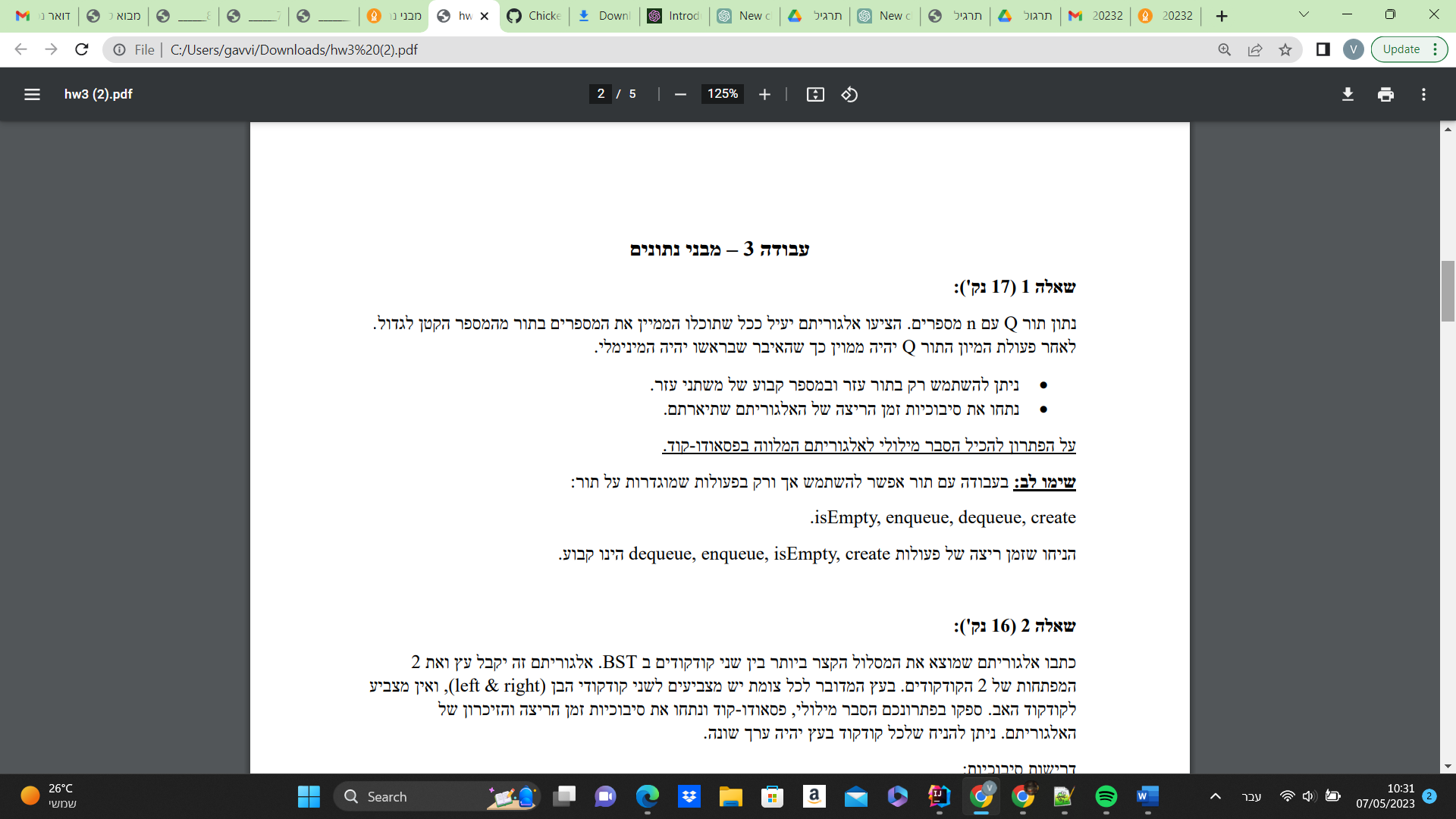
מבנה נתונים – עבודה 2

מגישים : ויקטור גברילנקו 209406255, תומר שולמן 208997551



הסבר האלגוריתם:

