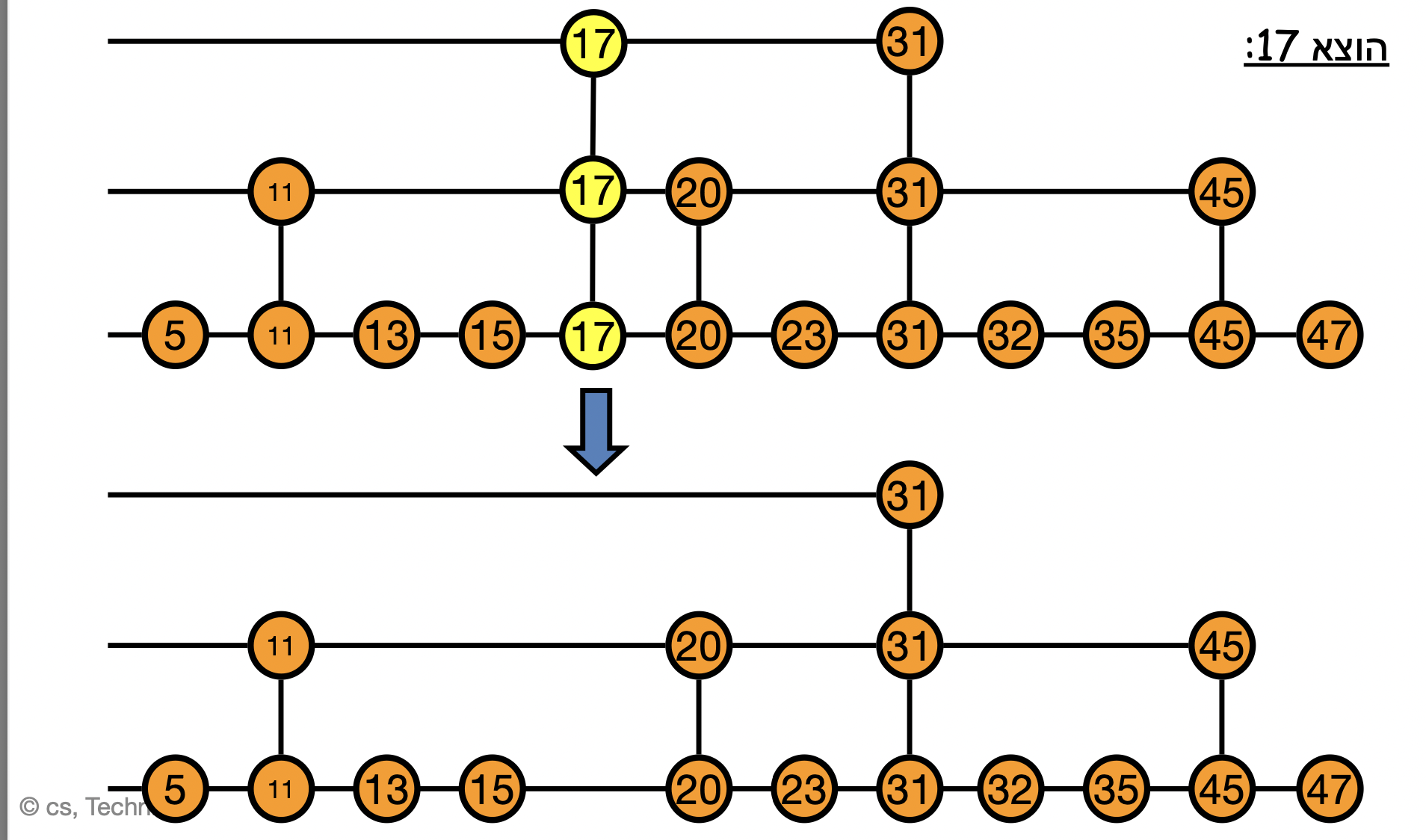
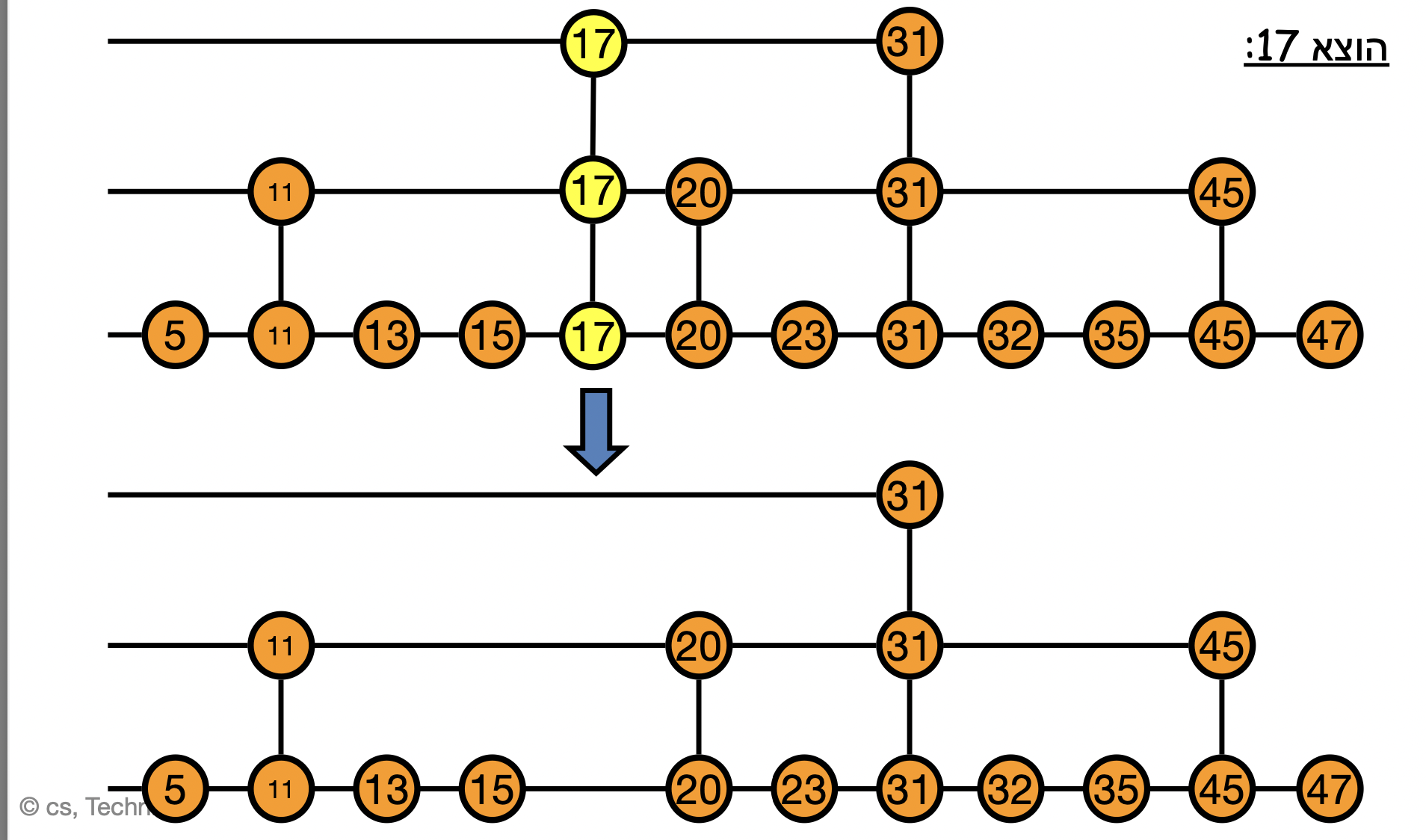
גיליון יבש 2

