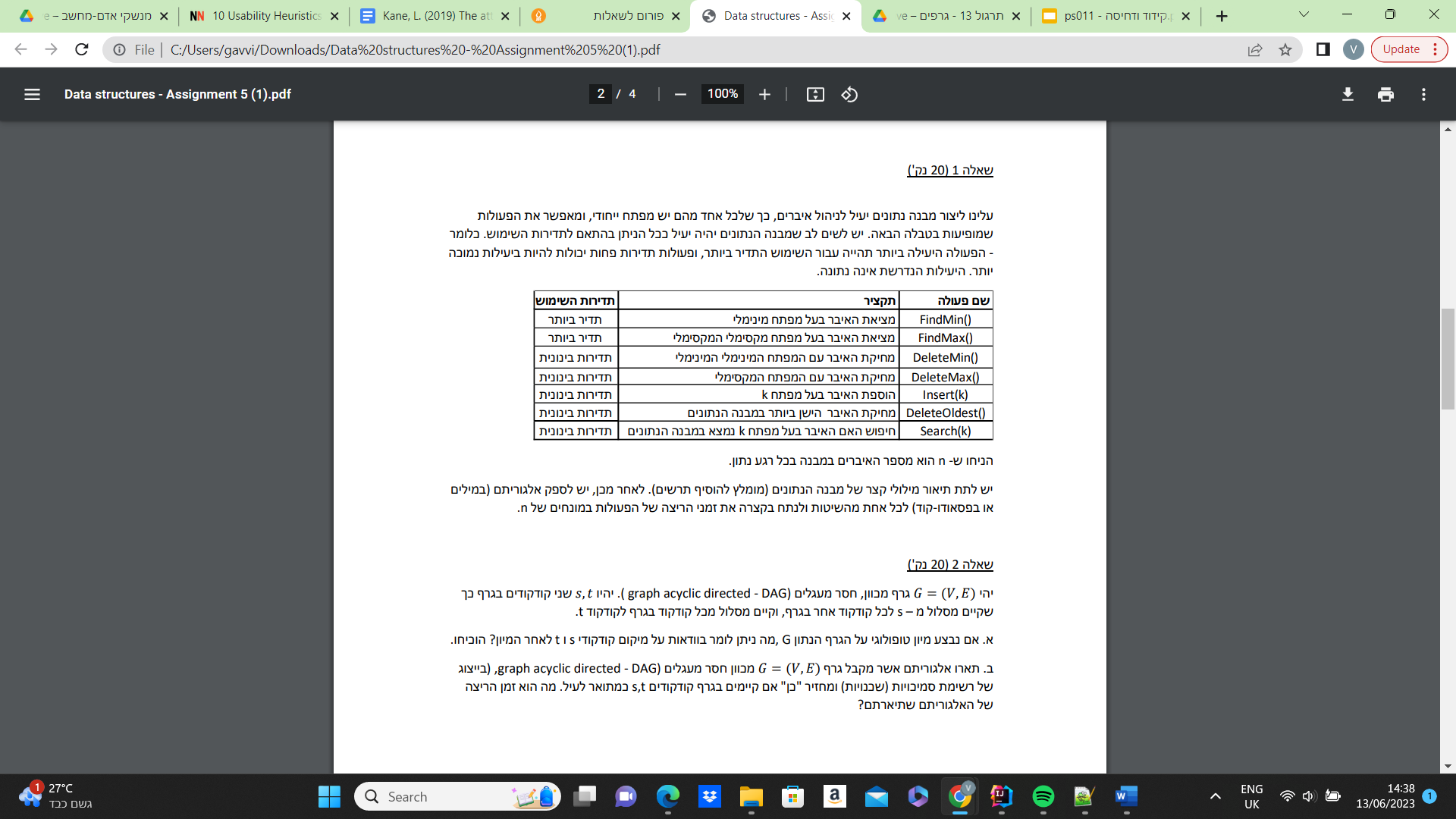
עבודה במבני נתונים מ'ס 5 – מגישים : ויקטור גברילנקו 209406255 ותומר שולמן 208997551



תיאור מבנה הנתונים:

לצורך השאלה נשתמש בשתי ערימות, אחת ערמת מקסימום ואחת ערמת מינימום, בנוסף גם עץ AVL.