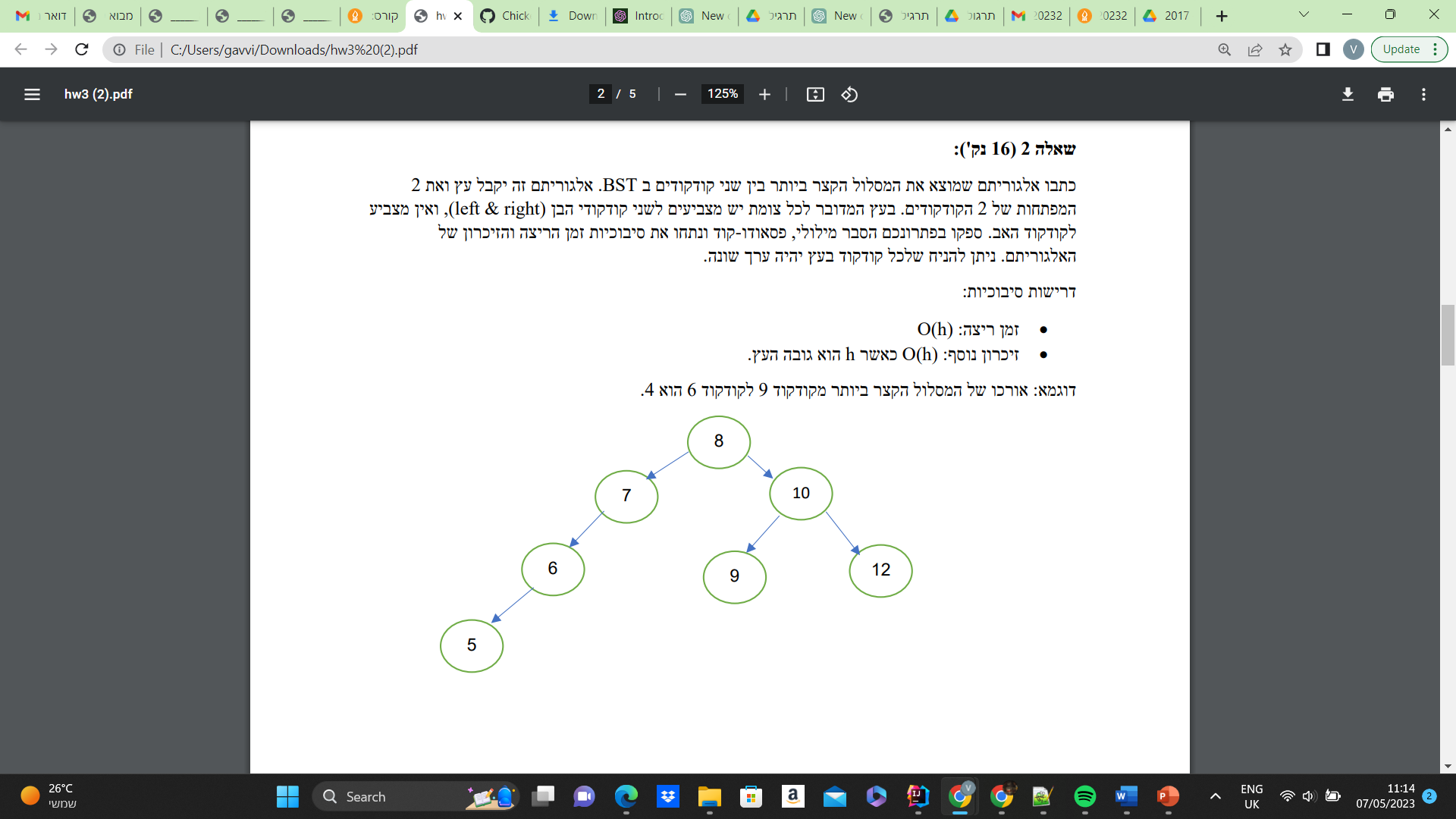
בלולאת הWHILE הראשונה בשורה 3 אנו מעבירים את התוכן של התור Q אל תוך תור העזר שיצרנו tmp ובנוסף סופרים את כמות האיברים בתור Q.

לאחר מיכן אני רצים בלולאת WHILE נוספת כל עוד התור עזר שיצרנו אינו ריק. הרעיון הוא בכל פעם לחפש את הערך המינימלי בתור העזר, להוסיפו לתור המקורי Q, ולבסוף למחוק אותו מתור העזר.  
בלולאת הFOR הראשונה בשורה 4.3 אנו מחפשים את האיבר המינימלי בתור העזר. בלולאת הFOR השניה בשורה 4.4 אנו מוחקים את האיבר המינימלי שמצאנו הרגע, ולבסוף מכניסים את אותו האיבר בשורה 4.5 לתור המקורי Q ומקטינים את גודל תור העזר באחד. בסוף האלגוריתם נקבל שבראש התור יעמוד האיבר המינימלי ונקבל תור ממוין מהקטן לגדול.

ניתוח סיבוכיות זמן ריצה:

באלגוריתם הנל יש לנו סהכ שתי לולאות מקוננות שנותנות לנו זמן ריצה של:

-------------------------------------------------------------------------------------------------------------------------