עדי צח ודניאל יצחק

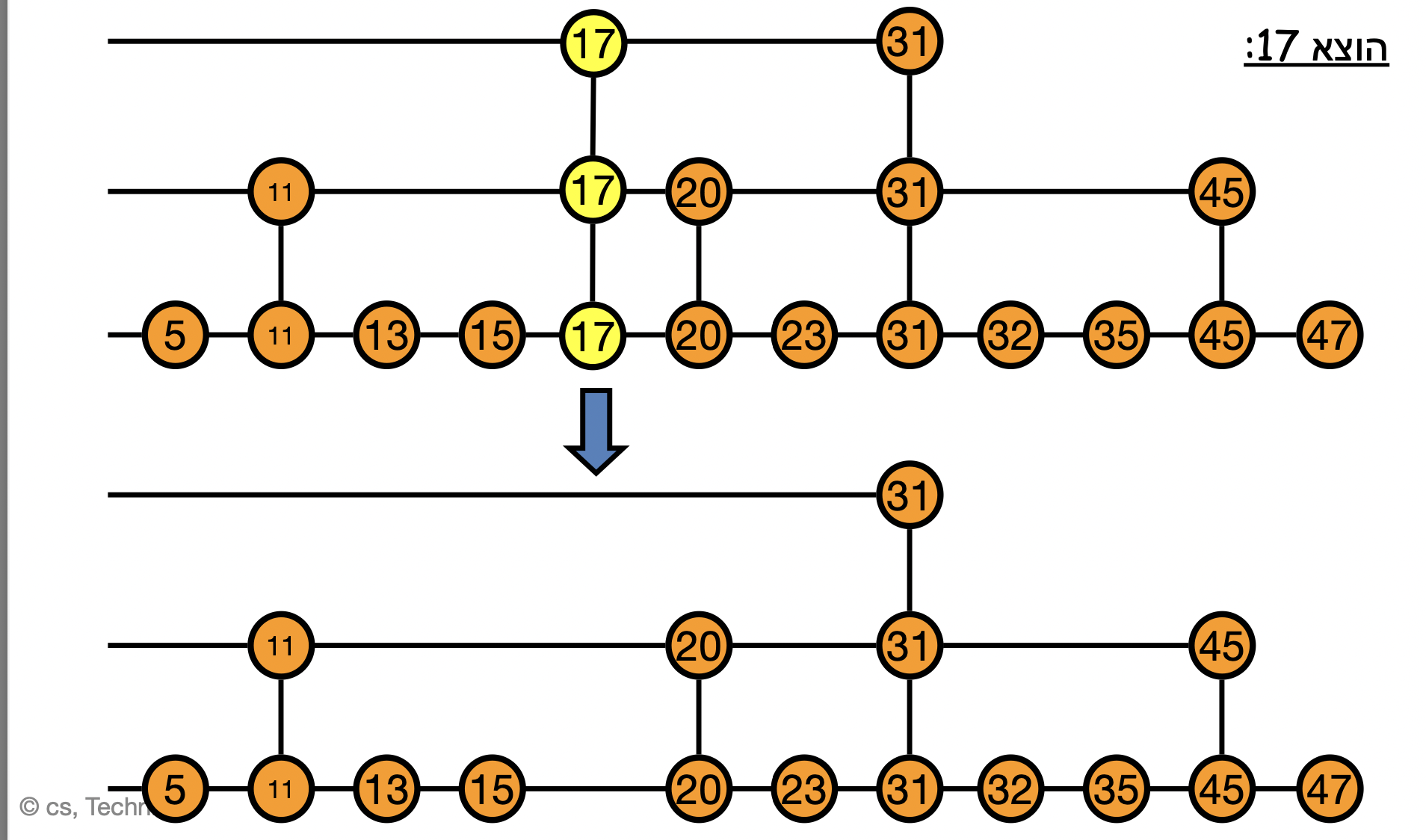
# שאלה 1

1. לכל רשימת דילוגים רנדומלית ולכל , עבור זוג הפעולות מתקיים שלכל פעולת ואחריה אינן משנות את .

הטענה אינה נכונה. נפריך טענה זו בעזרת דוגמא נגדית אשר כוללת הוצאה של הספרה 17 ולאחר מכן הכנסה מחודשת.









הרשימה המקורית אשר התקבלה הכילה את הספרה 17 בשלוש רמות שונות. ע״י הוצאה של המספר הנתון תתקבל רשימת דילוגים אשר כלל לא מכילה את המספר, ובפרט לא מכילה אותו בשלושת הרמות אשר בהן היא הכילה אותו לפני כן.

כעת, הכנסה מחודשת של הספרה 17 תמקם צומת 17 ברמה התחתונה ביותר, באותו המיקום היחסי בו היה הצומת התחתון ביותר ברשימת דילוגים הראשונה, אבל הפעם, לאחר ההכנסה הראשונית של הצומת התחתון ״יוצא לנו 1״ ועל כן נפסיק את תהליך הוספת הצמתים.

סה״כ נקבל רשימת דילוגים אשר נבדלת מהרשימה המקורית, זאת מכיוון שמספר צמתי 17 שונה מ-3 ל-1.

1. לכל רשימת דילוגים רנדומלית ולכל , עבור זוג הפעולות מתקיים שלכל פעולותואחריה אינן משנות את .

הטענה נכונה.

נניח בשלילה כי הטענה שגויה, כלומר ישנה רשימת דילוגים S רנדומלית כלשהי כך שישנו עבורו פעולת ואחריה פעולת יקיימו שרשימת הדילוגים המתקבלת הינה שונה מרשימת הדילוגים המקורית S.

מכיוון שרשימת הדילוגים המתקבלת הינה שונה מרשימת הדילוגים המקורית מתקיים כי ישנו לפחות צומת אחד שונה בין הרשימות או מצביע אחד שונה.

מכיוון שהפעולות היחידות אשר התבצעו הינן עבור האיבר x אזי לא יכול להתקיים כי הצומת השונה בין הרשימות הינו צומת שאינו x, בהתאם למה שלמדנו בהרצאה, אופי הפונקציות הנ״ל משפיע אך ורק על קיום או חוסר הקיום של איבר x. כמו כן, המצביעים היחידים אשר מושפעים משתי הפעולות הנ״ל הינם המצביעים שכנים אשר היו ״באינטרקציה״ (שונו להצביע על x ואז הפסיקו להצביע על x) עם הצומת x בשל אותן סיבות.

אבל, מאופן הגדרת הפונקציות נקבל כי תחילה תמקם את x במיקום הרלוונטי לו ברשימת הדילוגים ולאחר מכן תוסיף מעליו בהתאם לתוצאות ״הטלת המטבע״ את כמות הצמתים הנוספים הנחוצה (בעלי המפתח של איבר x). ללא תלות בכמות הצמתים עם המפתח של x נקבל כי פעולת אחר כך תסיר את כלל הצמתים של איבר x ותשנה את המצביעים השכנים לו באופן כזה שיצביעו חזרה על הצמתים שהיו שכנים לפני ביצוע .

כלומר הראנו כי הן כלל הצמתים יישארו כפי שהם והן כי כלל המצביעים ברשימת דילוגים יישארו כפי שהם, לפני ואחרי ביצוע שתי פעולות הנ״ל. לכן הגענו לסתירה להנחה בשלילה ומתקיים בהכרח כי לכל רשימת דילוגים רנדומלית ולכל , עבור זוג הפעולות מתקיים שלכל פעולותואחריה אינן משנות את .



1. הפרכה:

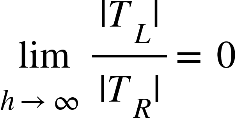
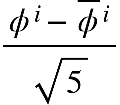


רעיון ההפרכה: נראה דוג' נגדית כך שתת העץ הימני (נסמנו TR ) יהיה עץ AVL מקסימלי (לפי ההרצאה, עץ בינארי מלא) בעוד תת העץ השמאלי (נסמנו TL ) יהיה עץ AVL מינימלי (לפי ההרצאה, עץ פיבונאצ'י). נציין כי T גם בדוגמה זאת שומר על היותו עץ AVL וזאת מכיוון שכל תת עץ שלו אכן עץ AVL ואיזון הגובה שלו נשמר. ונראה כי מתקיים ולכן בפרט לא מתקיים .

יהא עץ , בעל גובה h כך שTR עץ בינארי מלא בעל גובה h-1 ולכן, ע"פ ההרצאה :

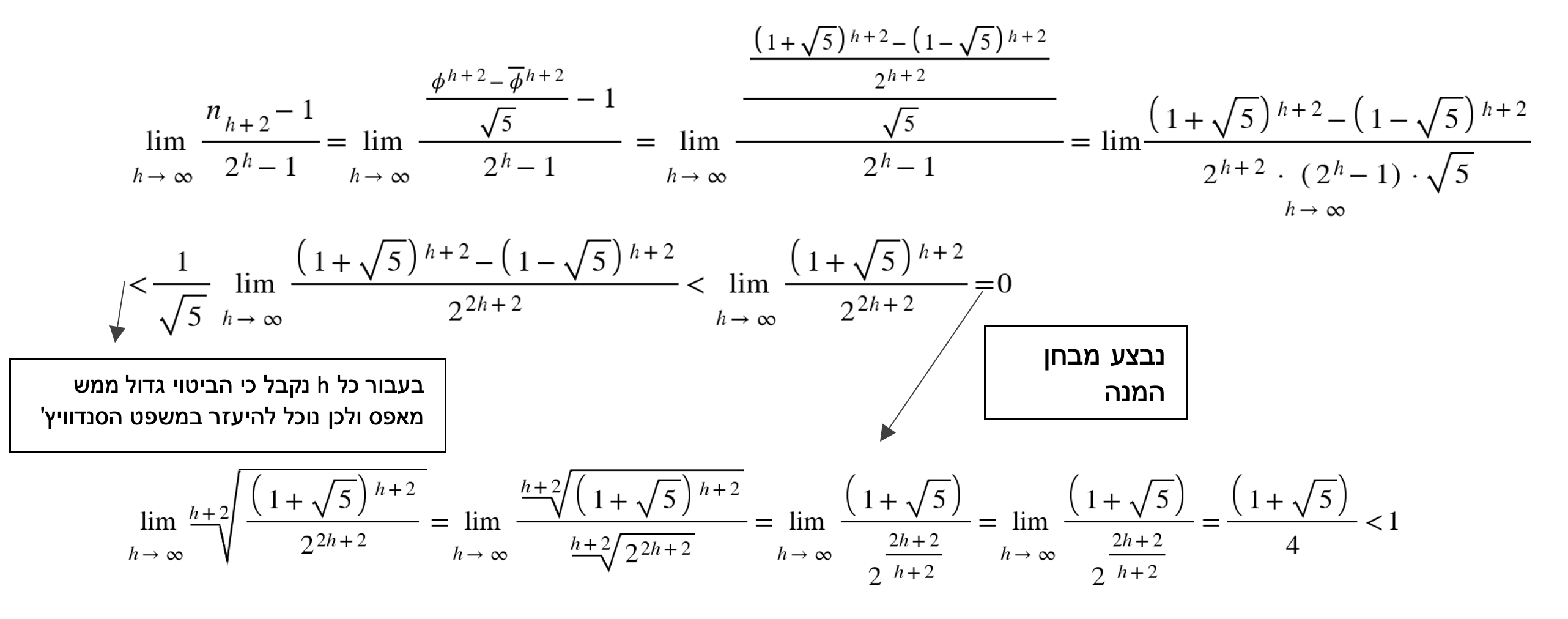
|TR| = 2h-1

וכך ש TL עץ פיבונאצ'י בעל גובה h-1 ולכן, ע"פ ההרצאה:

|TR| = nh+2-1 כאשר ni= .

