מבני נתונים קיץ 2019
תרגיל בית 2

תאריך הגשה: 4 בספטמבר 2019

את העבודה יש להגיש במערכת ההגשות submission system
על העבודה להיות מוגשות בכתב יד ברור סרוק או הקלדה במחשב

- הנכם נדרשים לנסח **תשובות ברורות וקצרות**
- עליכם לנסח תחילה את האלגוריתם במילים בכמה משפטים
- לאחר מכן בפסאדו קוד (לא בשפת תכנות אלא כמו שלמדנו בכיתה, ללא פרטים טכניים)
- לאחר מכן להסביר את נכונותו (אין דרישה להוכיח אך ההסבר חייב להיות **מאוד משכנע**)
- ולבסוף לנתח את זמן הריצה

שאלות לגבי העבודה ניתן לשאול בפורום של העבודה הנמצא באתר של הקורס.

שאלה 1:

תארו אלגוריתם יעיל ככל הניתן **לא רקורסיבי** המבצע מעבר inorder על עץ בינארי T.
יש להשתמש בלכל היותר $O(h)$ זיכרון נוסף כאשר h הוא גובה העץ.

שאלה 2:

א. עץ T נקרא כמעט BST אם קיימים בדיוק שני קדקודים ב T כך שאם נחליף ביניהם נקבל עץ BST חוקי.
הציעו אלגוריתם יעיל ככל הניתן לתיקון עץ שהינו כמעט BST ונתחו את זמן ריצתו.

ב. הציעו אלגוריתם יעיל ככל הניתן המשחזר עץ BST בהנתן סריקת preorder , ונתחו את זמן ריצתו.

שאלה 3:

עליכם לממש מערכת לניהול זיכרון עם N דפים (לכל היותר). פונים אל הדפים בזיכרון מתוך תוכנית כלשהי. מדיניות החלפת דפי הזיכרון הינה בהתאם לקריטריון הבא: הדף המסולק ברגע נתון הוא זה עם מספר הפניות הקטן ביותר מבין הדפים שבזיכרון. אם יש יותר מדף אחד עם מספר פניות מינימלי, אזי מבין כל הדפים עם מספר פניות מינימלי, יסולק מהזיכרון זה אשר זמן הפניה האחרונה אליו הוא הרחוק ביותר.
הציעו מבנה נתונים המבוסס על רשימות מקושרות התומך בפעולות הבאות בזמן הנתון:

שם פעולה	תאור פעולה	זמן ריצה במקרה גרוע ביותר
init()	אתחול מבנה הנתונים (בעת אתחול אין אף דף בזיכרון)	O(1)
insert(X)	הכנסת דף X לזיכרון (תוך סילוק דף קיים מהזיכרון במקרה שהזיכרון מלא, כלומר מכיל N דפים)	O(1)

reference(Y)	פניה אל הדף בזיכרון שאליו מתייחס המצביע Y	O(1)

על מבנה הנתונים להשתמש ב O(N) זיכרון בלבד.

שאלה 4:

הטענות הבאות מתייחסות **לעץ BST**.
עבור כל טענה ענו נכון או לא נכון ונמקו את תשובתכם.

א. אפשרי לשחזר עץ BST בעזרת סריקת preorder בלבד
ב. ייתכן שהסדרה הבאה התקבלה ע"י הדפסת preorder (משמאל לימין) :
24, 51, 12, 4, 71, 02, 31, 23, 37
ב. ההדפסות בסדר preorder ו postorder תמיד הפוכות זו לזו בסדר.
ג. יהי x צומת. לא יתכן של SUCCESSOR של x יש בן שמאלי
ד. אם x אינו שורש ויש לו שני בנים, ייתכן שהמפתח של אביו של x גדול מהמפתחות של שני בניו של x.
ה. אם y נכנס לעץ (ע"י פעולת INSERT) אחרי x והמפתח של y גדול מהמפתח של x אזי בהכרח y יהיה בתת העץ ימני של x.
ו. ייתכן שבחיפוש אחרי המפתח 24 נסרוק לפי הסדר קודקודים עם מפתחות (משמאל לימין) :
12,18,20,100,80,24
ז. אם המפתח של x הינו 20 והבן הימני של x הינו 25 , אזי ייתכן שהמפתח 21 ימצא באביו של x

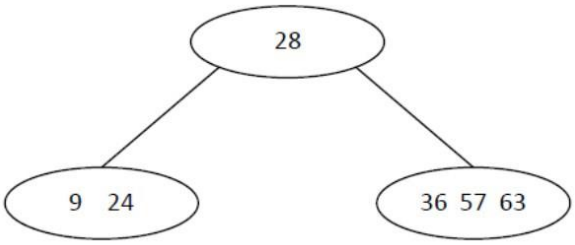
הטענות הבאות מתייחסות **לעץ AVL**.
עבור כל טענה ענו נכון או לא נכון ונמקו את תשובתכם.

א. אם T עץ AVL בגובה h, אז כל הרמות עד הרמה ה-$h-2$ (כולל) מלאות.
<u>הגדרה</u>: הרמה ה i של עץ היא אוסף כל הצמתים בעומק i.
נאמר שהרמה ה i מלאה אם מספר האיברים בה הוא 2^i.
ב. המספר המינימאלי של צמתים בעץ AVL בגובה h=4 הוא 12
ג. עבור AVL בגובה h עם n עלים מתקיים כי: $n \geq Fib(h)$ כאשר Fib(h) הינו האיבר ה h של סדרת הפיבונאצי .
להזכירכם הנוסחא של פיבונאצ'י: $Fib(0) =0, Fib(1) =1, Fib(n) = Fib(n-1) + Fib(n-2)$

הסעיפים הבאים מתייחסים לעץ **B-tree**:

א. נתונים 44,000,000 איברים. אנו רוצים לאחסן אותם ב B-tree -כך שגובה העץ הוא לכל היותר 5. מהו ערך t המינימאלי שניתן לבחור? הניחו כי כל האיברים ידועים לכם. נמקו את תשובתכם .

ב. בציור נתון B-tree עם .t=2 ציירו את מצב העץ לאחר כל אחת מהפעולות הבאות :
insert(30),insert(40),insert(4),delete(9),delete(24)



ג. הציעו גרסה "משופרת" של עץ B-Tree אשר תיהי בעלת פעולת אבסטרקטית חדשה. פעולה בשם order(k)- בהינתן מפתח k בעץ מחזירה בזמן O(logn) את מקומו בסדר הממוין של כל מפתחות העץ. הראו שהתוספת אינה פוגעת בזמני הריצה של הפעולות אבסטרקטיות הרגילות של על העץ. ניתן להישתמש ב O(n) זיכרון נוסף n, מספר המפתחות בעץ.

