**יבש 2**

מגישים: נמרוד קדיש ואלכס בלגודרסקי

**שאלה 1**

1. נשתמש ברשימת דילוגים דטרמיניסטית. האיברים ברשימה יהיו מס' הקומות השונים. בנוסף נחזיק מצביע Curr(בשורה התחתונה ביותר של האיברים) עבור הקומה הנוכחית.

מימוש הפעולות:

* Init():

ניצור ונאתחל את רשימת הדילוגים (עם איברים +∞ ו-∞ בקצוות כפי שנלמד בהרצאה) ואיבר בודד 0 שמציין את הקומה ההתחלתית O(1). בנוסף נעדכן את המצביע של הקומה הנוכחית להצביע על איבר ה-0 O(1). סה"כ סיבוכיות זמן O(1) כנדרש.

* AddStop(k):

נבצע insert(k) ברשימה ונכניס את האיבר הk במקומו ברשימת הדילוגים. O(log(n)) כנדרש.

* NextStop():

ניגש לאיבר השמור לנו במצביע Curr, ונבדוק האם איבר הnext שלו מאותחל. במידה ולא, לא קיימת קומה נוספת, ונדפיס את הקומה הנוכחית ונסיים. אחרת, נקדם את המצביע Curr להצביע על האיבר הnext ונדפיס את ערכו. מס' קבוע של פעולות O(1) ולכן סה"כ O(1) כנדרש.

1. **ניתוח והוכחת סיבוכיות משוערכת**

**שאלה 2**

1. נשתמש בעץ דרגות AVL הממוין מיון ראשוני לפי ערך ה-x של הנק' ומיון משני לפי ערך ה-y של הנק'. בנוסף נחזיק 2 משתנים Ksmallest, Kbiggest שיחזיקו מצביע לאיבר ה-k הקטן ביותר וה-k הגדול ביותר בהתאמה. נחזיק תא זיכרון נוסף לשמירת ערך ה-k.

מימוש הפעולות:

* Init(k):

ניצור עץ AVL ריק O(1) ונשמור את ערך ה-k שקיבלנו בתא הזיכרון המתאים O(1). בנוסף, נאתחל את המצביעים Ksmallest, Kbiggest. סה"כ O(1) כנדרש.

* Insert((x,y)):

נכניס את האיבר (x,y) לעץ הAVLO(log(n)). ונשתמש בפעולת select באופן הבא:

- Ksmallest=select(k+1)– זהו הקטע שיש k איברים קטנים ממנו בעץ.

- Kbiggest=select(n-k)– זהו הקטע שיש k איברים גדולים ממנו בעץ.

2 פעולות הselect לוקחות O(2log(n)), ולכן בסה"כ סיבוכיות הזמן הינה O(log(n)) כנדרש.

* Delete((x,y)):

נסיר את האיבר (x,y)מעץ ה-AVL (O(logn(n))). נעדכן את המצביעים בעזרת פעולת select באופן דומה:

- Ksmallest=select(k+1)– זהו הקטע שיש k איברים קטנים ממנו בעץ.

- Kbiggest=select(n-k)– זהו הקטע שיש k איברים גדולים ממנו בעץ.

2 פעולות הselect לוקחות O(2log(n)), ולכן בסה"כ סיבוכיות הזמן הינה O(log(n)) כנדרש.

* IsCentric((x,y)):

נבדוק האם ערך ה-x שהתקבל גדול או שווה לערך ה-x של Ksmallest וגם נבדוק האם ערך ה-y שהתקבל קטן או שווה לערך ה-y של Kbiggest– אם שני התנאים מתקיימים נחזיר True, אחרת נחזיר False. סה"כ מס' קבוע של פעולות O(1) ולכן סיבוכיות הזמן הינה O(1) כנדרש.

1. להשלים

**שאלה 3**

1. תחילה "נמקם" את העץ 1T משמאל ל2T כך שעומק העלים זהה בשני העצים (הגובה לאו דווקא שווה).

מהנתון כי כל מפתח בT1 קטן מכל מפתח בT2 נקבל כי שמורת עץ 2-3 נשמרת, ובנוסף כל אחד מן העצים הללו הוא בעצמו עץ 2-3 ולכן השמורה הפנימית בהם נשמרת גם כן. כעת נרצה לאחד את העצים לעץ 2-3 יחיד. נפעל לפי המקרים הבאים(באופן רקורסיבי כלפי מעלה לכיוון השורש של העצים – נתחיל מהרמה מעל העלים בעץ):

- אם העץ השמאלי (1T) הוא בעל 2 בנים נצרף לאביו כבן שלישי ימני את הבן השמאלי ביותר באותו העומק בעץ השני כך שיחזיק כעת 3 בנים. נמשיך כלפי מעלה באופן רקורסיבי.

-אחרת, נמשיך כלפי מעלה.

- אם העצים היו בעלי אותו הגובה ניצור שורש חדש ונחבר לו את שני השורשים של T1 ו T2 ונסיים.

אחרת, אחד העצים גבוה יותר מהשני, ולכן כאשר נגיע ברקורסיה לשורש העץ הנמוך יותר, נחבר את השורש שלו לאב של ההאיברים האחרים בעץ השני(במידה וכבר יש לו 3 בנים, ניצור שורש חדש נוסף ונחבר אליו את השורש של העץ הנמוך יותר בתוספת אחד הבנים הצמודים אליו באותה הרמה של העץ האחר ונמשיך באופן זה עד שנגיע לגובה העץ המקס' מבינהם).

את גובהיי העץ ניתן לגלות בעזרת פונ' רקורסיבית שלוקחת O(log(h)) כאשר h הוא גובה העץ ועבור עצי 2-3 מתקיים כי O(log(h))=O(log(n)) ולכן עבור הפעלת הפונצ' על שני העצים נעמוד בסיבוכיות הנדרשת של O(max{log(n1),log(n2)}) .