נראה כי מתקיים

ומזה נקבל לפי הגדרה כי מתקיים: כנדרש



לסיכום, קיבלנו כי בעבור הדוגמא הנגדית שמצאנו ולכן ישנו עץ AVL עבורו לא מתקיים וזאת מפריך את הטענה כנדרש.



1. *מבנה הנתונים אשר תומך בביצוע הפעולות שצוינו לעיל, כל פעולה בסיבוכיות המותרת לה, הינו מבנה נתונים מסוג Trie. נממש מבנה Trie אשר נבנה ומנווט, כפי שלמדנו בהרצאה, על בסיס ערכי ה-desc המתקבל מהמשתמש באופן לקסיקוגרפי. נוסף לכך, נתעד בכל צומת את מחיר המוצר הזול ביותר בתת העץ שלה.*

*תחילה נפרט אודות מבנה ה-trie שלנו. כחלק מהמבנה יהיו לנו סה״כ שני ״סוגים״ שונים של צמתים:*

*שורש ה-trie – צומת אשר יכיל מערך מצביעים לצמתים בגודל 27 כך שכל מקום במערך מתייחס לאות אחרת באלף בת האנגלי ולסימן $ (בפועל המצביעים הם לצמתים בנים). כמו כן,* בכדי לאפשר לבצע פעולת Undo נתעד בשורש של ה-Trie את פעולת ה-Add האחרונה שבוצעה, כלומר המחיר ותיאור של אותו המוצר אשר הוסף אחרון.

*צומת ״רגיל״ – צומת אשר יכיל מערך מצביעים לצמתים בגודל 27 כך שכל מקום במערך מתייחס לאות אחרת באלף בת האנגלי ולסימן $ (בפועל המצביעים הם לצמתים בנים). כמו כן, נחזיק מצביע נוסף לצומת האב של צומת זה. נוסף על כך, בכל צומת מסוג זה נתעד מספר אשר יהווה ״המחיר של המוצר הזול ביותר הנמצא בתת העץ״ של צומת זה.*

כעת נסביר אודות אופן מימוש הפעולות הנחוצות בסיבוכיות שהוגדרה לעיל:

פעולת **init**() – יצירת שורש חדש לעץ trie עם מערך ריק בגודל 27 (מצבעיו nullptr). כמו כן, התיעוד בו של המוצר האחרון אשר הוסף (p ו-desc) יהיו null-ים גם כן. סה״כ O(1).

פעולת **: נבצע את הפעולה בדיוק כפי שלמדנו בהרצאה כאשר אופן הניווט בעץ הינו מבוסס על ה-**desc **המתקבל. בכל צעד או שננוע בתוך העץ הקיים או שנייצר צומת חדש התואם את האות הנוכחית ב-**desc. **בעת תנועה בעץ הקיים נבחן לכל צומת את ״המחיר של המוצר הזול ביותר של תת העץ שלו״, במצב בו מחיר זה יהיה גבוה מהערך** p **אשר התקבל כפרמטר אז נעדכן ערך זה להיות** p**. מצד שני, לכל צומת חדש שנייצר נתעד בתוכו ב-״מחיר המוצר הזול ביותר אשר נמצא בתת העץ שלו״ את** p **באופן אוטומטי. באת כל תנועה בעץ נעדכן את המצביעים הרלוונטיים בכל צומת כנדרש. כלומר, במידה ומדובר בתנועה בין צמתים קיימים אזי לא נעדכן דבר בצמתים אלו. במידה ומדובר בתנועה בין צומת קיים לצומת חדש אזי נעדכן את מערך המצביעים בצומת האב כנדרש ונעדכן את מצביע האב בצומת הבן כנדרש. לאחר ניווט בעץ והגעה לסוף ה-** desc **נקשר את הצומת האחרון שיוצר לצומת $ בעזרת המצביע הרלוונטי ל-$ במערך. גם בצומת ה-$ נתעד את מחיר המוצר. קיבלנו כי סיבוכיות כל תנועה בעץ הינה** O(1) ומכיוון שאנו מבצעים תנועות בעץ בדיוק כגודל desc אז הסיבוכיות המתקבלת הינה כנדרש.

פעולת : נעזר במידע השמור בשורש העץ אודות פעולת ההכנסה האחרונה. בהתאם למידע השמור, נבצע תנועה בעץ של חיפוש הצומת המסיים $ התואם את המוצר האחרון שהוכנס . ברגע שנגיע לצומת זה נתחיל לבצע תנועה הפוכה לכיוון שורש העץ, נבצע תנועה זו ע״י שימור במצביעי האב של כל צומת. נסיר כל צומת אשר מערך המצביעים שלו מכיל רק מצביע יחיד עד שנגיע לצומת בו תנאי זה לא מתקיים כלומר גם כן . נדאג להפוך את המצביע במערך המצביעים, אשר הצביע לצומת הבן אשר נמחק, להיות nullptr. מכיון שבתע התנועה חזרה סיבוכיות המחיקה/עדכון של צומת הינה O(1) אזי נקבל כי הסיבוכיות הסופית של הפעולה הינה כנדרש.

פעולת **: ננווט בעץ בדיוק כפי שלמדנו בהרצאה וכאשר נגיע לסוף** desc **נחזיק את הערך p אשר שמור בצומת המסיים $. מכיוון שאנו מבצעים תנועות בעץ בדיוק כגדול ה-desc אזי סיבוכיות פעולה זו** הינה כנדרש.

פעולת **: ננוע בעץ באופן זהה לחלוטין לפעולת price שהוגדרה לעיל. במידה והגענו לצומת נתון ע״פ ה-**desc **ומתקיים כי המצביע שלו לצומת הרלוונטי הבא ב-**desc **הינו** null **אזי נחזיר null, לכן . אחרת, נמשיך לנווט בעץ עד סיום ה-desc, נחזיר את הערך של ״מחיר המוצר הזול ביותר בתת העץ״ אשר שמור בצומת שהגענו אליו, לכן גם במצב זה** .

# שאלה 2

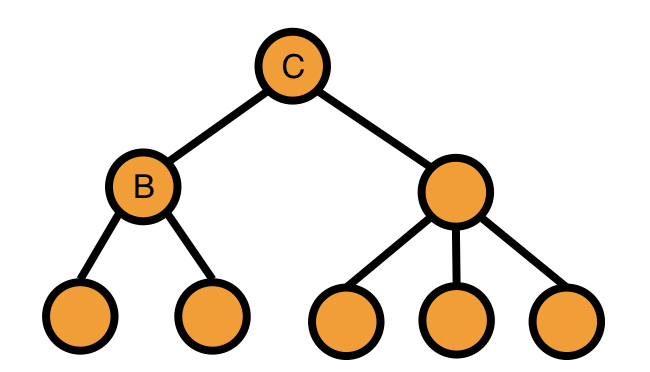
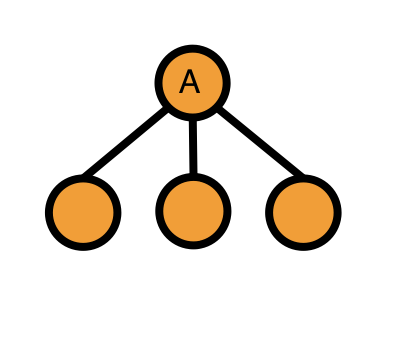
1. תקציר

תחילה נבצע טיול בשני העצים בכדי לאפיין את גובהם, מידע אשר משמש אותנו במהשך. פעולה בסיבוכיות של .

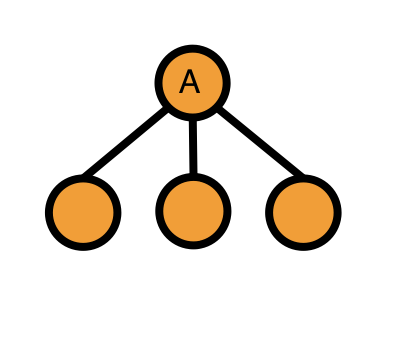
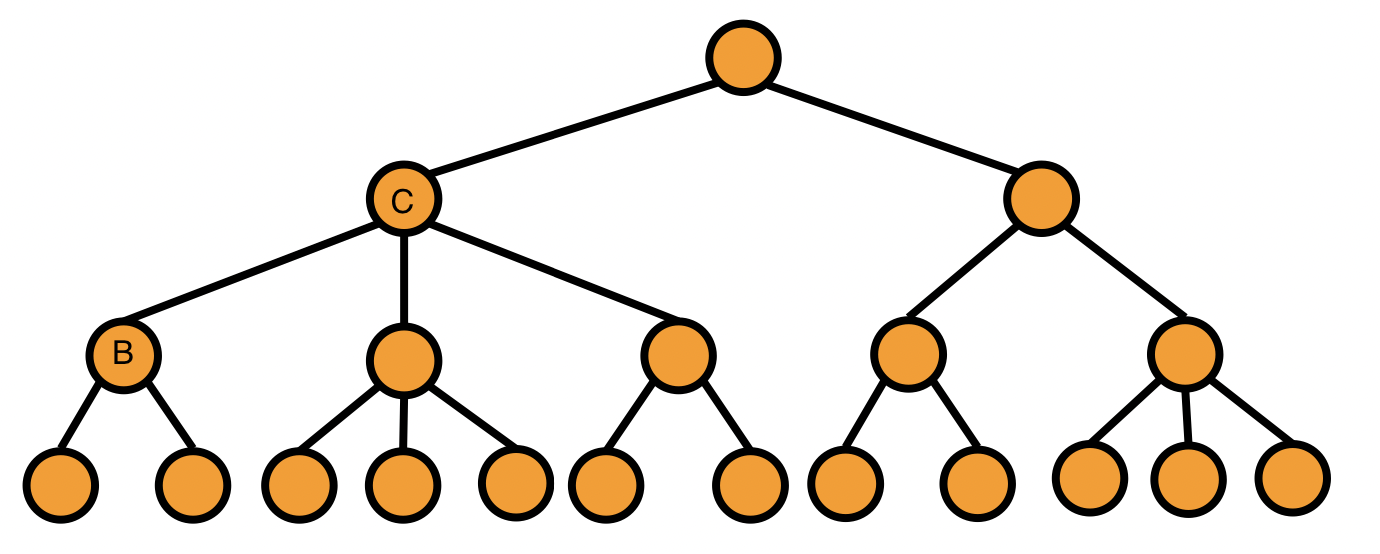
במידה ושני העצים הינם באותו גובה אזי כל שיש לעשות הינו פשוט לאחד בין שני העצים ע״י הוספת צומת חדש אשר יהיה שורש העץ המאוחד ויהיה בעל מפתח המחזיק את הערך שנמצא במפתח העלה בעל השמאלי ביותר בעץ T2. סיבוכיות הזמן הינה O הזמן שלוקח למצוא את הצומת הקטן ביותר בT2.