שאלה 5:

הציעו מבנה נתונים המחזיק מספרים שלמים ותומך בפעולות הבאות:

הפעולה	**תיאור הפעולה**	**זמן ריצה נדרש**
Init()	אתחול מבנה הנתונים (בהתחלה המבנה נתונים **ריק**)	$O(1)$
Insert(x)	הכנסת מספר חדש x למבנה הנתונים	$O(\log n)$
Delete(x)	מחיקת המספר x מהמבנה נתונים	$O(\log n)$
Search(x)	בדיקה האם המספר x נמצא במבנה הנתונים . הפונקציה תחזיר" Yes" או" No".	$O(\log n)$
Upgrade()	יהי y הערך המינימלי במבנה הנתונים, לאחר הקריאה לפעולה, כל המספרים יגדלו ב - $\|y\|$ (ערך מוחלט של y).	$O(1)$
Downgrade()	יהי y הערך המקסימלי במבנה הנתונים, לאחר הקריאה לפעולה, כל המספרים יקטנו ב - $\|y\|$ (ערך מוחלט של y).	$O(1)$

לדוגמה: לאחר הקריאה לפעולות הבאות:

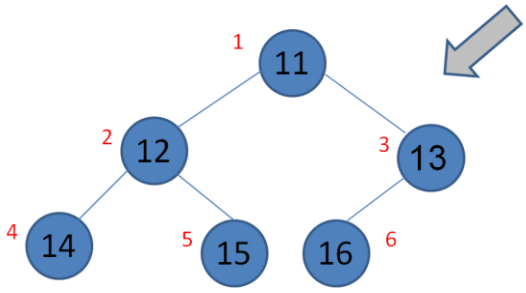
Init()
Insert(-1)
Insert(1)
Insert(4)
Insert(-2)
Insert(5)
Upgrade()
Search(7)
Search(-1)
Downgrade()

Search(-7)
יודפס:
Yes
No
Yes

שאלה 6:

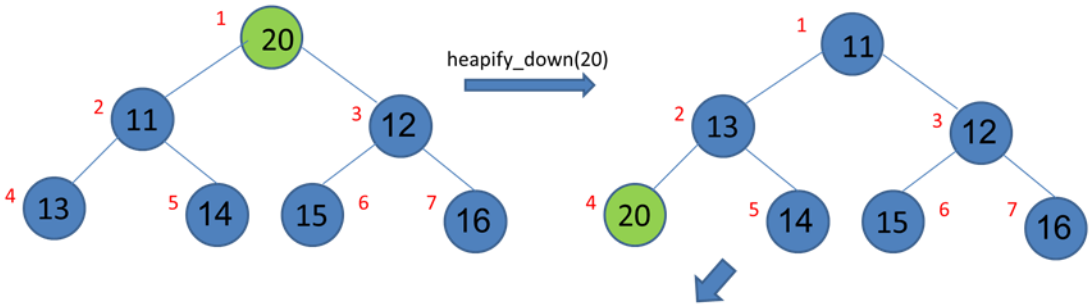
שני סטודנטים תומר ועומר קיבלו בשעורי בית לפתור שאלה הבאה: "הציעו אלגוריתם יעיל לבניית ערימת **מינימום** ממערך של n מספרים כאשר אם קוראים את המערך **מסוף להתחלה** הוא מהווה ערימת מינימום תקינה."
תומר אמר: "העובדה שהמערך ההפוך הוא ערימת מינימום לא עוזרת לנו. לדעתי אין ברירה אלא לבנות את הערימה מחדש בזמן O(n). "
עומר אמר: "בטח שזה עוזר, הרי זאת בכל זאת ערימה, פשוט בסדר הפוך! הינה האלגוריתם שאני מציע: לא נתייחס לאינדקסים האמיתיים של המספרים בערימה, אלה על כל אינדקס אמיתי i נחשב virtual index (i) = heap_size +1-i, ואילו יהיו אינדקסים שנשתמש בהם לערימת המינימום שלנו. לדוגמא:"

Index	1	2	3	4	5	6	
Virtual index	7-1=6	7-2=5	7-3=4	7-4=3	7-5=2	7-6=1	
Key	16	15	14	13	12	11	



תומר אמר: "אוקיי. אבל מה אם נרצה להכניס איבר חדש לערימה?"
עומר אמר:" אין בעיה. נכניס אותו לסוף המערך, ואז בעצם הוא הופך להיות לשורש הערימה. כעת , נעדכן את המשתנה heap_size++ ונבצע heapify_down לאיבר החדש שהכנסנו כדי לתקן את הערימה, ושוב פעם קיבלנו ערימת מינימום תקינה (הפוכה). לדוגמא:"

Index	1	2	3	4	5	6	7
Virtual index	8-1=7	8-2=6	8-3=5	8-4=4	8-5=3	8-6=2	8-7=1
Key	16	15	14	13	12	11	20



Index	1	2	3	4	5	6	7
Virtual index	8-1=7	8-2=6	8-3=5	8-4=4	8-5=3	8-6=2	8-7=1
Key	16	15	14	20	12	13	11

האם הפתרון של עומר עובד ? אם כן, הסבירו למה (אין צורך בהוכחה פורמלית אבל ההסבר צריך להיות ברור ומשכנע). אם לא, הביאו דוגמא נגדית.

שאלה 7:

הציעו מבנה נתונים עבור קבוצה S של מספרים שלמים התומך בפעולות הבאות:

הפעולה	תיאור הפעולה	זמן ריצה נדרש
Init()	אתחול מבנה הנתונים ריק	$O(1)$
search(x)	חיפוש מפתח x	$O(\log n)$
insert(x)	הכנסת x אם לא נמצא עדיין	$O(\log n)$
delete(x)	מחיקת x	$O(\log n)$
log_pair()	הפעולה מחזירה זוג מספרים a,b במבנה נתונים כך ש a=log(b) או b=log(a)	$O(1)$

שאלה 8:

א. בהנתן קבוצה סדורה של מספרים שלמים, הציעו אלגוריתם יעיל ככל הניתן המחזיר את המרחק המקסימלי בין שני מספרים זהים בקבוצה. האם ניתן לשפר את האלגוריתם עבור מקרה ממוצע ? אם כן, הציעו אלגוריתם לשיפור וחישבו את זמן ריצתו.
<u>לדוגמא</u>: בהנתן קבוצה סדורה (1 2 2 2 1 1), המרחק המקסימלי בין 1-ים הינו 5, והמרחק המקסימלי בין 2-ים הינו 2. לכן התשובה הינה 5.

ב. בהנתן קבוצת מספרים שלמים, הציעו אלגוריתם יעיל ככל הניתן המחזיר את הסדרה הארוכה ביותר של מספרים עוקבים בסדרה. האם ניתן לשפר את האלגוריתם עבור מקרה ממוצע ? אם כן, הציעו אלגוריתם לשיפור וחישבו את זמן ריצתו.
<u>לדוגמא</u>: בהנתן קבוצה {3 5 4 9 1 6 2 }, התת סדרה הארוכה של מספרים עוקבים בקבוצה הינה 6 5 4 3 2 1, כלומר תת סדרה באורך 6. שימו לב שבקבוצה המספרים של תת סדרה זו לא מופיעים כרצף.

