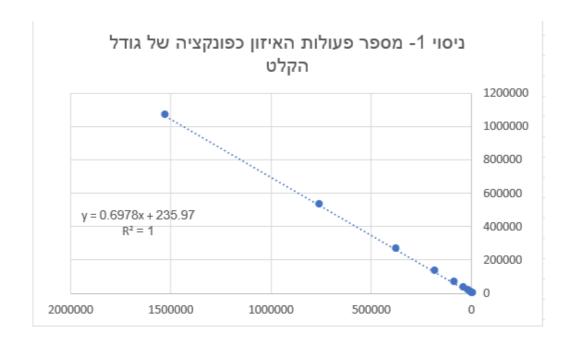
		צורך לרדת במורד אף עץ או לאזן
		יותר מצומת אחת בעץ, ולכן
1.7		O(1) הפונקציה רצה בסיבוכיות
search(val)	הפונקציה מחזירה את	תחילה הפונקציה קוראת ל-
	האינדקס הראשון	סיבוכיות – listToArray(self)
	ברשימה בו מופיע הערך	0(n)
	או 1- אם לא קיים val	ולאחר מכן הפונקציה רצה
	אינדקס כזה. – עידו	– בלולאת for על איברי הרשימה
	רוזינר	O(n) סיבוכיות
		סה"כ הפונקציה רצה בסיבוכיות
		.0(n)
rotateRight(node)	הפונקציה מבצעת סיבוב	הפונקציה פועלת כמעט כמו
5 , ,	ימינה על node. – תומר	שנלמד בהרצאה (לאחר
	רודניצקי	המודיפיקציה של הוספת השדה
	•	height-ı size לצמתים). ההבדל
		היחיד הוא הבדיקה של האם
		node הוא השורש של העץ. אם
		הוא השורש של העץ נעדכן node
		את השורש להיות הבן השמאלי
		של node – סיבוכיות $0(1)$.
		סה"כ הפונקציה רצה בסיבוכיות
		.0(1)
rotateLeft(node)	הפונקציה מבצעת סיבוב	כמעט כמו שנלמד בהרצאה
rotateLent(node)	וופונוןב וו מבצעוניט בוב שמאלה על node. –	לנועט כנוו סנזנוו בווו באוו (לאחר המודיפיקציה של הוספת
	סנארוז עז שחחו. תומר רודניצקי	(יאווו רומוו פיוןבירי פיי חוספונ השדה size ו-height
	1,22111 1,2131	ההבדל היחיד הוא הבדיקה של
		node האם node הוא השורש של העץ.
		אם node הוא השורש של העץ
		אם שבוח הווא חיפור פידי חיקן נעדכן את השורש להיות הבן
		נערכן אול חסורס זודות הימני של node – סיבוכיות
		וו נוני פין הוסטרוייט בול ווני פייט ווייט פייט הייט הייט הפונקציה רצה $O(1)$
		O(1)ט. טוז ל וופונון ביו די בוו בסיבוכיות $O(1)$.
rebalanceTree(node)	בפונדעוב מעזנת עת בעע	בט בול וונ (ד) ט. הפונקציה זהה לחלק של האיזון
rebalance riee(node)	הפונקציה מאזנת את העץ ומעדכנת את השדות כמו	וופונקציוז ווווד לוולק של וואיוון של העץ של הפונקציה
	ונעו כנונ אונ וושו וונ כמו שנלמד בהרצאה. – תומר	של וועץ של וופונקציוו Delete(D,x) שנלמדה בכיתה
	רודניצקי	(לאחר המודיפיקציה שלכל צומת
		קיים שדה size) - סיבוכיות
tro o C a la at/i)		$O(\log n)$
treeSelect(i)	הפונקציות מחזירות את	הפונקציות זהות לפונקציית או מאסוס אינלמדה ברותה
treeSelectHelper(node, i)	האיבר ה-i הכי קטן בעץ	Select(D, k) שנלמדה בכיתה
	כמו שנלמד בכיתה (לאחר	לאחר המודיפיקציה שמתחילים)
	המודיפיקציה של התחלת	את החיפוש בעץ מהצומת
	החיפוש מהאיבר	$O(\log i)$ המינימלית) - סיבוכיות
	המינימלי). – תומר	
	רודניצקי	
max_node(node)	הפונקציה מחזירה את	הפונקציה יוִרדת מהשורש של
	-האיבר המקסימלי בתת	התת עץ אל הצומת המקסימלית
	עץ שבו node הוא	שלו במסלול ישיר ולכן הסיבוכיות
	השורש. – תומר רודניצקי	$O(\log n)$ היא