נניח בה״כ כי גובה העץ T1 הינו קטן מגובה עץ T2, במצב כזה נבחן את שורש העץ T1 (נסמנו A). נתבונן בצמתים בעץ T2, נחפש את תת העץ השמאלי ביותר אשר הינו תת העץ בגובה זהה ל-A (נסמנו B) , מציאת צומת זה נעשית ע״י ביצוע x מעברים מאב לבן שמאלי כאשר מתחילים משורש העץ ו-x=h1-h2, סה״כ סיבוכיות של O מתכונות עצי 2-3. נבחן את צומת האב שלB (חייב להיות צומת כזה מאופן הגדרת עץ 2-3 ומהעובדה כי גובה העץ T1 הינו קטן ממש מגובה T2), נסמנו C ונבדוק כמה בנים יש לו. נחלק למקרים:

1. שני בנים – במידה ו-C מחזיק שני בנים אזי פשוט נוסיף לו את צומת A כבן נוסף ובכך נחבר את כלל העץ 1T. נבצע ״תיקון לערכי הורים״ ע״פ הגדרת עצי 2-3, כלומר נשתמש במפתח של צומת B ונוסיפו בצד שמאל כמפתח חדש בצומת C. הדבר ישמור על עקרונות עצי 2-3 מהנתון כי כלל המפתחות של עץ T1 הינם קטנים יותר ומהעבודה כי שני העצים המקוריים הינם עצי 2-3 בעצמם. סה״כ סיבוכיות של O(1).

** **

1. שלושה בנים – במידה ו-C מחזיק שלושה בנים אזי נוסיף את A כבן רביעי ונוסיף לצומת C מפתח בצד שמאל בעל הערך של המפתח של צומת A (כלומר כעת הוא בעל 3 מפתחות שונים ו-4 בנים). כעת, נבצע פילוג לצומת C בדיוק כפי שנלמד בכיתה. נפריד אותו לשני צמתים, כל אחד בעל מפתח יחיד משני המפתחות הקיצוניים בצומת C ושני הבנים התואמים את המפתח הנתון. שני הצמתים אשר התקבלו כתוצאה מפילוג צומת C יהפכו לבניו של צומת האב של C (במקום צומת C אשר לא יהיה קיים יותר). אם לאותו צומת האב ישנם כעת שלושה בנים אזי סיימנו (סה״כ O(1) לפעולה זו). אחרת, אם יש לו כעת 4 בנים אזי נאלץ לבצע גם לו פילוג זהה לפילוג שתואר לעיל עבור צומת C (גם פה O(1)). במידה וצומת האב הינו בעצמו שורש העץ אזי נבצע לו פילוג מעט שונה, כאשר ההבדל היחיד הינו שנייצר צומת אב חדש אשר יצביע לשני הצמתים אשר התקבלו מפילוג השורש. הצומת החדש הזה יהפוך להיות השורש של העץ בפועל והמפתח אשר יוחזק בו יהיה המפתח האמצעי אשר היה קיים בצומת C..

****

נשים לב כי האלגוריתם שהוצג לעיל בהכרח יניב תוצאה סופית. אם שני העצים הנוכחיים הינם בעלי גובה זהה h אזי בפעולת איחוד בודדת נסיים את תהליך האיחוד. אחרת, מתקיים כי גובה אחד העצים (נסמנו H) ואילו גובה העץ השני הינו קטן יותר, נסמנו h. מכיוון שבכל פעולת ״פילוג״ מתקדמים תנועה 1 כלפי מעלה, לאחר לכל היותר h-H פעולות פילוג נסיים את תהליך האיחוד המלא, מהגדרת עצי 2-3 מתקיים כי גובה כל עץ הינו Olog(n). סה״כ נקבל כי סיבוכיות האלגוריתם הינה: כנדרש.

1. תחילה נשים לב כי האלגוריתם אשר הוצג לעיל מקיים שהפעולות היחידות אשר מצריכות סיבוכיות של הינן פעולות חיפוש מידע מיותרות במצב בו ידוע לנו מראש גבהי העצים וערכי המינימום והמקסימום בכל אחד מהעצים .

פירוט – ביצענו, באלגוריתם שהוגדר לעיל, חיפוש בסיבוכיות של ()O בשביל לאפיין את גבהי העצים. כמו כן, נעזרנו בחיפוש O בשביל למצוא את ערך המפתח של העלה המינימאלי בעץ T2 (כתלות בשווין ביחס הגודל בין שני העצים), חיפשנו את הערך הזה בשביל לכתוב אותו כמפתח בשורש החדש של איחוד העצים.

כלומר ניתן לוותר על פעולות החיפוש הנ״ל ובעצם לבצע את האלגוריתם בדיוק כפי שהוצג לעיל בלעדיהן. הסיבוכיות החדשה אשר תתקבל הינה . הסבר – אנו עתידים לבצע ״הוספה״ אחת של העץ הקטן לעץ הגדול ובנוסף לה אנו נבצע לכל היותר פילוגים של צמתים. מספר הפילוגים תלוי בכמות הצמתים במסלול אשר מחזיקים 3 צמתים לפני ההוספה של העץ הקטן לעץ הגדול.

1. תחילה נציין כי אנו נבצע את איחוד k העצים לפי סדר גודלם, כלומר מהקטן לגדול. מכיוון שעלינו לבצע k-1 ״הוספות״ של עצים קטנים לעצים גדולים ולכל פילוגים של צמתים אזי דרישת הסיבוכיות מתקיימת. נסביר את תהליך איחוד כלל העצים באופן סמיאינדוקטיבי, כאשר האינדוקציה תבוצע על מספר העצים אשר נאחד (נסמן מספר זה ב-n):

בסיס – k=2 תחילה נאחד שני עצים, כלומר נבצע איחוד בודד. ״נטפס״ משורש העץ הקטן יותר לשורש העץ המאוחד, במהלך הטיפוס נבצע פעולות ״פילוג״ לכל צומת עד בעל 4 בנים עד שהדבר לא יהיה נחוץ יותר. ידוע מסעיף ב׳ כי סיבוכיות הפעולה הזו הינה ולכן גם באופן ישיר מתקיים , כלומר דרישות הסיבוכיות מתקיימות כנדרש.

טענה – יכול להיות שכתוצאה מאיחוד שני העצים הקטנים ביותר (כאמור מתחילים איתם) ומהעובדה כי העץ השלישי בסדר הינו שווה בגובהו לעץ השני, אזי נקבל מהאיחוד עץ אשר הינו גבוה ב-1 (לכל היותר) מהעץ הבא בסדר (זאת מאופן הגדרת אלגוריתם האיחוד בין עצים שציינו לעיל). במידה ומצב כזה מתקיים אזי יש לבצע את האיחוד הבא בסדר באופן זהה לאלגוריתם שהוצג לעיל. נשים לב כי כל אשר יבוצע בפועל הינו הוספה של שורש העץ הקטן יותר להיות בן של שורש העץ הגדול יותר ובוודאות לא יבוצע פילוג נוסף לשורש זה. האפשרות היחידה בה העץ בעל המפתחות הקטנים ביותר נהייה גבוה יותר מהעץ הבא בסדר הינה שבוצע ״פילוג״ לשורש העץ כתוצאה מאיחוד האחרון, כלומר לשורש העץ הנוכחי יהיו רק שני בנים ונוכל להגדיר לו בן שלישי נוסף.

צעד האינדוקציה – נניח כי טענת הסיבוכיות מתקיימת עבור k-2 איחודים של k-1 עצים ונראה כי היא מתקיימת עבור k-1 איחודים של k עצים. כלומר סיבוכיות כלל פעולות האיחוד עד כה הינה . כעת אנו נאחד שני עצים. לפי סעיף ב׳ והטענה שצויינה לעיל, מתקיים כי סיבוכיות פעולה זו הינה או O(1) במידה והעץ של k-1 העצים המאוחדים הינו גדול יותר ב-1 מהעץ האחרון. ועל כן, סה״כ מתקיים כי סיבוכיות כלל הפעולות אשר בוצעו לאיחוד k העצים הינה .

לכן טענת האינדוקציה מתקיימת כמצופה ושיטת האיחוד שלנו של k עצים פועלת כנדרש.