ג. בהנתן קבוצת מספרים שלמים, הציעו אלגוריתם יעיל ככל הניתן למיין קבוצה לפי שכיחות המפתחות שבה. האם ניתן לשפר את האלגוריתם עבור מקרה ממוצע ? אם כן, הציעו אלגוריתם לשיפור וחישבו את זמן ריצתו.
<u>לדוגמא</u>: בהנתן קבוצה { 4 6 4 5 5 5 }, הקבוצה הממויינת לפי שכיחות תהיה 5 5 5 4 4 6, מכיוון שמספר 5 מופיע שלוש פעמים, מספר 4 מופיע פעמיים, ומספר 6 מופיע פעם אחת.

שאלה 9:

הציעו מבנה נתונים המאפשר את הפעולות הבאות על קבוצה S של n מספרים שונים מתוכם יש m מספרים "מיוחדים" ($m < n$).

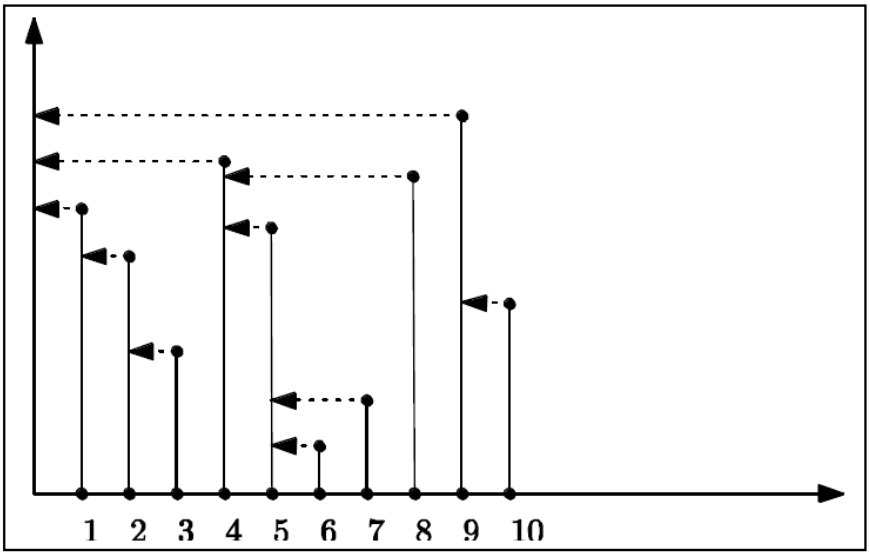
שם הפעולה	תאור הפעולה	זמן ריצה
add(k)	בהינתן מספר k (שידוע אם הוא "מיוחד") הוסף אותו לקבוצה S	O(n)
removeSmallest()	הוצא את האיבר הקטן ביותר בקבוצה S	O(1)
removeLargest()	הוצא את האיבר הגדול ביותר בקבוצה S	O(1)
oldest()	החזר את האיבר המיוחד הכי "ותיק" (לפי סדר ההכנסה)	O(1)

ניתן להשתמש ב O(n) זכרון למימוש המבנה.

שאלה 10:

נתונים n קטעים אנכיים $s_1, s_2, \dots, s_n$, כך שנקודת הקצה התחתונה של s_i היא הנקודה (i,0).
תארו אלגוריתם יעיל ככל הניתן המוצא לכל קטע s_i את הקטע שרואים מנקודת הקצה העליונה שלו כאשר מסתכלים שמאלה.

<u>דוגמה</u>:
בציור להל"ן: s_1 רואה את ציר ה – y, s_2 רואה את s_1, s_3 רואה את s_2, s_4 רואה את ציר ה – y וכו').



מבני נתונים - עבודה 2

מגישים: שגב נגר ואלמוג למאי

שאלה 1

פירוט האלגוריתם

- ניישם באמצעות מחסנית שתהיה אחראית לסידור איברי העץ על פי סדר ההדפסה ב-In order, ומצביע Current שיסייע לנו לעקוב אחר הצומת הנוכחי בעץ.
- האיברים השמאליים מקבלים תעדוף ב-In Order, לכן קטע הקוד יתחיל בהכנסתם למחסנית (שורה 4.1).
- לאחר שאין יותר בנים שמאליים, נדפיס את האיבר האחרון במחסנית ונוציא אותו (שורות 4.2-4.3)
- בסוף לולאת ה-While נטפל בבנים הימניים ונכניסם למחסנית (שורה 4.4). אותם לא נדפיס ישירות, מאחר שאם יש להם בנים שמאליים יש להדפיסם קודם. חלק זה יתבצע בשורה 4.1 כאשר לולאת ה-while תחזור על עצמה.

פסאודו קוד:

```
1.     Initialize Stack S
2.     Node current = root
3.     S.push(current)
4.     While S not empty
4.1. While current has left child
4.1.1.     current = current.left
4.1.2.     S.push(current)
4.2. Current = s.pop
4.3. Print current
4.4. if Current has right child
4.4.1.     Current = current.right
4.4.2.     S.push Current
```

ניתוח זמן ריצה

פעולות ה-Push וה-Pop ועדכון Current כולן פעולות קבועות המתבצעות ב-O(1)
ללולאת ה-While החיצונית ניכנס בסה"כ כמס' הצמתים שנכנסים למחסנית. אנו מכניסים כל צומת למחסנית פעם אחת, לכן בסה"כ הלולאה עולה O(n) וכך גם זמן הריצה של האלגוריתם.

ניתוח זיכרון

האלגוריתם בנוי כך שאנו יורדים לעבר כל עלה בנפרד כאשר צמתיו במחסנית, ולפני שאנחנו עוברים לעלה אחר אנו מדפיסים ומוציאים מהמחסנית את הצמתים שלא במסלול שלו. לכן בכל רגע נתון המספר המקסימלי של איברי המחסנית הוא כגובה העץ – כלומר זיכרון O(h) כנדרש.

שאלה 2

סעיף א'

פירוט האלגוריתם

- נסרוק את העץ שלנו סריקת InOrder ונשרשר את הערכים לפי הסדר למערך עזר בגודל n.
- נסרוק את המערך איבר-איבר, ונספור את כל האיברים שגדולים מהאיבר שקדם להם במערך.
- אם במהלך הסריקה הגענו לשלושה מספרים כאלה – סימן שלא ניתן לתקן את העץ ע"י החלפת 2 קודקודים בלבד, לכן נחזיר False. נחזיר True אחרת.