בסיור שתיארנו מהרמה התחתונה ועד לשורש העץ הגבוהה יקח לנו גם O(max{log(n1),log(n2)}) (לפי גובה העץ הגבוהה מבינהם) ובכל רמה אנו מבצעים מס' קבוע של פעולות תיקון O(1). ולכן בסה"כ סיבוכיות הזמן של האלגוריתם המוצע הינה O(max{log(n1),log(n2)}) כנדרש.

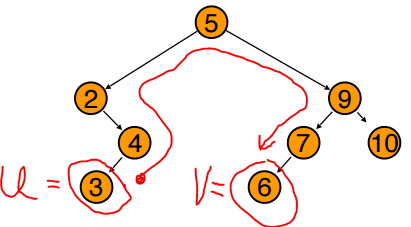
1. להשלים
2. להשלים
3. להשלים

**שאלה 4**

להשלים

**שאלה 5**

1. לא נכון, נראה טענה חזקה יותר: לא קיים עץ בינארי עם לפחות שני צמתים ולו סיורי pre-order ו-post-order זהים. בכל סיור post-order השורש של העץ יהיה הראשון בסיור, בעוד שבאף סיור pre-order זה לא מתקיים.
2. לא נכון, נפריך בעזרת דוגמה נגדית:



עבור u=3 וv=6 מתקיים כי u<v אך הצמתים במסלול בינהם אינם ממויינים לפי מפתחות בסדר עולה (**4,2**,5,9,7)

1. נכון. יהי v צומת פנימי כלשהו בעץ 2-3.

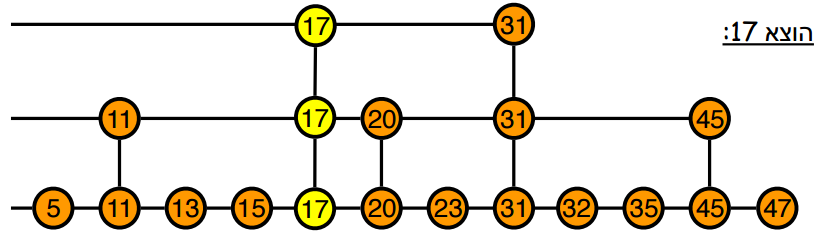
-עבור צומת בעל בן שמאלי בלבד (כלומר M=R=0,L≠0) מתקיים L=ϴ(L) באופן ברור.

-עבור צומת בעל בן שמאלי ואמצעי בלבד(כלומר R=0,L≠0,M≠0)מתקיים כי שני תתי העצים TL וTMבאותו הגובה(בנים של אותו צומת אב v ) נסמן גובה זה בh ולכן מתקיים: Log2(h)≤M≤Log3(h) ובאופן דומה Log2(h)≤L≤Log3(h) ולכן אכן מתקיים כי L= ϴ(L+M).

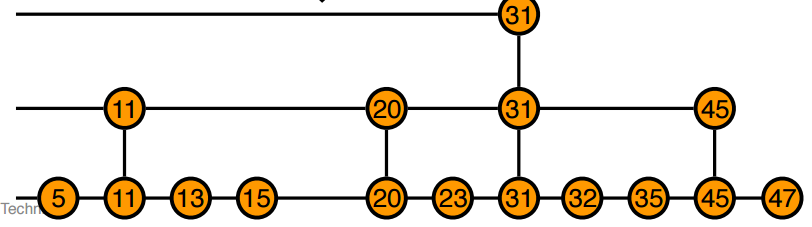
-עבור צומת בעל 3 בנים (כלומר R≠0,L≠0,M≠0) מתקיים כי שלושת תתי העצים TR,TL וTMבאותו הגובה(בנים של אותו צומת אב v ) נסמן גובה זה בh ולכן מתקיים: Log2(h)≤M≤Log3(h) ובאופן דומה Log2(h)≤L≤Log3(h)ובנוסף גם Log2(h)≤R≤Log3(h)ולכן אכן מתקיים כי L= ϴ(L+M+R).

קיבלנו כי לכל המקרים האפשריים של צומת v פנימי בעץ 2-3 מתקיים L= ϴ(L+M+R) ולכן הטענה נכונה.

1. לא נכון, נפריך בעזרת דוגמא נגדית:



עבור delete(17), ברשימה הנ"ל, נקבל את הרשימה הבאה:



כעת עבור הפעולה insert(17) לא בהכרח נקבל את אותה הרשימה מכיוון שברשימת דילוגים אי -דטרמיניסטית מס' ההופעות של איבר שנוסף בעזרת insert תלוי בהטלות המטבע שנבצע, ולכן במקרה ולא נקבל 3 הטלות מטבע בדיוק של "1" ואז הטלה של "0" ,הרי שלא יהיו 3 מופעים לאיבר כמו מקודם ומבנה S לא יישמר.

1. הטענה נכונה.

פעולת insert(x) תכניס לנו איבר חדש למבנה S ומס' המופעים שלו ברמות השונות ייקבע לפי הטלות המטבע כפי שציינו בסעיף הקודם. כעת, בניגוד לסדר הפעולות בסעיף הקודם, פעולת delete(x) מוחקת את כל המופעים של האיבר x במבנה S ללא תלות במס' מופעיו וההטלות שהתקבלו בעת ביצוע פעולת הinsert. כלומר לפני הפעולה insert(x) היה לנו מבנה S ללא האיבר x בתוכו, ומיד לאחר סיום פעולת delete(x) נקבל את אותו המבנה S בדיוק – ללא האיבר S וללא שינוי האיברים האחרים.