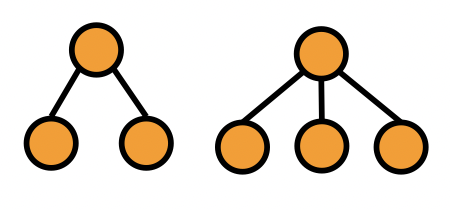
1. הרעיון הכללי הינו לקחת את העץ הנתון וליצור שני עצים חדשים ממנו ״תוך כדי תנועה״. אנו נבצע זאת ע״י חיפוש המפתח x והתייחסות בכל תנועה לתת העץ של הצומת האחרונה בה היינו. נחלק את תת העץ הזה לחלקים ונקטלג אותם, בסוף אנו נוסיף אותם לשני עצי התוצר החדשים שלנו, עץ גדול ממש ועץ קטן שווה.

תחילה נבצע חיפוש ראשוני לאפיון גובה העץ הנתון .

כעת נבצע חיפוש נוסף של האיבר x אלא שהפעם נבצע מספר מוגבל של פעולות בכל תנועה. בכל פעם שאנו נעבור מצומת אב לצומת בן נבחן את תת העץ בצומת האב בו היינו ונחלקו לשניים, חלק ימני וחלק שמאלי. קו החילוק נקבע בעזרת המסלול אשר נלקח בפועל בשביל להגיע לאיבר x. החלקים הללו הינם תתי עצי 2-3 ואנו נקטלג אותם לעצים שאנו עתידים להחזיר, העץ הגדול והעץ הקטן שווה מאיבר x. בנוסף

הסבר אודות המקרים השונים – במידה ולצומת האב ישנם שלושה בנים ואנו בחרנו ללכת לבן האמצעי, אזי הבן הימני יהיה השורש של החלק הימני בתת העץ (נוסיפו לעץ גדול ממש) והבן השמאלי יהיה השורש של החלק השמאלי בתת העץ (נוסיף אותו לעץ קטן שווה). במידה ובחרנו ללכת לבן הימני אזי נגדיר כי צומת האב יהיה השורש של תת העץ אשר נוסיף לעץ קטן שווה, זאת לאחר שנבטל את המצביע שלו לבנו הימני, המפתח שלו יהיה רק המפתח השמאלי שלו (נשים לב כי פעולה זו יוצרת אפשרות בה נאלץ לאחד בין עצים בגובה זהה, נתייחס למצב זה מאוחר יותר). במצב כזה לא יהיה תת עץ להוסיף לעץ הגדול ממש. במידה ולצומת האב ישנם שני בנים ואנו בחרנו ללכת לבן השמאלי אזי הבן הימני יהיה השורש של תת העץ שנוסיף לעץ הגדול ממש.

להלן מספר תמונות להמחשה:



נחזור על תהליך הקטלוג הזה עד שנגיע לאיבר x, נגדיר כי נוסיף גם אותו לעץ הקטן שווה.

טענת עזר איחוד של עצים בעלי אותו גודל – במידה ומתקיים כי רשימת העצים הסופית שיש לאחד בשביל לקבל את אחד משני עצי התוצר הסופיים מכילה שני עצים צמודים בעלי אותו גובה, מתקיים כי שורשי העצים הללו היו בנים של אותו האב. כלומר אנו יודעים מה המפתח שנרצה להשתמש בו בשביל האיחוד שלהם. נציין כי גם לאחר האיחוד הזה במידה ופעם נוספת מתקיים כי עלינו לאחד בין העץ אשר התקבל לאחד איחוד זה ועץ נוסף בגובה זהה לו אזי הטענה עדיין תקפה.

כעת נציין כי בכל תנועה שאנו מבצעים העצים החדשים שאנו מגדירים כי יש להוסיף לעצי התוצר הינם בהכרח קטנים מהעצים שכבר הוגדרו לפני כן, אלא במצב הקצה אשר הגדרנו לעיל בסוגריים בו יתקיים כי ישנם שני עצים באותו הגדול. מכיוון שישנם לכל היותר רק שני עצים בעלי אותו גובה נתון h ומכיוון שאנו יודעים את הגבהים של העצים והמפתחות הפוטנציאליים שנרצה להכניס בעת איחוד עצים שויים אזי ניתן להעזר באלגוריתם אשר הוצג בסעיף ג׳ בשביל לאחד את שתי רשימות העצים הללו לשני העצים הרצויים העץ גדול ממש ועץ קטן שווה. פעולה זו תבוצע פעמיים בסיבוכיות של .

לכן סה״כ לאחר ביצוע האלגוריתם הנ״ל נקבל כי הסיבוכיות שלו הינה כנדרש.

# שאלה 3

*א.*

*מאתחל את המבנה עם פרמטר k.*

*סיבוכיות זמן: .*

*נאתחל מבנה נתונים המכיל את השדות הבאים:*

* *הפרמטר – מאותחל ע"י k*
* *קצה תחום שמאלי (נסמנו min ) עבור כל מי שמקיים את התנאי – מאותחל לערך שגיאה מכיוון שאין קטעים במערכת*
* *קצה תחום ימני (נסמנו max ) עבור כל מי שמקיים את התנאי – מאותחל לערך שגיאה מכיוון שאין קטעים במערכת.*
* *עץ דרגות "ימני" (נסמנו TR ) אשר יכיל בצמתים את הקצוות הימניים ממויינים בסדר עולה של הקטעים ומונה למספר המופעים של קצה הקטע. – מאותל לעץ ריק.*
* *עץ דרגות "שמאלי" (נסמנו TL ) אשר יכיל בצמתים את הקצוות השמאליים ממויינים בסדר עולה של הקטעים ומונה למספר המופעים של קצה הקטע. – מאותחל לעץ ריק.*
* *עץ דרגות אשר כל צומת יכיל קטע ומונה עבור כמה פעמים הוכנס(נסמנו T )*

*סה"כ נבצע מס' סופי של פעולות ולכן סיבוכיות הזמן תהיה כנדרש.*

*מכניסה למבנה קטע שמתחיל בנקודה , ומסתיים בנקודה .*

*סיבוכיות זמן: , כאשר הוא מספר הקטעים במבנה.*

*תחילה נבצע* ***הכנסה*** *של y לTR, הכנסה של x לTL (אם קצה הקטע נמצא כבר, נעדכן את המונה) והכנסה של הקטע לT– הכנסה של איבר לשלושה עצים*

*לאחר מכן נבדוק בעבור כל אחד מהגבולות האם: ( x < min) והאם (y > max)*

*ונחלק למקרים בהתאם לתוצאה שהתקבלה.*

*אם x < min:*

*"נגדיל את התחום"*

*לכל הקטעים שהתחילו עם או מימין לx נוסף עוד קטע שהתחיל לפניהם או איתם. לכן נוצר מצב בו יכולים להיות עוד קטעים שהפכו להיות k-מרכזיים ולכן יש צורך בעדכון הגבול השמאלי. הוספנו רק קטע אחד משמאל ולכן* ***נעדכן*** *את הגבול להיות קצה שמאלי של קטע אחד משמאל לקצה השמאלי של הגבול הנוכחי(min) יכול להיות גם קצה הקטע שנוסף.*

*את הגבול החדש נוכל למצוא בעזרת מציאת האיבר הקודם בin-order לגבול הנוכחי*

*אחרת*

*כל הקטעים שמתחילים עם או אחרי כבר נמצאים בתוך הקצה השמאלי לתחום של הk-מרכזיים ולכן הוספת קצה הקטע x אינה משנה דבר.*

*אם y > max:*

*"נגדיל את התחום"*

*לכל הקטעים שהסתיימו עם או משמאל לy נוסף עוד קטע שנגמר אחריהם או איתם. לכן נוצר מצב בו יכולים להיות עוד קטעים שהפכו להיות k-מרכזיים ולכן יש צורך בעדכון הגבול הימני. הוספנו רק קטע אחד מימין ולכן* ***נעדכן*** *את הגבול להיות קצה ימני של קטע אחד מימין לקצה הימני של הגבול הנוכחי(max).*

*את הגבול החדש נוכל למצוא בעזרת מציאת האיבר הבא בin-order לגבול הנוכחי*

*אחרת*

*כל הקטעים שמסתיימים עם או לפני y כבר נמצאים בתוך הקצה הימני לתחום של הk-מרכזיים ולכן הוספת קצה הקטע y אינה משנה דבר.*

*סה"כ נבצע מס' סופי של ולכן נקבל כי סיבוכיות הזמן היא*

*מוציאה מהמבנה קטע שמתחיל בנקודה , ומסתיים בנקודה .*

*סיבוכיות זמן: , כאשר הוא מספר הקטעים במבנה.*

*פעולה זאת דומה באופייה לפעולת הinsert והפעולות מתבצעות בסדר הפוך.*

*תחילה נבצע בדיקה האם הקטע נמצא בT אם לא, נזרוק שגיאה.*

*אחרת, נבצע* ***הסרה*** *של הקטע y מTR והסרה של x מTL (אם המונה גדול מאחד, נעדכן את המונה) – הסרה של איבר משלושה עצים*

*לאחר מכן נבדוק בעבור כל אחד מהגבולות האם: ( x <= min) והאם (y >= max)*

*ונחלק למקרים בהתאם לתוצאה שהתקבלה:*

*אם x <= min:*

*"נצמצם את התחום"*

*לכל הקטעים שהתחילו עם או מימין לx הוסר קטע שהתחיל לפניהם או איתם. לכן נוצר מצב בו יכולים להיות פחות קטעים שהפכו להיות k-מרכזיים ולכן יש צורך בעדכון הגבול השמאלי. הסרנו רק קטע אחד משמאל ולכן* ***נעדכן*** *את הגבול להיות קצה שמאלי של קטע אחד מימין לקצה השמאלי של הגבול הנוכחי(min).*