פסאודו קוד

```
1.    Create empty array in size N
2.    Perform InOrder traversal on the BST and copy the values to the array
3.    for i = 1 to n
      3.1.    if A[i] < A[i-1]
              3.1.1.    counter ++
      3.2.    if counter > 2
              3.2.1.    return false
4.    return true
```

ניתוח זמן ריצה

- יצירת מערך לא מאותחל - O(1)
- סריקת InOrder וההעתקה למערך – O(n)
- מעבר על איברי המערך – O(n)

Total of O(1) + O(n) + O(n) = O(n)

סעיף ב'

פירוט האלגוריתם

- לצורך מימוש הקוד, כל צומת בעץ שנבנה יחזיק שדה שיאחסן את הערך של קודקוד האב שלו.
- האלגוריתם עובר על המערך לפי הסדר ומוסיף כל איבר כצומת במקום המתאים בעץ.
- אם ערכו של האיבר הנוכחי במערך קטן מהערך של הצומת הנוכחי (הצומת Node שנשלח לפונקציה הרקורסיבית), האיבר יתווסף לעץ כבן שמאלי שלו.
- אם ערכו של האיבר גדול מהערך של הצומת הנוכחי וגם קטן מהערך של האבא של צומת זה, הוא יתווסף כבן ימני שלו.
- אם הוא גם גדול מהאבא - משמע שיש לחברו לאב קדמון כלשהו. תתבצע עליה ברמות הרקורסיה עד אשר נגיע לצומת שיקיים את הסעיף הקודם.

פסאודו קוד

```
1.    Create new root with value A[0];
2.    New global variable i =1
3.    PostOrderToTree(A, i node)

PostOrderToTree(A, Node)
1.    if A[i] < Node Value
      1.1.    Create NewNode holding A[i] value as left child to Node
      1.2.    i++
      1.3.    PostOrderToTree(A, NewNode)    // recursive call
2.    else
      2.1.    if A[i] < Parent node value
              2.1.1.    Create NewNode holding A[i] value as right child to Node
              2.1.2.    i++
              2.1.3.    PostOrderToTree(A, NewNode)    // recursive call
```

ניתוח זמן ריצה

בסה"כ יתבצעו n הוספות של איברים לעץ כמספר איברי המערך, לכן O(n)

שאלה 3

פירוט האלגוריתם

- נשתמש בשתי רשימות מקושרות עם מצביעים דו כיווניים מטיפוסים שונים:
  - רשימה מסוג 1 - המכילות חוליות מטיפוס INT שמשמען מס' פניות לזיכרון.
  - רשימה מסוג 2 - מכילות דפים.
- כל חוליה ברשימה 1 תפנה לרשימה מסוג 2 המאחסנת רק איברים בעלי מס' תואם של פניות לזיכרון.
- לכל חוליה ברשימה 1 יהיו מצביעים לאיבר הראשון ולאיבר האחרון ברשימה 2 הכפופה לה.
- **Init:** מאתחלת רשימה מקושרת ריקה.
- **Insert:**
  - בהינתן פחות מ-N איברים במבנה הנתונים:
    מכניסה את הדף לראש רשימה מסוג 2 הכפופה לרשימה מסוג 1 בעלת הספרה 0 - (לדף חדש טרם יש פניות לזיכרון). אם לא קיימת רשימה עם הספרה 0 - יוצרים אחת כזו.
  - בהינתן N איברים במבנה הנתונים:
    ניגשים לרשימה הראשונה מסוג 1, מגיעים לאיבר האחרון שלה דרך המצביע ומוחקים אותו. לאחר מכן מבצעים את הכנסת הדף כמו בסעיף הקודם.
- **Reference:** מגיעים לדף ברשימה מקושרת מסוג 2 דרך המצביע שלו שקיבלנו כקלט, ומוחקים אותו מרשימה זו. עוברים לרשימה מסוג 1 העוקבת לרשימה שהכילה את הדף. אם היא גדולה ב-1 מרשימה זו, שומרים בראשה את הדף. אם לא - יוצרים שגדולה ממנה ב-1 ושומרים בה את הדף.

* במהלך Insert ו - Reference נעדכן בעת הצורך מצביעים בין חוליות בשני הרשימות המקושרות, וכן את מצביעי האיבר הראשון והאחרון.

פסאודו קוד

```
Insert(X)
1.  If numOfPages = N
    1.1.  Delete last element of the 1th linked list
    1.2.  Update Last pointer
2.  If first linked list value != 0
    2.1.  Create new linked list with value = 0
    2.2.  Update pointers
3.  Add X as the first element in the sub linked list of List 0
4.  Update First pointer

Reference(Y)
Let A be the upper linked list that contained Y
1.  Delete Y from A
2.  if A.Next.Value != A.Value +1
    2.1.  Create Linked list with value A.value + 1
    2.2.  Update links pointers
```

Let B be the upper linked list with A.value + 1
3. Insert Y at the top of B sub linked list
4. Update First pointer

<u>זמן ריצה</u>

- Init: O(1).
- Insert: In the worst case it deletes page, creates new link and update some pointers - all of which are O(1). So total of O(1).
- Reference: Same as Insert worst case - O(1).

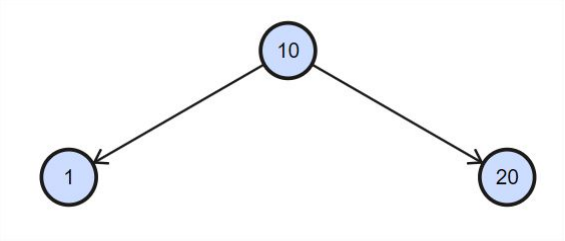
<u>שאלה 4</u>

<u>עצי BST</u>

א. נכון. ראו שאלה 2.

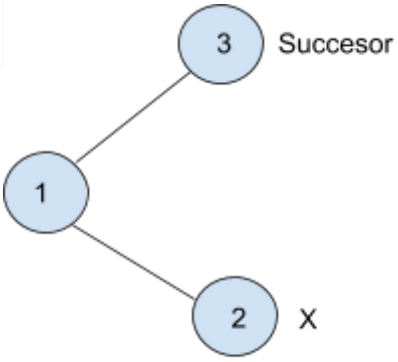
ב. לא נכון. בסריקת In Order השורש יודפס ראשון, לאחר מכן האיברים המושרשים בתת העץ השמאלי שלו ולאחר מכן האיברים המושרשים בתת העץ הימני שלו. 24 הוא המס' הראשון לכן הוא השורש. לאחר מכן יש את המס' 51 שגדול מ-24 - מכאן אנו מסיקים שאין לשורש תת עץ שמאלי, לכן על כל האיברים העוקבים ל51 להיות גדולים מ-24, וזאת בסתירה למס' 12 שמופיע מיד אחר כך.