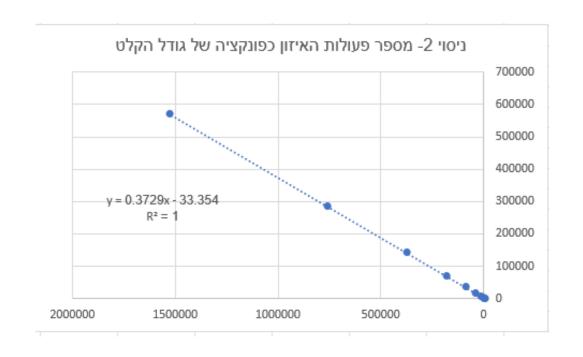
	T	
min_node(node)	הפונקציה מחזירה את	הפונקציה יורדת מהשורש של
	האיבר המינימלי בתת-עץ	התת עץ אל הצומת המינימלית
	שבו node הוא השורש. –	שלו במסלול ישיר ולכן הסיבוכיות
	תומר רודניצקי	$O(\log n)$ היא
predecessor(x)	הפונקציה מחזירה את	הפונקציה זהה לפונקציית
	האיבר הקודם ברשימה.	שנלמדה Predecessor(D, x)
	כלומר, אם x הוא האיבר	$O(\log n)$ בשיעור - סיבוכיות
	במקום ה-i הפונקציה	
	.i-1-תחזיר את האיבר ה	
	תומר רודניצקי –	
successor(x)	הפונקציה מחזירה את	הפונקציה זהה לפונקציית
	האיבר הבא ברשימה.	שנלמדה בשיעור Successor(D, x)
	כלומר, אם x הוא האיבר	. $O(\log n)$ סיבוכיות -
	במקום ה-i הפונקציה	
	.i+1.תחזיר את האיבר ה	
	עידו רוזינר –	
buildTreeFromList(lst, left,	הפונקציה מקבלת רשימה	כל פעם הפונקציה מבצעת שתי
right)	של ערכים ומחזירה צומת	קריאות רקורסיביות על שני
	שמהווה שורש לעץ	חצאים שונים של הרשימה (אנו
	הנוצר AVLTreeList	עושים זאת באמצעות מצביעים
	מהרשימה. – עידו רוזינר	כך שלא תיווסף סיבוכיות) ולכן
		$\log n$ עומק עץ הרקורסיה יהיה
		כאשר בכל קריאה מתבצעת
		עבודה. על כן, סיבוכיות $\mathit{O}(1)$
		הפונקציה היא
		$2^{\log n} = O(n)$
merge_sort(lst)	הפונקציות מחזירות	mergeSort הקוד זהה לקוד של
merge(lst1, lst2)	רשימה מסודרת בסדר	שלמדנו בקורס מבוא מורחב
	עולה. – תומר רודניצקי	$O(n \cdot \log n)$ – למדעי המחשב
Shuffle(lst)	הפונקציה "מבלגנת" את	הפונקציה עוברת על הרשימה
	in- הרשימה באופן אקראי	וכל פעם מחליפה בין for בלולאות
	place. – עידו רוזינר	O(n) - שני איברי הרשימה
		, ,

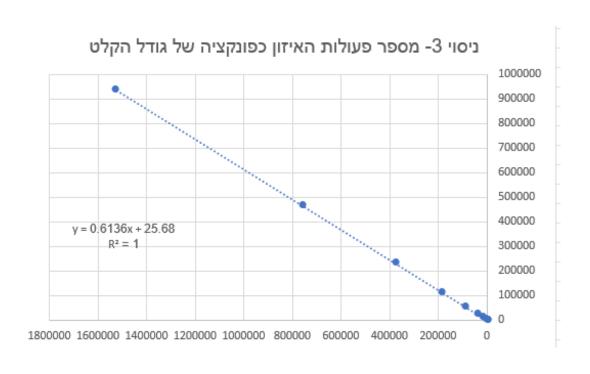
1.1. להלן טבלה המתארת את מספר פעולות האיזון שנדרשו כדי לתקן את העץ בכל ניסוי:

ניסוי 3- הכנסות ומחיקות לסירוגין	ניסוי 2- מחיקות	ניסוי 1- הכנסות	i
1830	1146	2146	1
3699	2143	4130	2
7367	4478	8332	3
14625	8933	16817	4
29600	17812	33467	5
58789	35793	67365	6
117405	71788	134828	7
236183	142998	268134	8
471642	286055	537312	9
942261	572806	1071498	10