*את הגבול החדש נוכל למצוא בעזרת מציאת האיבר הבא בin-order לגבול הנוכחי*

*אחרת*

*כל הקטעים שמתחילים אחרי x ימשיכו להימצא בתוך הקצה השמאלי לתחום של הk-מרכזיים ולכן הסרת קצה הקטע x אינה משנה דבר(יש מספיק קטעים שמתחילים לפניהם גם בלי x).*

*אם y >= max:*

*"נצמצם את התחום"*

*לכל הקטעים שהסתיימו עם או משמאל לy הוסר קטע שנגמר אחריהם או איתם. לכן נוצר מצב בו יכולים להיות פחות k-מרכזיים ולכן יש צורך בעדכון הגבול הימני. הסרנו קטע אחד מימין ולכן* ***נעדכן*** *את הגבול להיות קצה ימני של קטע אחד משמאל לקצה הימני של הגבול הנוכחי(max).*

*את הגבול החדש נוכל למצוא בעזרת מציאת האיבר הקודם בin-order לגבול הנוכחי*

*אחרת*

*כל הקטעים שמסתיימים עם או לפני y ימשיכו להיות בתוך הקצה הימני לתחום של הk-מרכזיים ולכן הסרת קצה הקטע y אינה משנה דבר.*

*סה"כ נבצע מס' סופי של ולכן נקבל כי סיבוכיות הזמן היא*

*מקבלת קטע הנמצא במבנה, מחזירה אם הקטע הוא קטע -מרכזי,   
ו- אחרת. ניתן להניח כי הקלט חוקי.*

*סיבוכיות זמן: .*

*נחזיר את הבדיקה הבאה: (min<=x && max>=y)*

*סה"כ התבצעו השוואות סופיות בין מס' סופי של משתנים ולכן*

*ב.*

*הרעיון הכללי:*

*עץ דרגות הממויין לפי נקודת תחילת כל קטע*

*מאתחל את מבנה הנתונים עם שתי קבוצות ריקות.*

*סיבוכיות זמן: .*

*מבנה הנתונים:*

* *עץ דרגות (נסמנו T) מאותחל להיות עץ ריק , הממוין בסדר עולה אך ורק ע"פ ה-x (לדוגמה, העץ יזהה את הקטע (0,1) ב-A ואת הקטע (0,2) ב-A* ***כזהים****) כאשר כל צומת יכיל בתוכו:*

1. *x של הקטעים שהוכנסו לNode זה.*
2. *קטע yA - ה-y הינו הy של הקטע שהוכנס ראשון מ - A לNode זה.*
3. *קטע yB - ה-y הינו הy של הקטע שהוכנס ראשון מ - B לNode זה.*
4. *מונה עבור מספר הקטעים ב-A* ***המתחילים באותו ה – x*** *נסמנו בCA*
5. *מונה עבור מספר הקטעים ב-B* ***המתחילים באותו ה – x*** *נסמנו בCB*

* *מצביע(נסמנו opt) אשר מחזיק את הצומת אשר הינו הכי קרוב לקיום התנאים של הפונקציה השלישית -* *ומתעדכן בinsret (פירוט בהמשך). נאתחל לNULL*
* *משתנה אשר יחזיק את ״הסוג״ אליו ה-opt מתייחס בתוך הצומת (כלומר A או B). נחוץ מכיוון שייתכן והצומת ש-opt מצביע עליו מכיל התייחסויות הן ל-A והן ל-B. נאתחל לNULL*
* *משתנה השומר את מצב האיזון (נסמנו balance) בהתאם לתנאי הפונקציה השלישית ביחס ל-opt, כאשר 0 מראה על מצב מאוזן (במצב זה הפונקציה תחזיר את הקטע לפי opt, נפרט בהמשך איך) ולכל balance<0 מראה על מצב שבו ישנם יותר קטעים מ-B המתחילים עם opt או אחריו מאשר קטעים מ-A המתחילים איתו או לפניו. ואותו הדבר רק הפוך בעבור balance>0. מאותחל לערך שגיאה .*

*מכניסה את הקטע לקבוצה , כאשר .*

*סיבוכיות זמן: , כאשר הוא מספר האיברים בשתי הקבוצות יחד.*

*תחילה נכניס את הקטע לעץ T. במידה וה-x של הקטע קיים, נבדוק את ערך המונה הרלוונטי לסוג הקטע (A או B) אם שווה ל-0, נעדכן את ערך ה-y הרלוונטי בצומת להיות ערך ה-y של הקטע שהתקבל. אחרת לא נשנה דבר בערך ה-y הרלוונטי של צומת זה, נעדכן את CG בהתאם לצבעו.*

*נרצה לתחזק את opt, נחלק למקרים:*

1. *אם opt הוא NULL נעדכן אותו להיות הקטע שהוכנס. מכיוון שאין עוד קטעים במערכת ולכן, מכל קבוצה, מספר הקטעים המתחילים לפניו ולאחריו שווים לאפס ובפרט שווים ולכן עומד בתנאי הפונקציה השלישית. לאחר מכן נעדכן את balance להיות 0.*
2. *אם הוכנס קטע השייך לA המתחיל* ***אחרי ממש*** *opt או קטע השייך לB המתחיל* ***לפני******ממש*** *opt הקטע אינו משפיע כלל על תנאי הפונקציה השלישית ולכן לא נעשה דבר המשפיע opt או על balance ונסיים.*
3. *אחרת:*
   1. *אם הוכנס קטע השייך לA המתחיל* ***לפני או עם*** *opt* ***נוסיף*** *1 ל-balance*
   2. *אם הוכנס קטע השייך לB המתחיל* ***אחרי או עם*** *opt* ***נפחית*** *1 מ-balance*
4. *כעת נבדוק: האם balance == 0*
   1. *אם כן, המצב חזר לאיזון ולכן opt עומד בתנאים ויוחזר במקרה של קריאה לפונקציה השלישית.*
   2. *אם לא, ננסה לחזור לאיזון:*
      1. *תחילה, במידה וישנו בצומת אשר עליו מצביע opt התחלה של קטע כלשהו מהסוג השני – ש-opt אינו מצביע עליו, אזי נבחן הצבעה פוטנציאלית על קטע זה. כלומר נבחן שינוי של ערכו של ה-balance. נוסיף לו-2 או נחסר ממנו 2 כתלות בכיוון המעבר. במידה ו-opt התייחס קודם לכן לקטע מסוג A אזי נבחן מה יקרה אם נוסיף לbalance 2, אחרת נחסר 2. לבינתיים נשמור ערך זה לבחינה עתידית.*
      2. *כעת נבחן את הצומת הרלוונטי הקרוב ביותר לצומת הנוכחית של opt. במידה והקטע אשר הוכנס כעת ושינה את המאזן הינו מסוג A אזי נבחן את הצומת הקרוב ביותר לצומת זה שנמצא בדיוק לפניו בסדר inorder כלומר הצומת ה- prev in-order* לכל היותר *. אם הקטע אשר הוכנס הינו מסוג B אזי נבחן את הצומת הבא בסדר inorder, צומת ה- next in-order* לכל היותר *. אחרי שנגיע לצומת החדש אזי נאפיין את המאזן החדש המתקיים ״מהצבעה״ על שני סוגי הקטעים הנמצאים בו (במידה ומכיל רק סוג אחד של קטעים אז נבחן רק אותו). אופן חישוב ה-balance החדש יתקיים כדלקמן:*

|  |  |  |  |
| --- | --- | --- | --- |
| *הסוג של opt ישן* | *הצומת שבוחנים* | *מסתכלים כעת על* | *resulted balance* |
| *A* | *prev in-order* | *A* | *Balance פחות 1 פחות כמות ה-b בצומת החדש פחות כמות ה-a בצומת הישן* |
| *A* | *prev in-order* | *B* | *Balance ועוד 1 פחות כמות ה-b בצומת החדש פחות כמות ה-a בצומת הישן* |
| *A* | *next in-order* | *A* | *Balance ועוד כמות ה-a בצומת החדש ועוד כמות ה-b בצומת הישן.* |
| *A* | *next in-order* | *B* | *Balance ועוד 2 ועוד כמות ה-a בצומת החדש ועוד כמות ה-b בצומת הישן* |
| *B* | *prev in-order* | *A* | *Balance פחות 2 פחות כמות ה-b בצומת החדש פחות כמות ה-a בצומת הישן.* |
| *B* | *prev in-order* | *B* | *Balance פחות כמות ה-a בצומת הישן של opt פחות כמות ה-b בצומת החדש* |
| *B* | *next in-order* | *A* | *Balance ועוד כמות ה-a בצומת החדש ועוד כמות ה-b בצומת הישן פחות 1* |
| *B* | *next in-order* | *B* | *Balance ועוד כמות ה-b בצומת הישן של opt ועוד כמות ה-a בצומת החדש* |

*כעת נבצע השוואה בין ארבעת הערכים הבאים: ה-balance של opt המקורי, ה-balance לאחר הצבעה על הסוג השני באותה הצומת של opt, ה-balance בצומת הקרובה הרלוונטית מסוג A (במידה וקיים) וה-balance בצומת הקרובה הרלוונטית מסוג B (במידה וקיים). נבחר את האופציה אשר מניבה את ה-balace הכי קרוב ל-0, נעדכן את opt ואת המשתנה אשר מתעד את סוג הצומת המאוזן ביותר בהתאם לתוצאת ההשוואה הנ״ל .*

*בכל סיבוב נוסף לכל היותר קטע אחד ולכן לא יתכן שקיימת אפשרות לפתרון טוב יותר על ידי ביצוע בחינה של צומת נוסף מעבר לצומת הצמוד שהוגדר כרלוונטי (כי אם היה כזה היינו מבצעים אותו בסיבוב שלפני) ולכן נסתפק בניסיון תיקון אחד לסיבוב גם אם עדיין לא מאוזן*

*סה"כ נקבל מס' סופי של O(1) ומס' סופי של ולכן מתקיים*

*מחזירה קטע כלשהו הנמצא ב או ב , כך שמספר הקטעים המתחילים לפני (או איתו) ב שווה למספר הקטעים המתחילים אחריו (או איתו) ב , או מחזירה שגיאה אם לא קיים קטע כנדרש.*

*סיבוכיות זמן: .*

נבדוק: balance == 0

אם כן – נחזיר את הקטע מתוך opt, נעשה זאת בעזרת המשתנה אשר מגדיר לאיזה ״סוג״ opt מתייחס ואז בעזרת opt עצמו נפאיין את גבולות הקטע הרלוונטיים בצומת עליו opt מצביע. החזרת שדה והשוואה = סיבוכיות זמן

אם לא – נחזיר שגיאה = באותו אופן

# שאלה 4



תחילה נגדיר כי בחישובי הסיבוכיות בשאלה זו n מוגדר להיות המקסימום בין מספר האנשים ומספר המענקים (לפי piazza(.