ג. לא נכון - דוגמה נגדית:



PreOrder: 10, 1, 20
InOrder: 1, 20, 10

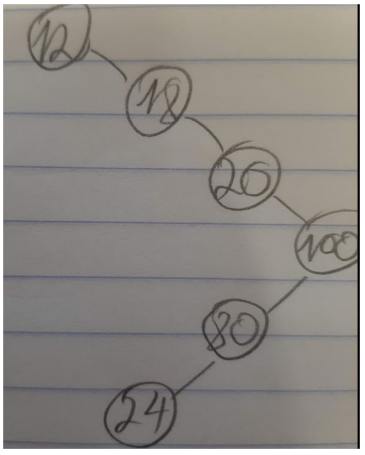
ד. לא נכון - דוגמה נגדית:



ה. נכון - אם X הוא בן שמאלי, מהגדרת עץ בינארי אבא שלו גדול ממנו ומכל תת העץ המושרש בו.

ו. לא נכון - בהינתן מקרה בו X בן שמאלי של צומת Z, ו-Y גדול גם מ-X וגם מ-Z, בהכרח Y לא ישורשר כבן של X - כי אז הוא יצור סתירה כאשר הוא יהיה גדול יותר מ-Z אך מושרש בתת העץ השמאלי שלו.

ז. נכון - הסריקה תתבצע כך במקרה להלן, והוא חוקי:

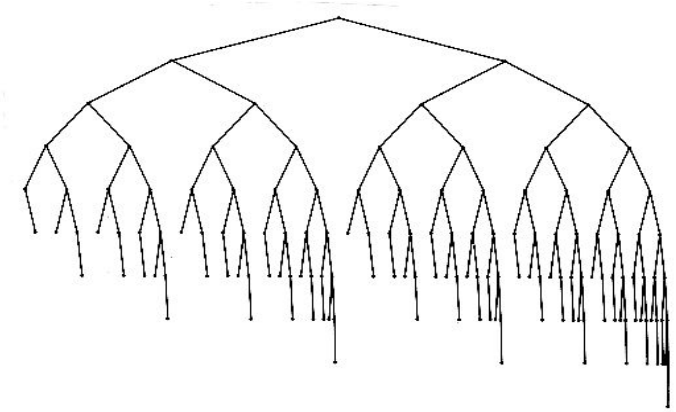


ח. לא נכון - 21 גדול מ-20 לכן 20 יכול להיות הבן שמאלי של 21 בלבד. אך אז תתקיים סתירה אם 25 הוא הבן של 20 - כי 25 גדול מ-21 אך מושרש בבנו השמאלי.

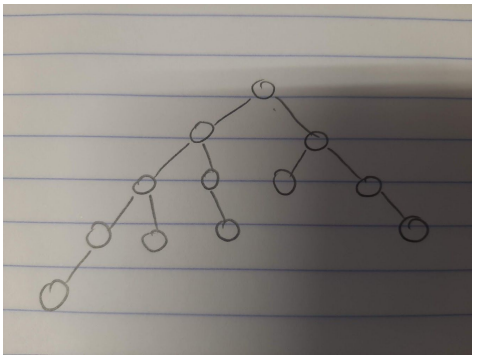
<u>עצי AVL</u>

א. לא נכון - נתבונן בעץ המפורסם הבא:

כבר ברמה h-4 ניתן לראות רמה לא מלאה (חסר הבן השמאלי ביותר).



ב. נכון - דוגמה לעץ בגובה 4 בעל מס' צמתים המינימלי, והוא אכן מכיל 12 צמתים:



ג. נכון

נוכיח באינדוקציה על h:

<u>בסיס:</u> h=1, כלומר יש רק עלה אחד (השורש). fib(1)=1 ולכן אי השוויון מתקיים.

<u>הנחת האינדוקציה:</u> נניח כי אי השוויון מתקיים לכל h < H ונוכיח עבור H.

<u>צעד האינדוקציה</u>: נתבונן בתת העץ הימני ובתת בעץ השמאלי של השורש.

נסמן את גובהם h1 ו-h2, ואת מס' העלים בהם n1 ו-n2 בהתאמה.

ברור כי h1,h2<H לכן הנחת האינדוקציה תקפה, כלומר:

n1 >= fib(n1) AND n2 >= fib(n2)

מס' העלים הכולל בעץ או איחוד העלים בשני עצים אלו, לכן מתקיים: n = n1 + n2

נציב בנוסחאות למעלה ונקבל:

n >= fib(h1) + fib(h2)

כעת, מהגדרת עץ AVL קיימות 2 אופציות:

1. h1 = h2 = H - 1
2. h1 + 1 = h2 = H-1 OR h2 + 1 = h1 = H-1

נציב את אופציה 1 בנוסחת פיבונאצ'י ונקבל:

fib(h1) + fib(h2) > fib(H)

נציב את אופציה 2 בנוסחת פיבונאצ'י ונקבל:

fib(h1) + fib(h2) = fib(H)

ובסה"כ:

fib(h1) + fib(h2) >= fib(H)

נציב את אי השוויון המודגש מעלה ונקבל:

n >= fib(H)

מ.ש.ל

<u>עצי B</u>

סעיף א':

על מנת ש-t יהיה מינימלי עלינו לדאוג שהעץ יהיה מלא ככל הניתן על מנת שנגיע לכמות האיברים בגובה הנתון. כלומר, נבנה עץ מקסימלי בגובה 5: לכל צומת מס' מפתחות מקסימלי 2t-1 ולו מס' ילדים מקסימלי 2t. נחשב:

2t-1+ 2t(2t-1) + 2t(2t(2t-1)) + 2t(2t(2t(2t-1))) + 2t(2t(2t(2t(2t-1)))) + 2t(2t(2t(2t(2t(2t-1)))))

כאשר כל מחובר מייצג את מס' המפתחות בכל רמה של העץ (סה"כ 5 רמות).