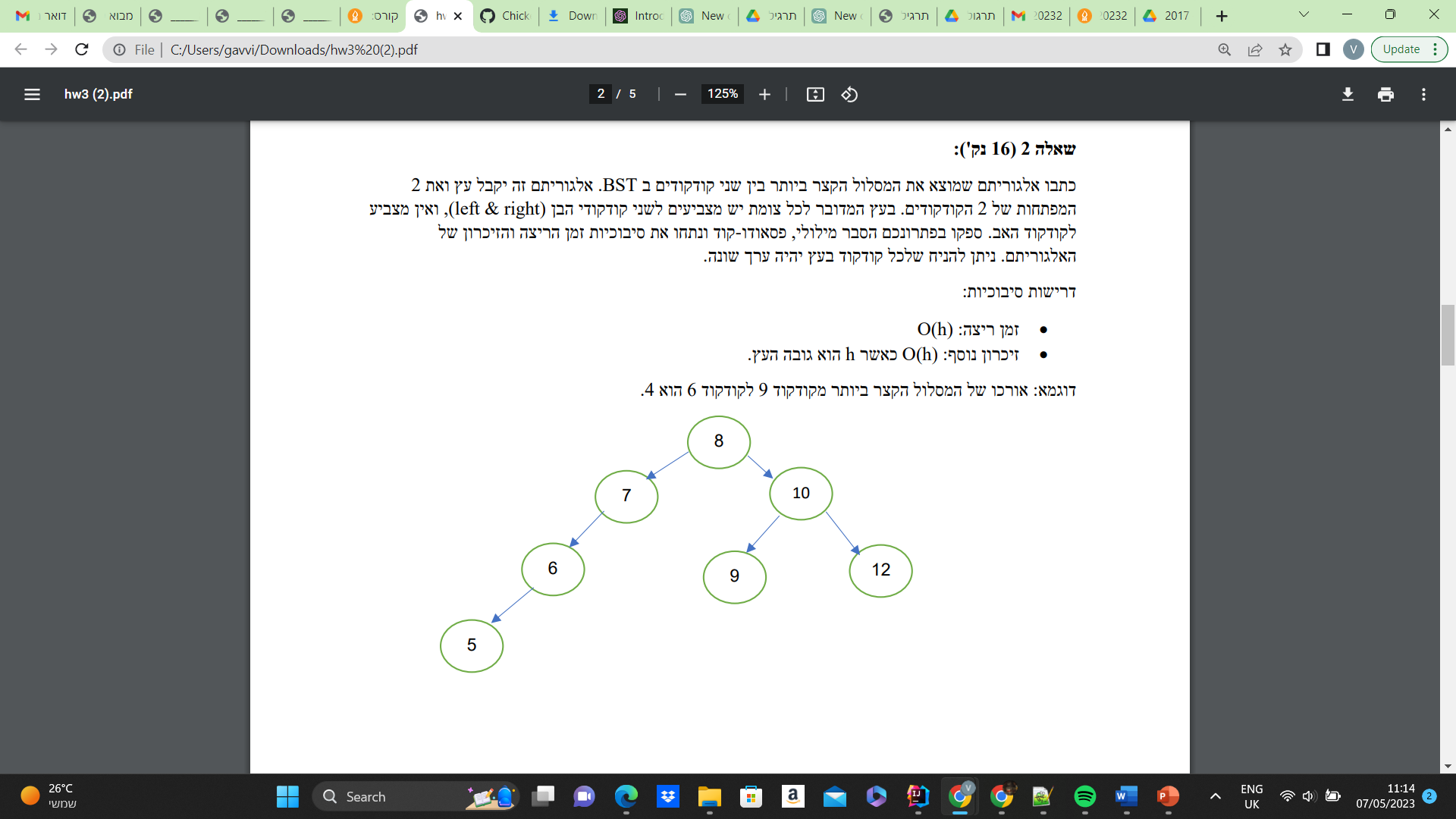


**findPath(T, left, right)**

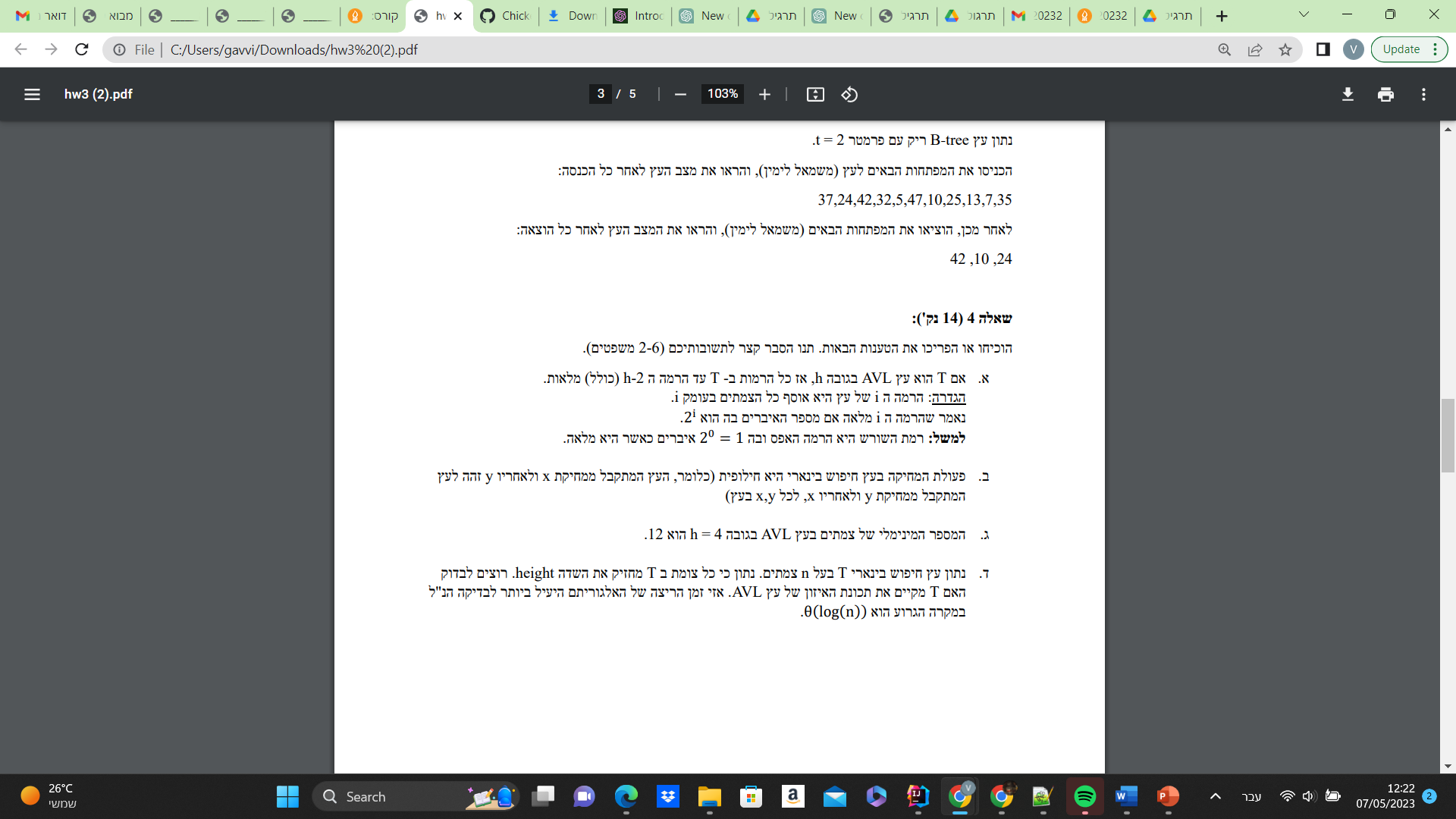
פירוט האלגוריתם:  
תחילה אנו מאתחלים שלוש רשימות מקושרות. לאחר מיכן נחפש את left ואת right בעץ הנתון. בעת החיפוש של כל אחד מהם, נכניס את כל האיברים במסלול החיפוש לכל רשימה בהתאם לאיבר שאותו מחפשים – עלות החיפוש הינה לכל היותר היות ואנו מחפשים את האיברים הללו בעץ חיפוש בינארי שגובהו הנתון הינו h.