תחילה נתאר את מבני הנתונים אשר בחרנו. נפתור בעיה זו בעזרת מימוש שני עצים:

עץ AVL אשר יכיל את כלל האנשים אשר הוזנו למערכת ויהיה ממוין ע״פ ה-id שלהם, כל צומת של עץ זה יתייחס לאדם אחד ויכיל בנוסף ל-id של האדם את שנת לידתו.

עץ דרגה של AVL אשר יכיל צמתים של שנות לידה זרות ויהיה ממוין על פיהן באופן כרונולוגי. מעבר לשנת לידה, כל צומת יכיל פרמטר נוסף אשר מגדיר את כמות הכסף הניתן לאנשים אשר במהלך המסלול לשנת הלידה שלהם הם חוצים את הצומת המדובר.

כעת נסביר את אופן ביצוע הפעולות:

פעולות **: נייצר שני שורשים לעצי AVL ריקים ().**

פעולות **: כל שנבצע במסגרת פעולה זו זה חיפוש בעץ ה-AVL הקיים של אנשים ע״ב הת״ז () ונוסיף לעץ צומת חדש במקום הרלוונטי (). בצומת זה נתעד את ת״ז הבן אדם ושנת לידתו. סה״כ**  כנדרש.

פעולות **: בשביל לממש פונקציה זו אנו נבצע את הפעולות הבאות:**

1. **נכניס בסדר הבא לעץ ה-AVL של השנים את הצמתים y2 , y1-1 , y1 ו- y2+1 , כלומר בכל פעולות הכנסה נכניס לכל היותר 4 צמתים (במידה והשנה כבר קיימת בצומת כלשהו בעץ לא נבצע הוספה שלה). לכן המספר הכולל של צמתים בעץ הינו לכל היותר n4 (יכול להיות גם פחות במידה וכמות האנשים הינה גדולה מכמות המענקים). לאחר הכנסה של כל אחד מארבעת ה-node-ים אנו נבצע גלגול במקרה הצורך בשביל לשמר את היות העץ AVL. את אופן הגדרת הגלגול נסביר בהמשך.**

**כל פעולת חיפוש או הכנסה לעץ עבור 4 = 𝑛0 > n נעשת בסיבוכיות: F(n) ≤ log(4n) + c = O(log(4n)) = O(log(4) + log(n)) ≤ O(2\*log(n)) = O( log(n))**

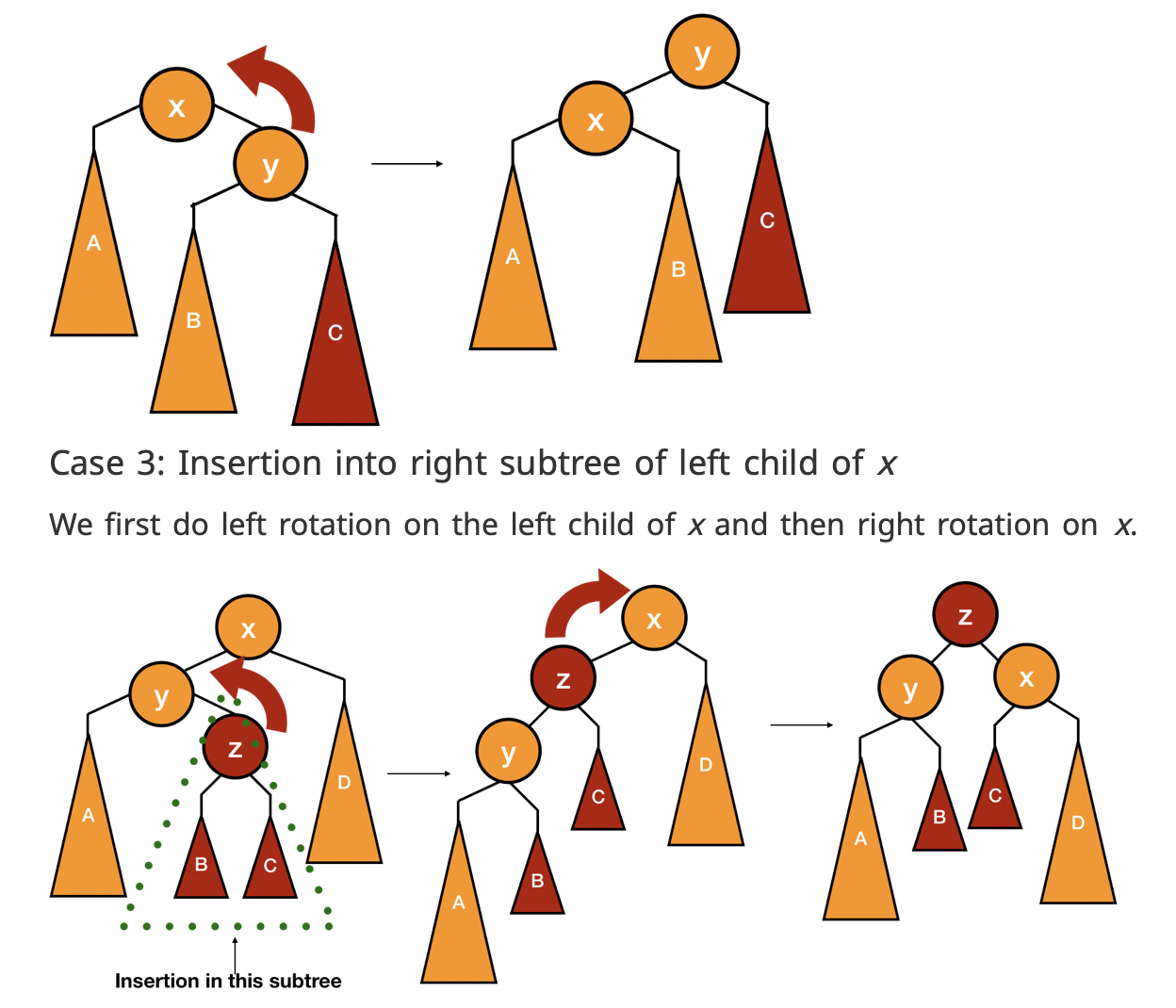
1. **כעת נממש פונקציה בשם Add(i,x) אשר תחפש את i החל מהשורש (בדיוק בהתאם לאלגוריתם אשר הוצג בתרגולו של עידו). בכל רצף פניות ימינה נוסיף x לצומת הראשון ממנו פנינו ימינה. אם רצף הפניות ימינה נשבר שמאלה נקטין ב-x את הצומת ממנו פנינו שמאלה. כשנגיע לצומת היעד, אם לא הגענו לאחר רצף פניות ימינה נגדיל את ערכו ב-x, ובכל מקרה נקטין את ערך בנו הימני ב-x (אם קיים). נפעיל את הפונקציה פעמיים:**
   1. **Add(,a)**
   2. **Add(,-a)**

**(כתוצאה מביצוע הנ״ל אנו נקבל כי חיפוש בעץ השנים של שנה נתונה יניב לנו את סכום המענק אשר האנשים שנולדו בשנה זו זכאים לו, זאת ע״י סכימת המענקים המצוינים בכל צומת.**

1. **אופן ביצוע גלגול – לכל גלגול אשר נבצע בעת הוספת צמתים נדאג לשנות את ערכי המענקים המוגדרים בצמתים באופן כזה כך שלכל שסכום המענק המוחזר מכל מסלול נתון יישאר זהה לאחר ביצוע הגלגול הנחוץ. כלומר נעדכן את ערכי המענקים כך שגם לאחר הסכימה בסדר החדש יתקבלו אותם ערכים לכל שנה בעץ. נעשה זאת באופן הבא:**