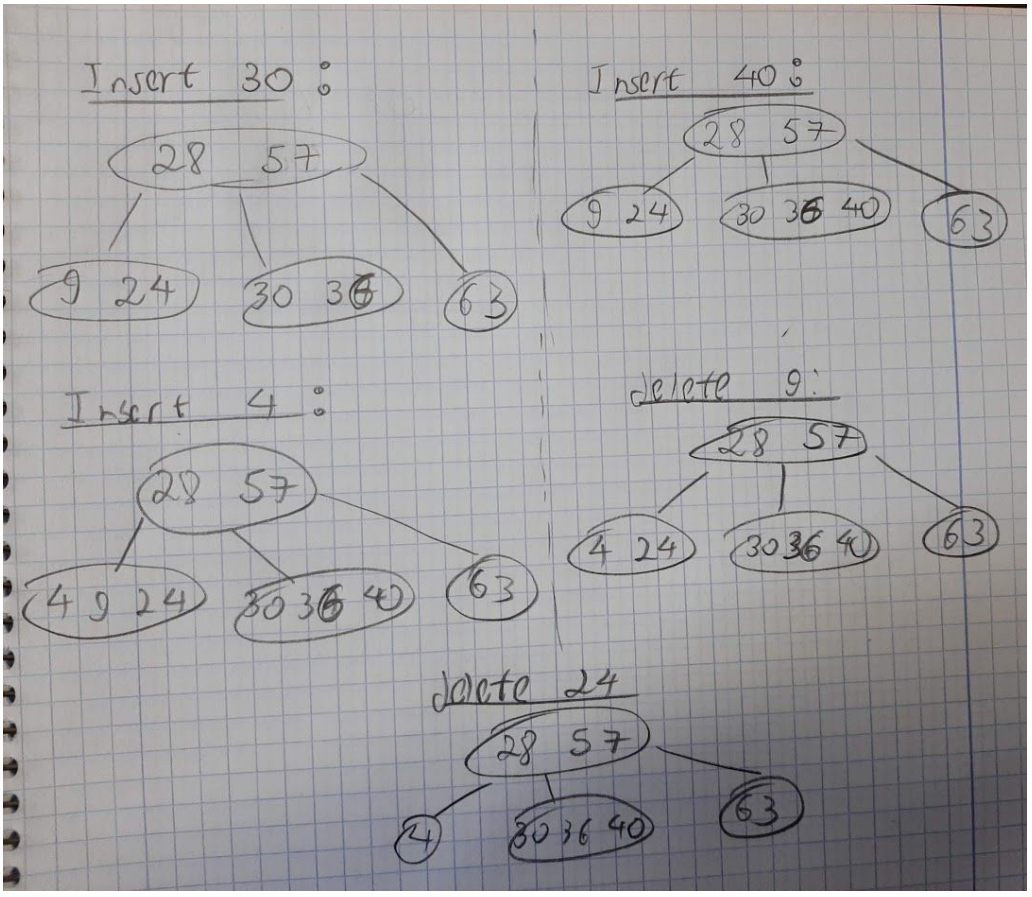
נשתמש בקירוב 2t-1 = 2t כדי לפשט את החישוב:

$$2t + 2^2 t^2 + 2^3 t^3 + 2^4 t^4 + 2^5 t^5 + 2^6 t^6$$

נציב t=9 ונקבל כ-36 מיליון, כלומר מעט מידי

נציב t=10 ונקבלו כ-67 מיליון, כלומר מעל 44 מיליון וזוהי התשובה.

סעיף ב':



סעיף ג':

- נוסיף שדה Size לכל צומת בעץ - אשר מכיל את מספר הצמתים הכולל בעץ המושרש בצומת זה.
- בהינתן צומת X - נחפש אותו בעץ.
- נתחזק משתנה lessThanX שיסכום את כל הצמתים בעלי מפתחות קטנים משל X.
- אם יש ל-X בן שמאלי: X גדול מכל תת העץ המושרש בו, לכן נוסיף ל-lessThanX את ה-Size של בנו השמאלי.
- אם X הוא בן ימני: אזי אבא של X וכל תת העץ השמאלי שלו קטנים גם הם מ-X. לכן נוסיף ל-lessThanX את ה-Size של בנו השמאלי של האב + 1 (האב עצמו).
- אם אבא P של X הוא גם בן ימני, באופן דומה יש להוסיף ל-lessThanX את אבא של P ואת ה-Size של בנו השמאלי. כל עוד יש לאבא בן ימני - נמשיך להגדיל בצורה דומה את lessThanX.

<u>ניתוח זמן ריצה:</u>

- חיפוש X בעץ: O(logn
- עליה למעלה כל עוד לאבא יש בן ימני: O(logn
- סה"כ:
- O(logn) + O(logn) = O(logn

שאלה 5

פירוט האלגוריתם

- INIT: נאתחל עץ AVL ריק ושלושה משתנים גלובלים מאותחלים לאפס:
  - min, max: ישמרו את ערך האיבר המקסימלי והמינימלי בעץ.
  - balance: ייצג את סכום ההעלאות וההורדות שבוצעו לערכים בעץ.
- Insert:
  - נכניס את האיבר X-balance לעץ - כלומר העץ שלנו לא יאחסן את הערכים המקוריים שאנו נדרשים להכניס, אלא את ההפרש בינם לבין ה-balance בעת ההכנסה.
  - במקרה ש X - balance גדול מ-max או קטן מ-min נעדכן את min ו-max בהתאם.
- Delete:
  - נמחק את האיבר x-balance מהעץ.
  - נבדוק אם ערך זה שווה ל min או max. במידה שכן - נעדכן את min ב-max בהתאם לעצי AVL תוך התחשבות ב-balance:
    - min = ערך הקודקוד השמאלי התחתון פחות balance
    - max = ערך הקודקוד הימני התחתון פחות balance
- Search: נחפש בעץ את הערך x-balance.
- Upgrade: נעדכן balance = |min| + balance
- Downgrade: נעדכן balance = |max| + balance

פסאודו קוד:

```
Init()
        initialize AVL t;
        max=0, min=0, balance =0;

Insert(x)
        t.insert(x-balance );
        if(min>x-balance )
                min=x-balance ;
        if(max<x-balance )
                max=x-balance ;

Delete(x)
        t.delete(x-balance );
        if(min==x-balance )
                min=t.Min - balance;
        if(max==x-balance)
                max=t.Max() - balance;
```

```
Search(x)
        t.Seacrh(x-balance);

Upgrade()
        balance = balance+ |min + balance|;

Downgrade()
        balance = balance- |max + balance|;
```