לאחר מיכן, נרוץ בלולאת WHILE על שני הרשמות המקושרות הללו בעזרת שני מצביעים שהגדרנו לטובת זאת.

במידה ושני המצביעים מצביעים על איבר המכיל בתוכו את אותו הערך בשתי הרשימות המקושרות (ז"א איבר זה הינו איבר משותף במסלול מהשורש [כולל] עד ל left/right) נכניס את אחד מהם לרשימה המקושרת השלישית ונקדם את שתי המצביעים אחד קדימה בכל רשימה מקושרת.  
במידה ואינם מצביעים על אותו ערך, נבדוק בכל אחד מהם שאינם NULL ונכניס אותם לרשימה המקושרת השלישית, ולבסוף נקדם את המצביעים של מי שאינו NULL אחד קדימה.   
לבסוף, יהיה לנו ברשימה המקושרת השלישית את כל האיברים בין left ו- right ללא כפילויות, לכן נרוץ בלולאה על הרשימה המקושרת השלישית, נספור את כמות האיברים ונחזיר אותה פחות אחד, שזה בדיוק אורך המסלול הקצר ביותר.  
ניתוח סיבוכיות זמן ריצה ומקום :



שאלה מספר 3 בסוף הקובץ !!



א. טענה זו איננה נכונה. קל לראות זאת עבור עץ שגובהו 5 :

h-2

h

ב. טענה זו איננה נכונה. דוגמא נגדית:

נמחק בעץ הבא קודם את 5 ואז את 1:

2

2

2

4

1

4

1

4

5

כעת, נמחק את 1 ואז את 5:

2

2

4

2

5

1

4

4

5

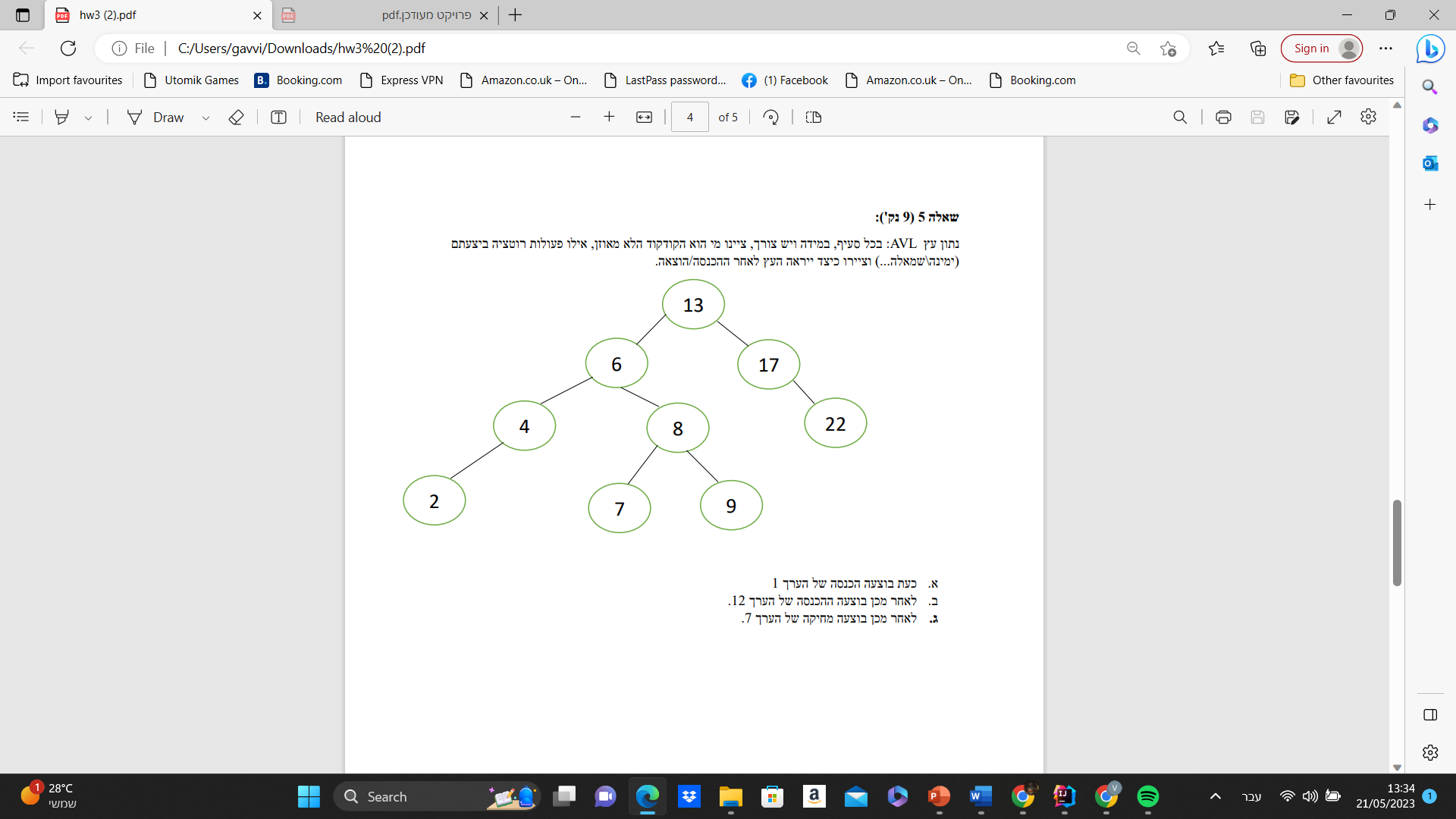
קל לראות שהעצים שהתקבלו אינם זהים.