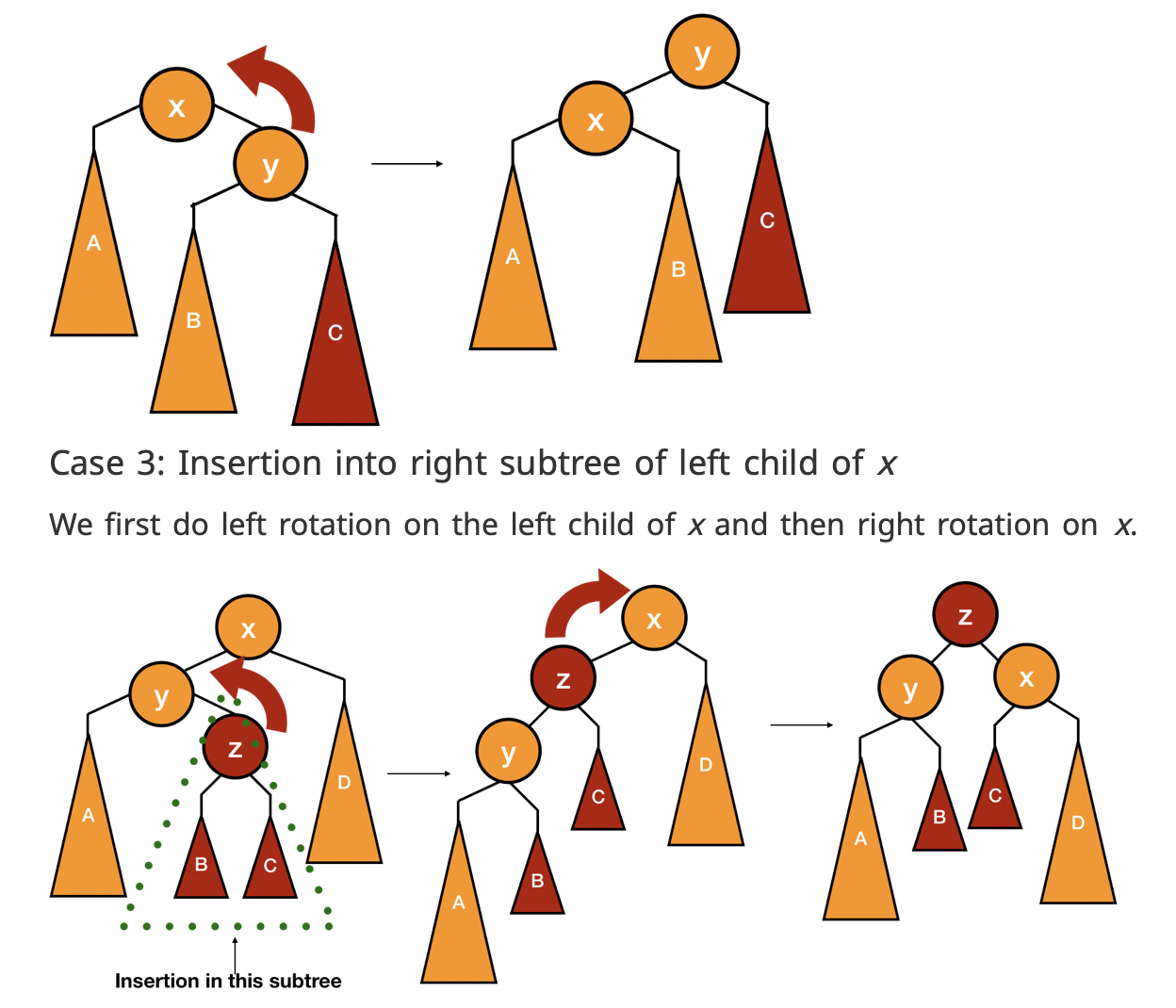
**בהתאם לציור המצורף מטה, נגדיר כי בעת ביצוע גלגול פשוט שמאלה אנו נחליף את ערכי המענק המוגדרים בצמתים באופן הבא:**

|  |  |  |  |
| --- | --- | --- | --- |
| **הצומת הרלווטי** | **ערך מענק מקורי בצומת** | **ערך מענק בצומת לאחר גלגול** | **סכום מענק של מסלול עד הצומת (נשאר קבוע לאחר גלגול)** |
| **צומת Y** | **המענק של Y** | **Y+X** | **X+Y** |
| **צומת X** | **המענק של X** | **-Y** | **X** |
| **שורש A** | **המענק של השורש A** | **A** | **A+X** |
| **שורש B** | **המענק של השורש B** | **B+Y** | **B+Y+X** |
| **שורש C** | **המענק של השורש C** | **C** | **B+Y+X** |



**מדובר בעדכון של 5 צמתים, שאר הצמתים בעץ ישמרו על ערכי המענקים בהם לכן מדובר בסיבוכיות של . בחינה של כלל המסלולים האפשריים בעץ מראה כי סכום המענק אשר כל שנה תקבל נשאר זהה גם לאחר הגלגול המתואר לעיל. נגדיר את הגלגול ימינה באופן סימטרי לחלוטין להגדרה הנ״ל כך שגם הוא יהיה ״משמר״.**

**מכיוון שגלגולים מתקדמים מצורת ימינה-שמאלה ושמאלה-ימינה הינם מורכבים בעצמם מהגלגולים הפשוטים אזי נקבל כי המענקים המתקבלים לכל שנה עדיין נשמרים גם לאחר ביצועם.**



פעולת**: תחילה נבצע חיפוש בעץ השחקנים ונאפיין את שנת הלידה של האדם הנתון . לאחר מציאת האדם נעשה חיפוש בעץ השנים ונבצע סכימה של כלל המענקים אשר מצוינים בצמתים בהם אנו חולפים.**

**כאשר נגיע לסוף החיפוש בעץ, כלומר לצומת אשר מכיל את השנה המבוקשת או ל-nullptr אז נחזיר את הערך אשר סכמנו לאורך המסלול כערך החזרה מהפונקציה.**

כל שיש לעשות הינו להוסיף משתנה בכל צומת של אדם פרטי בעץ האנשים. במשתנה זה נשמור את מה שהיה סכום המענק לו היו זכאים אנשים אשר נולדו בשנת הלידה של האדם הנ״ל ברגע בו האדם הזה התווסף לעץ. בעת סכימת המענק אשר אדם נתון זכאי לו אנו נבצע את אותו תהליך הסכימה אשר הוגדר לעיל ונחסר ממנו את המשתנה הנ״ל.

בפעולת ההכנסה של אדם חדש למערכת, נבצע קודם חיפוש של שנת הלידה שלו בעץ השנים, ונסכום את ערכי המענק בכל הצמתים במסלול החיפוש, עד שנמצא את השנה המבוקשת או שנגיע ל-null (כפי שעשינו בסעיף א בפונקציה Getgrantamount(ID)).

חיפוש בעץ בינארי ולכן . את הסכום שקיבלנו נשמור בצומת של אדם זה, ורק לאחר מכן נכניס אותו לעץ המתאים במערכת. הכנסה לעץ כמו קודם, ועוד מספר פעולות קבועות ולכן . קיבלנו כי סיבוכיות הפעולה של הכנסת אדם חדש למערכת לא נפגעה : addPerson (ID, Year) ≤ 2\* O( log (n)) + C = O( log(n))

כעת, כשנקרא לפונקציה GetGrantAmount , נמצא את האדם המבוקש בעץ של האנשים וניקח את סכום המענקים הקודם השמור אצלו, ולאחר חיפוש שנת הלידה שלו בעץ השנים וסכימת המענקים במסלול החיפוש, ממש כמו בסעיף א, נחסיר מסכום זה את הערך ששמור אצל האדם, ונחזיר : Returned= full value-value before entering

השינוי בפונקציה זו כולל רק מספר פעולות קבוע של גיה למשתנה והחסרת ערכים, ולכן הסיבוכיות שלה לא נפגעה : ))GetGrantAmount(ID) = O( log(n)) + C = n

1. ננהל מערך בגודל k של int-ים כאשר כל אינדקס בו מתייחס לשנה אחת מתוך תווך השנים החוקי סיבוכיות מקום – . בנוסף לתחזוקת העצים שהגדרנו לעיל, בכל פעם שנבצע הכנסה של תחום חדש של מענק כלשהו, נוסיף את ערך המענק באינדקס של השנה המוקדמת יותר כערך חיובי, ובאינדקס של השנה המאוחרת יותר ועוד 1 נוסיף את המענק במינוס. כמובן כי (כתלות בפעולות המבוצעות) יכול להתקיים מצב בו אינדקס נתון יחווה תחילה הוספה של ערך חיובי כתוצאה של פעולת ולאחר מכן חיסור מערכו כתוצאה של  **אחר**.

כאשר נקבל d כלשהו, נבחן את ערך המענק השמור ב-d עצמו נגדיר את אינדקס d להיות תשובת הדיפולט שלנו u, נסמן את ערך המענק של אינדקס זה ב-x. כעת נקפוץ 9 פעמים ימינה ונאפיין את השנה בעלת המענק הגדול ביותר, נעשה זאת באופן הבא:

1. נתחיל לנוע ימינה ולבחון את תוכנו של כל אינדקס, במידה וזה ריק אזי מתקיים כי שווי המענק הניתן בשנה הזו הינו שווה למענק הניתן באינדקס של u, כלומר x (לכן אין חשיבות להחזיר דווקא את האינדקס החדש במקום d עצמו).
2. ברגע שנפגוש אינדקס לא ריק (כלומר אינדקס אשר שמור בו ערך חיובי או שלילי נסמנו a) אזי נסיק כי המענק הניתן מאינדקס של שנה זו הלאה נסמנו ב-y מקיים y = x + a.
3. אם y>x אז נעדכן את u כך שהתשובה הדיפולטיבית תהייה האינדקס החדש, כמו כן נעדכן את x כך שיהיה שווה בערכו ל-y הנ״ל. אם התנאי לא מתקיים אז u ללא שינוי יהיה תשובתנו בסיום הריצה.
4. בכל פעם שנפגוש אינדקס לא ריק נעדכן את ערכו של y בהתאם ונשווה אותו לערך של x, במידה ויש צורך לעדכן את u נעשה זאת בהתאם.

מכיוון שסה״כ אנו בוחנים 10 אינדקסים שונים ומבצעים מספר פעולות מוגבל בכל אחד מהם אזי סיבוכיות הזמן של כל הפעולה הזו הינה .