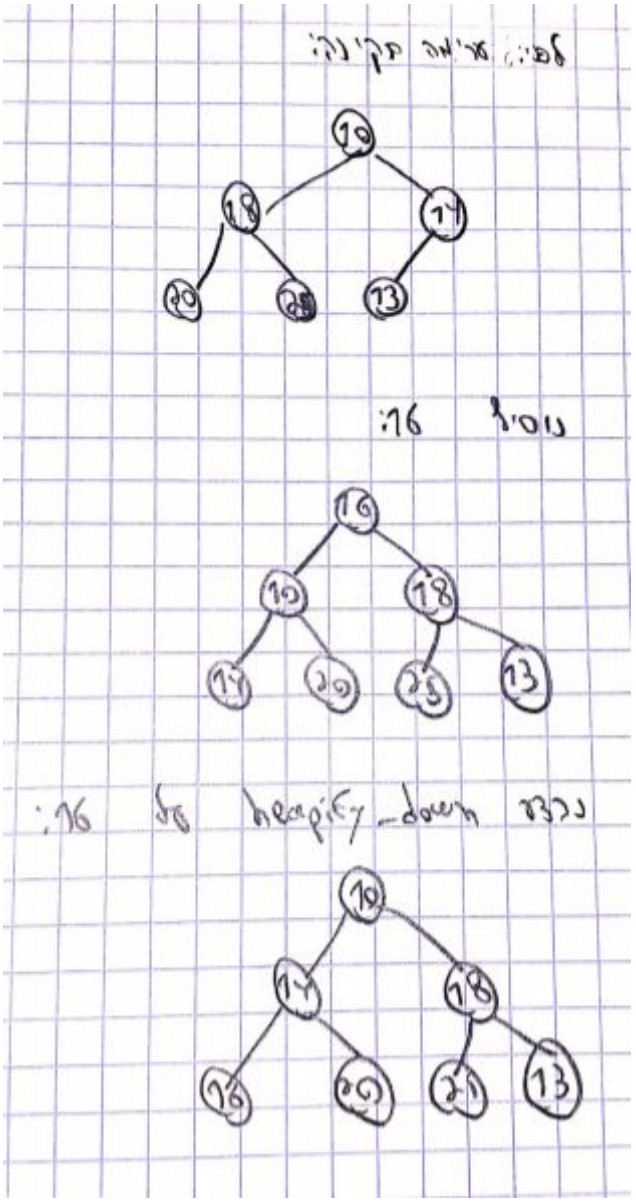
<u>ניתוח זמן ריצה</u>

- Init: פעולות קבועות בזמן O(1)
- Insert: הכנסה בעץ AVL עולה O(logn) ועדכון המשתנים O(1)
- Delete: מחיקה בעץ AVL עולה O(logn), ומציאת min ו - max תלויה בגובה העץ לכן גם עולה O(logn)
- Search: חיפוש בעץ AVL עולה O(logn)
- Ugrade: עדכון שדה ב-O(1)
- Downgrade: עדכון שדה ב-O(1)

(Total: O(1) + O(logn) + O(logn) + O(logn) + O(logn) = O(logn

שאלה 6

הפתרון של עומר לא עובד. נציג דוגמא נגדית:



אולם, כעת 18 נמצא מעל 13 ולכן לא התקבלה ערימה תקינה.

<u>שאלה 7</u>

נפתור את השאלה באמצעות שני עצי AVL, הראשון T1 יאחסן את האיברים בקבוצה S, והשני T2 יאחסן את זוגות האיברים המקיימים a = log(b).

Init: אתחול שני עצי AVL ריקים.

Insert:

- נכניס את X לעץ T1 כרגיל.
- נחשב את 2^x - אם קיבלנו מס' שלם, נחפש את התוצאה ב-T1
  אם מצאנו - משמע שקיים הזוג 2^x ו-X, לכן נכניס אותם כצומת אחד לעץ הזוגות T2.
- נחשב את logx - אם קיבלנו מס' שלם, נחפש את התוצאה ב-T1. באופן דומה נכניס את הזוג אם קיים לעץ T2.

Delete:

- נחפש את X ב-T1 ונמחק אותו משם.
- נחפש את X ואת logx ב-T2. אם מצאנו את הצומת נמחק אותה.

Search: נשתמש בחיפוש סטנדרטי של עץ AVL בעץ T1.

log_Pair: מדפיסים את הזוג שבשורש בעץ T2.

<u>זמני ריצה</u>

Insert: Inserting to T1: O(logn), Searching in T1 O(logn), Inserting to T2: O(logn).
O(logn) + O(logn) + O(logn) = O(logn)

Delete: Searching in T1: O(logn), Searching in T2: O(logn)
total of O(logn) + O(logn) = O(logn)

Search: O(logn)

log_pair: accessing T2 root: O(1)

שאלה 8

סעיף א':

אלגוריתם בזמן גרוע

- נעבור על כל איברי הקבוצה לפי הסדר ונכניס אותם לעץ AVL.
- כל צומת בעץ יכיל בשני שדות נוספים שיאותחלו לאפס:
  - minIndex: ישמור את האינדקס הראשון של הספרה בקבוצה.
  - diff: ישמור בתום ריצת האלגוריתם את ההפרש המקסימלי בין 2 אינדקסים של המספר.
- בעת הכנסת כל איבר מהקבוצה לעץ:
  - במקרה שהוא לא מופיע בעץ נעדכן את minIndex לאינדקס הנוכחי בקבוצה.
  - במקרה שהוא כבר מופיע בעץ, נעדכן את diff להיות האינדקס הנוכחי פחות minIndex.
  - נתחזק משתנה גלובלי maxDiff שיהיה מאותחל לאפס ונעדכן אותו ל-diff בהנחה ש-diff גדול ממנו.
- למציאת ההפרש המקסימלי נחזיר את maxDiff

ניתוח זמן ריצה

- מעבר על כל איברי הקבוצה O(n), הכנסה של כל איבר לעץ O(logn), סה"כ: O(nlogn)
- עדכון שדות O(1)
- מעבר על כל איברי העץ למציאת האיבר עם ההפרש המקסימלי: O(n)

Total: O(nlogn) + O(1) + O(n) = O(nlogn)

אלגוריתם בזמן צפוי

- נשתמש בטבלת Hash Chaining שתאחסן את כל איברי הקבוצה.
- לכל איבר שנכניס לטבלה יהיו 2 שדות נוספים מאותחלים לאפס:
  - minIndex: ישמור את האינדקס הראשון של הספרה בקבוצה.
  - diff: ישמור בתום ריצת האלגוריתם את ההפרש המקסימלי בין 2 אינדקסים של המספר.
- נתחזק משתנה גלובלי נוסף maxValue = 0.
- בעת הכנסת איבר לטבלה, במידה שהרשימה המקושרת לתא ריקה, או שאינה מכילה את המספר שלנו, נכניס את האיבר כרגיל ונשמור את האינדקס שלו בקבוצה ב-minIndex.
- במידה שהרשימה המקושרת כבר מכילה את המספר:
  - נעדכן את diff להיות האינדקס הנוכחי פחות minIndex.
  - אם diff>maxValue אזי נעדכן: maxValue=diff.
- למציאת ההפרש המקסימלי נחזיר את maxValue.

ניתוח זמן ריצה

- הכנסת איבר לטבלת האש: O(1) צפוי. הכנסת n איברי הקבוצה: O(n) צפוי.
- בהנחה שקיים פיזור אחיד, עולה לנו O(1) לוודא האם האיבר שאנו מכניסים מופיע ברשימה המקושרת של התא הרלוונטי.
- עדכון שדות: O(1) צפוי.

Total: O(n) Expected.