2.1. להלן תרשימים המתארים את תוצאות הניסוי בצורה גראפית. הוספנו קו מגמה, מדד R^2 ומשוואה המתארת בקירוב את הקו. מכל אלה המסקנה היא כי הביטוי האסימפטומטי הוא (o(n) בשלושת הניסויים:







2. להלן שלושת הטבלאות המתארות את זמן הריצה בממוצע עבור עץ AVI, רשימה מקושרת ועבור מערך:

זמן ריצה- מערך הכנסות להתחלה	זמן ריצה- רשימה מקושרת הכנסות להתחלה	זמן ריצה- עץ AVL הכנסות להתחלה	i
3.295837e-07	5.654738e-07	3.664356e-05	1
5.545469e-07	6.545347e-07	2.923434e-05	2
7.467864e-07	6.878794e-07	3.192972e-05	3
9.565846e-07	6.564767e-07	3.123943e-05	4
1.113233e-06	6.747463e-07	3.225311e-05	5
1.515454e-06	6.657457e-07	3.487298e-05	6
1.724939e-06	6.754744e-07	3.123936e-05	7
1.835562e-06	6.336785e-07	3.286947e-05	8
1.964126e-06	6.112436e-07	3.112393e-05	9
2.021410e-06	6.868746e-07	3.133183e-05	10

זמן ריצה- מערך הכנסות אקראיות	זמן ריצה- רשימה מקושרת הכנסות	AVL זמן ריצה- עץ הכנסות אקראיות	i
•	אקראיות [.]	·	
8.469857e-07	1.775654e-05	4.129876e-05	1
8.965450e-07	3.748905e-05	3.700732e-05	2
1.193485e-06	7.969767e-05	3.774336e-05	3
1.327493e-06	9.021049e-04	4.012432e-05	4
1.344458e-06	1.156784e-04	4.498348e-05	5
1.230485e-06	1.331895e-04	4.744344e-05	6
1.446758e-06	1.567349e-04	4.425390e-05	7
1.546281e-06	1.936695e-04	5.345783e-05	8
1.748480e-06	2.188605e-04	4.745946e-05	9
1.645678e-06	2.324536e-04	5.449583e-05	10

זמן ריצה- מערך הכנסות בסוף	זמן ריצה- רשימה מקושרת הכנסות בסוף	זמן ריצה- עץ AVL הכנסות בסוף	i – מספר סידורי
2.241384e-07	5.812048e-07	2.802437e-05	1
1.934423e-07	6.458395e-07	2.923845e-05	2
1.748259e-07	6.835368e-07	3.001652e-05	3
1.539534e-07	6.412367e-07	3.168567e-05	4
1.934423e-07	6.503696e-07	3.432853e-05	5
1.493735e-07	6.436063e-07	3.023855e-05	6
1.495953e-07	6.130633e-07	3.359357e-05	7
1.495560e-07	6.543698e-07	3.004939e-05	8
1.535794e-07	6.663460e-07	3.583897e-05	9
1.569639e-07	6.346602e-07	3.668567e-05	10

לפני תחילת הניסוי, היינו מצפים כי התוצאות האמיתיות יהיו דומות לאלה שיצאו בסופו של דבר. ברשימה מקושרת, הכנסות לתחילתה ולסופה מתבצעות ב(0) (זוהי רשימה עם מצביע לסוף) ולכן התקבלו תוצאות מהירות יותר מאשר בהכנסות לסוף ולתחילת עץ AVL. מנגד, בהכנסות אקראיות ברשימה מקושרת "נאלץ" לעבור על חלק גדול מהרשימה בכל הכנסה((n) במקרה הגרוע) ולכן זמן הריצה היה איטי יותר. נשאלת השאלה מדוע במערך התקבלו התוצאות הטובות ביותר בכמעט כל המקרים? ובכן, המימוש של פייתון למערך משודרג ויעיל, ככה שקשה להתחרות בו. בהכנסות אקראיות למשל, כשהמערך מתמלא, למדנו בהרצאה כי למרות שמדי פעם ייווצר מערך חדש הגדול פי α + 1 (הערך הדיפולטי של α מוגדר להיות 1/8 בפייתון) מהמערך הישן, דבר שיוביל לפעולה בגודל (o(n), סיבוכיות ממחרדובות היי (1).

הערה: במידה והיינו ממשים רשימה מקושרת ללא מצביע לסוף, אנו סבורים כי זמן הריצה היה האיטי ביותר בהכנסות לסוף, שכן במצב כזה בכל הכנסה היינו נאלצים לעבור על כל איבר ואיבר ברשימה((n)) ולא במקרה הגרוע, אלא תמיד!).