ג. הטענה נכונה.

נראה זאת באינדוקציה לפי ההוכחה שראינו בהרצאה לחישוב מספר הצמתים עבור עץ AVL:

ד. טענה זו אינה נכונה, נסביר זאת:

אנו יודעים שכדי לבדוק כי עץ מקיים את תכונת האיזון ,יש לבדוק שכל אחד מצמתים בעץ מקיים את תכונת האיזון אשר נלמדה בכיתה. לכן, היות ובעץ יש n צמתים, לא אפשרי לבצע בדיקה זו בפחות מ-.



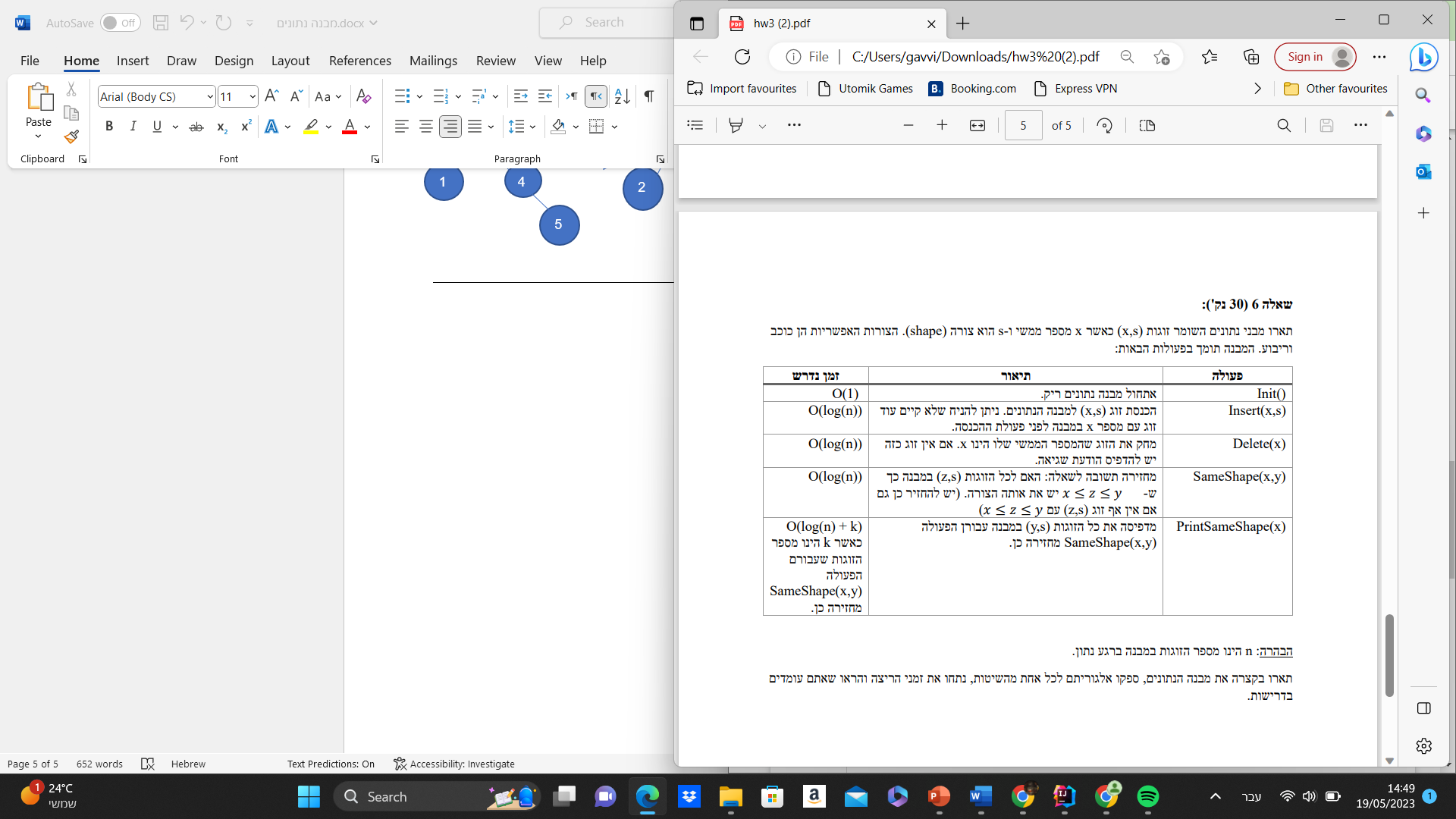
א.