סעיף ב'

אלגוריתם בזמן גרוע

- נמיין את איברי הסדרה באמצעות Merge Sort
- נתחזק 4 משתנים גלובלים מאותחלים לאפס:
  - CurrSeqIndex: אינדקס ההתחלה של הסדרה הנוכחית
  - CurSeqLen: אורך הסדרה הנוכחית
  - MaxSeqLen: אורך הסדרה המקסימלית
  - MaxSeqIndex: אינדקס ההתחלה של הסדרה המקסימלית
- נעבור איבר-איבר על המערך הממוין, ונבדוק האם כל איבר גדול מקודמו. אם כן, נעלה את CurSeqLen באחד. במקרה שהגענו לאיבר שאינו גדול באחד בקודמו:
  - אם CurrSeqLen גדול מ-MaxSeqLen נעדכן:
    MaxSeqLen = CurrSeqLen וגם MaxSeqIndex = CurrSeqIndex
  - נאפס את CurrSeqLen ונעדכן את CurrSeqIndex להיות האינדקס הבא.
- נמשיך לסרוק איבר איבר במערך.

זמן ריצה

- מיון מיזוג: O(nlogn)
- מעבר על המערך איבר איבר ועדכון שדות: O(n)

Total: O(nlogn) + O(n) = O(nlogn)

אלגוריתם בזמן צפוי

- נאחסן את כל איברי הקבוצה בטבלת Hash with Chaining.
- נעבור על כל איברי הקבוצה ונבדוק האם כל איבר מתחיל סדרה חדשה. נעשה זאת באמצעות בדיקה האם קיים איבר קטן ממנו באחד בטבלת ההאש.
  - אם האיבר אינו מתחיל סדרה חדשה, נמשיך לסרוק את האיבר הבא בקבוצה.
  - אם האיבר מתחיל סדרה חדשה, נמצא את גודלה ע"י חיפוש בטבלת ההאש באופן איטרטיבי מספרים הגדלים ממנו באחד, עד אשר איננו מוצאים מספר כזה. נסכום את כמות האיברים בסדרה הנ"ל.
- נתחזק עוד שני משתנים MaxSeqLen, MaxSeqIndex בדומה לאלגוריתם הקודם ונעדכן אותם במידה וגילינו סדרה באורך גדול יותר.
- לבסוף ניגש לאיבר באינדקס MaxSeqIndex בקבוצה ונדפיס את MaxSeqLen האיברים העוקבים לו.

זמן ריצה

- הכנסת כל האיברים לטבלת האש: O(n) הצפוי.
- מעבר איבר איבר ובדיקה אם הוא מתחיל סדרה: O(n) צפוי
- בדיקת האיברים העוקבים למספר שמתחיל סדרה: בסה"כ יש לנו n איברים בקבוצה, לכן מספר איברים זה לא יעלה על n. לכן O(n) צפוי.

Total: O(n) + O(n) + O(n) = O(n) Expected.

סעיף ג'

אלגוריתם בזמן גרוע

- נמיין את המערך באמצעות מיון מיזוג
- ניצור מערך דו מימדי בעל n שורות ו2 עמודות, שעמודה אחת שלו תייצג את המספר בקבוצה ועמודה שניה תייצג את השכיחות שלו.
- נעבור על איברי המערך הממוין, נספור את מס' המופעים של כל מספר ונוסיף למערך הדו מימדי שורה של המספר לצד השכיחות שלו.
- נמיין את המערך הדו מימדי על פי עמודת השכיחות שלו
- נדפיס כל מספר כמספר המופעים שלו על פי סדר השורות במערך הדו מימדי.

זמן ריצה

- מיון מיזוג O(nlogn)
- אתחול מערך דו מימדי O(2n)
- מעבר על איברי המערך הממוין O(n)
- מיון המערך הדו מימדי O(nlogn)
- הדפסה O(n)

Total: O(nlogn) + O(2n) + O(n) + O(nlogn) = O(nlogn)

<u>שאלה 9</u>

<u>פירוט האלגוריתם</u>

- נשמור שתי רשימות דו כיווניות:
  - Sorted: תכיל את כל האיברים בסדר ממוין.
  - Special: תכיל את כל האיברים המיוחדים.
- בשתי הרשימות יהיו מצביעים לאיבר הראשון ולאיבר האחרון
- **Add:**
  - נסרוק את האיברים ברשימה Sorted ונכניס את k במקומו המתאים בסדר ממוין.
  - אם k מיוחד נכניס אותו ל-Special ונוסיף מצביע הדדי של האיבר בין 2 הרשימות.
- **removeSmallest:** ניגש לאיבר הראשון ברשימה Sorted ונוציא אותו. נבדוק באמצעות המבציע ההדדי אם הוא קיים ברשימה Special, ונוציא אותו גם משם במידה שכן.
- **remove Largest:** ניגש באמצעות מצביע לאיבר האחרון ברשימה Sorted ונוציא אותו. נבדוק אם קיים ב-Special באמצעות המצביע ההדדי ונוציא אותו משם.
- **oldest**: ניגש לאיבר הראשון ברשימה special ונחזיר אותו.

<u>פסאודו קוד:</u>

```
add(k)
        Sorted.addSorted(k);
        if(Sorted.lastKey==k)
                last=Sorted.lastKey;
        if(k.isSpecial())
                Special.addAtLast(k);
        Add dual pointers
removeSmallest()
        first=Sorted.first;
        Sorted.removeFirst();
        if(first.isSpecial())
                Special.delete(first);
        if(Sorted.first==null)
                first=null;
removeLargest()
        Sorted.delete(last);
        if(last.isSpecial())
                Special.delete(last);
oldest()
        return Special.first;
```

<u>ניתוח זמן ריצה</u>

- Add: מתבצע מעבר על שתי הרשימות, לכן O(n) + O(n)
- removeSmallest/removeLargest: גישה לאיברים ומחיקתם מתבצעים ב-O(1)
- oldest: גישה לאיבר O(1)

Total: O(n) + O(n) + O(1) = O(n)

שאלה 10

פירוט האלגוריתם

- נשתמש במחסנית, ונכניס אליה תחילה איבר המייצג את את ציר y.
- לאחר מכן נעבור בלולאה על כל הקטעים האנכיים ונבדוק האם הקטע שנמצא בראש המחסנית גבוה ממנו.
- במידה שכן, נעדכן את את קטע זה להיות הקטע שמשמאלו של הקטע הנוכחי. ולאחר מכן נכניס את הקטע הנוכחי למחסנית.
- במידה ולא - נשלוף קטעים מהמחסנית עד שנקבל קטע גבוה יותר.

פסאודו קוד:

```
getView()
        Stack s=new Stack();
        s.push(y);
        for i from
                lastSeg=s.Peek()
                while (Si.height > lastSeg.height)
                        s.Pop();
                        lastSeg=s.Peek()
                Si.left=lastSeg;
                s.push(Si);
```

ניתוח זמן ריצה

כל אחד מ-n הקטעים נכנס ויוצא מהמחסנית פעם אחת בלבד.

במסגרת כל פעולת הכנסה או הוצאה מהמחסנית מתבצעות פעולות קבועות ב-O(1).

ישנם n קטעים, לכן בסך הכל נקבל O(n).