לאחר ההכנסה של קודקוד עם הערך 1, הצומת שלא תהיה מאוזנת הינה הצומת שמכילה על הערך4. סוג חוסר האיזון הינו שמאלה שמאלה, לכן נבצע רוטציה על הצומת עם הערך 4 לקבלת העץ המאוזן הבא:

ב. לאחר ההכנסה של הערך 12, הצומת הראשונה שאינה מאוזנת הינה הצומת שמכילה את השורש 13. סוג חוסר האיזון הינו שמאלה ימינה. לכן נבצע רוטציה על צומת המכילה את הערך 6 לקבלת העץ הבא:

לבסוף, נבצע רוטציה על השורש 13 ימינה

ג. לאחר המחיקה של הצומת עם הערך 7, נוצר חוסר איזון על הצומת המכילה את הערך 6. סוג חוסר האיזון הינו שמאלה שמאלה, לכן נצבע רוטציה יחידה על הצומת המכילה את הערך 6 לקבלת העץ המאוזן הבא\\



תיאור מבנה הנתונים: לצורך הפתרון, נשתמש בשלוש עצי AVL כך :

* יהא עץ AVL בשם ג, העץ ישמור בתור מפתח את הערץ המספרי של כל זוג .

כמו כן, העץ ישמור שדא נוסף המציין את סוג "הצורה" אותו NODE בעץ מכיל (ריבוע או כוכב)

* יהא עץ AVL בשם , העץ ישמור בתור מפתח את הערך המספרי של כל זוג .

כמו כן, כל NODE בעץ יכיל מצביע לרשימה מקושרת דו-כיוונית אשר תכיל בתוכה את הערך של הNODE הנוכחי, וכל מי שגדול ממנו בעץ . (ראה איור בהמשך שממחיש)

* יהא עץ AVL בשם , העץ ישמור בתור מפתח את הערך המספרי של כל זוג .  
  כמו כן, כך NODE בעץ יכיל מצביע לרשימה מקושרת דו-כיוונית אשר תכיל בתוכה את הערך של הNODE הנוכחי, וכל מי שגדול ממנו בעץ . (ראה איור בהמשך שממחיש)

*דגשים והנחות:*

* בין כל NODE בעץ , יהיה מצביע לNODE המקביל לו בעץ AVL השני המתאים לו לפי צורתו( או ).
* בין כל NODE בעץ יהיה מצביע לכל איבר בכל רשימה מקושרת של הNODES האחרים (ז"א אם NODE ספציפי בעץ, נגיד המקסימלי, יכיל מצביע לכל NODE אחר לרשימה המקושרת שאותו NODE מכיל, ההצבעה תהיה לעצמו באותה רשימה מקושרת כי היא בהכרח תכיל אותו (כי הוא המקסימלי)). המטרה של כך היא ברגע שנרצה למחוק איבר מהעץ AVL וניגש לעץ לNODE הספציפי שאותו נרצה למחוק, יתכן שNODES אחרים יכילו אותו במידה והם קטנים ממנו, לכן ע"י המצביע נוכל ב(1)O זמן לגשת לכל ההצבעות ולמחוק אותו (ראה איור בהמשך שממחיש)
* אותה הנחה כמו לעיל לעץ

*דגשים להוספה ומחיקה בעץ:*

* בעת הוספה לעץ AVL, כשנרצה להוסיף את הזוג לעץ המתאים לו לפי צורתו, נרצה שברשימה המקושרת שאותו NODE חדש יהיה את עצמו בתוספת לרשימה מקושרת של הקודם לו ( כיוון שהוא בהכרח יהיה קטן מכל מי שהקודם קטן לו), לכן, בהוספה, ראש הרשימה המקושרת יהיה הערך של הNODE הנוכחי ומצביע נוסיף לראש הרשימה המקושרת של הNODE הקודם לו. (ראה איור בהמשך שממחיש)
* בעת המחיקה בעץ AVL, כשנרצה למחוק את הזוג מהעץ המתאים לו לפי צורתו, ניגש לNODE המתאים בעץ המתאים, נמחק אותו, וכעת כל רשימה מקושרת אחרת בעץ הזה המכילה אותו כרגע תאבד את ההצבעה שלה אליו ותצביע על NULL (אשר יתן לנו אינדיקציה שלא קיים בNODE הספציפי ברשימה המקושרת הספציפית ערך כל שהו, נשתמש בזה בסעיף האחרון) – ראה איור הממחיש זאת בהמשך

*–*

-

18

15

18

8

18

15

**אותו הרעיון גם עם העץ**

*מימוש המתודות:*

* - נאתחל שלוש עצי AVL ריקים כפי שתואר לעיל – (1)O במקרה הגרוע
* - נכניס את הזוג בעץ פעולה אשר תיקח לנו במקרה הגרוע כפי שנלמד בכיתה לגבי הכנסה לעץ AVL.

לאחר מיכן, ניגש לעץ AVL המתאים לאותו זוג לפי הצורה של הזוג ונכניס את הערך המספרי לעץ המתאים עם שינוי קטן : בזמן שנרד בענפים בעץ, בכל פעם שנרד ימינה, ז"א לערך גדול יותר, נוסיף לרשימה המקושרת של הערך הנוכחי לפני שירדנו בעץ (במידה וקיים) את הערך של הNODE החדש, בדרך זו כל ה"באים" לNODE שהוספנו לו כרגע את הערך החדש לרשימה המקושרת שלו יכלו גם כן את הערך הזה מהסיבה שכל רשימה מקושרת מכילה קודם את הערך של הNODE הנוכחי ואז מצביע לרשימה המקושרת של הקודם לו (אם קיים). שוב הכנסה לעץ AVL נוספת תיקח לנו זמן במקרה הגרוע.

לאחר מיכן, ניצור מצביע מאותו NODE חדש לרשימה מקושרת חדשה אשר תכיל בראשה את הערך של הNODE הנוכחי ומצביע לראש הרשימה המקושרת של הקודם לו (אם קיים כזה) כאשר אתחול רשימה מקושרת חדשה עם ערך יחיד ויצירת ההצבעה מידה ויש לקודם של הNODE החדש תיקח לנו (1)O זמן במקרה הגרוע.

לבסוף, ניצור הצבעות הדדיות בין כל NODE בעץ לNODE המקביל לו (החדש) בעץ AVL השני בהתאם לצורתו או ).

סה"כ זמן הריצה של פעולת ההכנסה תהא:

* - תחילה נמחק את הNODE המתאים בעץ AVL לפי הערך שנשלח למתודה עם שינוי קטן:

לפני המחיקה עצמה, ניגש לעץ המתאים לNODE לפי צורתו, גישה שתיקח לנו ע"י ההצבעה שתוארה לעיל זמן של (1)O, ונמחק את הNODE המתאים. לאחר המחיקה, כל רשימה מקושרת בעץ המתאים שהכילה הצבעה לערך הנמחק (במידה והיית כזו), תכיל כעת NODE עם ערך NULL שמציין שאין שם ערך מספרי.

לבסוף נמחק את הNODE מהעץ .סה"כ מחיקה משתי עצי AVL עם שימוש במצביעים לוקחת לנו זמן של במקרה הגרוע כנדרש.

* - בעץ אם ריק נחזיר אמת ישר. אחרת, נחפש את x בעץ בזמן החיפוש נבצע את הבדיקה הבאה: ומידה וערך של הNODE שעתה ירדנו אליו קטן מהערך של הקודם לו ולא שווה לy- נבדוק האם הצורה של אותו NODE קודם שווה לצורה של הNODE הנוכחי, במידה ולא נחזיר ישר FALSE, אחרת נמשיך לחפש את x. אם הגענו לנקודה זו אני יודעים בוודאות שהחלק זה שבדקנו תקין וכל הNODES שעומדים בתנאי הקודם אכן בעלי אותה צורה, נשמור את השם שלו הצורה הזו בערך חדש שנקרא לו . – בדיקה ראשונה הכולל חיפוש של -X מהשורש הינה במקרה הגרוע.

לאחר מיכן, נחפש את y- (אם לא קיים אז את הקודם שלו, ז"א הראשון שקטן ממנו בעץ) החל מהשורש עם הבדיקה הבאה : בכל פעם שאנחנו יורדים לערך שגדול מהערך של הNODE הנוכחי, נבדוק האם הצורה של הNODE הנוכחי שווה לצורה של הNODE הבא אליו אנו יורדים ושווה לצורה , במידה ולא, נחזיר ישר FALSE. לבסוף נחזיר TRUE.

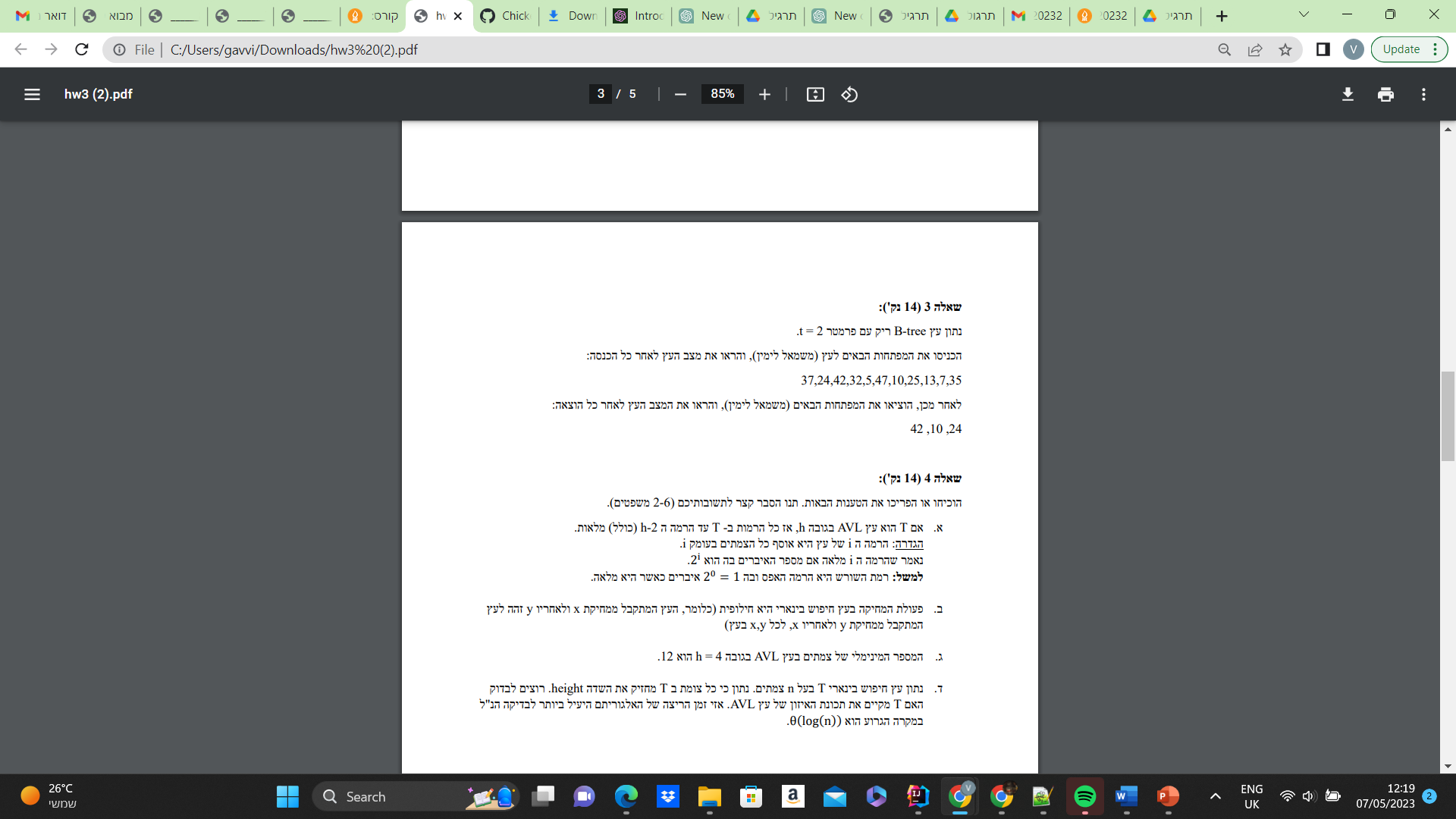
* - נחפש את x בעץ חיפוש אשר יקח במקרה הגרוע, לאחר מיכן ניגש לNODE המתאים בעץ השני לפי צורתו המתאימה של הNODE הנוכחי וע"י שימוש במצביע בינו לבין העץ המתאים, גישה אשר תיקח לנו (1)O במקרה הגרוע.

כזכור, לפי אין שבנינו את מבני הנתונים, אותו NODE בעץ AVL המתאים לו לפי צורתו מכיל מצביע לרשימה מקושרת שמכילה את כל האיברים הכוללים את עצמו, ואת כל מי שגדול ממנו ובעל אותה צורה כמוהו, לכן פשוט לרוץ בלולאה על כל הרשימה המקושרת ונדפיס את כל אותם האיברים.

ההנחה היא שיש K איברים כאלו, ז"א גודל הרשימה המקושרת הינה K, ולכן ריצה בלולאה והדפסה תיקח לנו (K)O.

(נשים לב, שבגלל איך שהגדרנו את העץ ובכלל כל השינויים שאנו עושים בו בהכנסה/מחיקה, לא היה צורך להשתמש במתודה כיוון שכל עץ מהסוג השני מכיל רק את האיברים שבאותה צורה, וכן כל רשימה מקושרת מכילה את כל האיברים החל מאותו הNODE ועד לNODE הגדול ביותר בעץ, ז"א y)

סה"כ זמן הריצה יהיה:



47 42

32 24 5

37

24

42

37

37

42 37 24

4ג4

37 24

4ג4

כשנרצה כעת להכניס, השורש מלא לכן נפצל:

47 42

32

5

37 24

47 42

32 25

13 10 5

37 24

37 24 10

13

5

47 42

32 25

37 24 10

7 5

13

47 42

32 25

7 5

37

10

13

35 32 25

47 42

24

פיצוי השורש + הכנסה של 32 לעלה  
תשובה סופית להכנסות:

**חלק ב: מחיקות**

24

47

כעת נוכל למחוק את 42 מהעלה

7 5

13

35 32 25

37 24 10

47 42

35 32 25

13

7 5

37 24 10

הצומת שמכיל את 37 הינו צומת מלאה, והאח 10 לא יכול להלוות לו, לכן נתאחד עם השורש

35 32 25

7 5

47 42

13

10

37

47

נחליף את 10 ב7 ונמשיך למחוק את 7 רקורסיבית

7 5

13

35 32 25

37 24 7

47

נחליף את 10 ב7 ונמשיך למחוק את 7 רקורסיבית

7 5

13

35 32 25

37 24 7

37 24 7

47

נחליף את 10 ב7 ונמשיך למחוק את 7 רקורסיבית

7 5

13

35 32 25

37 24 7

נרצה למחוק את 24, לכן נוריד אותו לעלה ע"י החלפתו עם 25 והמשך מחיקה 25 רקורסיבית

47

13

35 32 25

5

47

נחליף את 10 ב7 ונמשיך למחוק את 7 רקורסיבית

7 5

13

37 24 7

37 25 7

47

13

35 32 25

5

כעת נוכל פשוט למחוק את 25 מהעלה לקבלת התשובה הסופית למחיקות:

37 25 7

47

13

35 32

5