* יהא ערמת מקסימום - , כאשר המפתח הינו הערך אותו המשתמש רוצה לשמור ("מפתח ייחודי")
* יהא ערמת מינימום - ,כאשר המפתח הינו הערך אותו המשתמש רוצה לשמור ("מפתח ייחודי")
* יהא עץ AVL - אשר יחזיר את המפתחות שהמשתמש יחליט שיהיו במבנה הנתונים
* רשימה מקושרת דו- כיוונית עם מצביע להתחלה ולסוף שתייצג תור קדימויות - Q

חשוב לציין שבין כל הערכים בין שלושת מבני הנתונים קיימים הצבעות הדדיות, כך שנוכל לעבור בין מבני נתונים אחד לאחר מאותו ערך בשני מבני נתונים שונים בזמן קבוע של (1)O.

ציור מבנה הנתונים:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 | 4 | 10 | 12 | 15 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 15 | 12 | 10 | 4 | 2 |

***:***

H

T

**Q:**

- ניגש לערמת המינימום שלנו, ובזמן קבוע של (1)O כפי שנלמד בהרצאה, נחזיר את הערך במיקום הראשון במערך שמייצג את הערימה הזו, שכאמור הוא בהכרח יהיה האיבר המינימלי.

15

2

12

10

4

זמן ריצה : (1)O

- ניגש לערמת המקסימום שלנו, ובזמן קבוע של (1)O כפי שנלמד בהרצאה, נחזיר את הערך במיקום הראשון במערך שמייצג את הערימה הזו, שכאמור הוא בהכרח יהיה האיבר המקסימלי.

זמן ריצה : (1)O

- בזמן קבוע של (1)O ניגש לאיבר המינימלי שנמצא בראש המערך שמייצג את ערמת המינימום, ע"י ההצבעות ההדדיות שהגדרנו ניגש קודם לעץ AVL שלנו ונמחק את האיבר בעץ שהתוכן שלו תואם להצבעה שהגדרנו. מחיקה מעץ AVL כפי שנלמד בהרצאה הינו במקרה הגרוע.

לאחר מיכן, ע"י ההצבעה ניגש לאיבר זה בQ ונמחק אותו, כיוון שיש הצבעות דו-כיווניות בין אברי Q המחיקה תקח זמן של (1)O. (להגיד לקודם להצביע להבא של מי שאנו רוצים למחוק אם האיבר נמצא באמצע של Q, אם הוא ראש הQ אז פשוט להזיז את ההצבעה להבא, ואם הוא הזנב, להזיז את ההצבעה אחד אחורה)

לאחר מיכן, ניגש לערמת המקסימום ע"י אותה הצבעה מערמת המינימום, ונמחק את הערך התואם להצבעה.

לבסוף נמחק את הערך המינימלי מערמת המינימום. שתי מחיקות מהערמות

(הערה: קודם מחקנו את הערך מהעץ AVL ולאחר מיכן מהערמת מקסימום ורק לבסוף מערמת המינימום כדי לא לאבד את ההצבעות שלנו מהערמת מינימום לשאר מבני הנתונים)

זמן ריצה סה"כ :

***-*** *אותו האופן בדיוק כמו מחיקה של המינימלי, רק שניגש מהאיבר הראשון בערמת המקסימום ע"י המצביעים לערמת המינימום אל Q ואל עץ AVL ונמחק שם קודם, לבסוף נמחק מערמת המקסימום.*

זמן ריצה סה"כ :

*- נוסיף את האיבר K לכל אחד משלושת מבני הנתונים בזמן של סה"כ (כאמור הוספה לעץ AVL כולל איזונו הינה וגם הוספה לערמה כולל תיקונה הינה ). כעת, נוסיף את K להיות בראש הרשימה המקושרת הדו-כיוונית שלנו Q, זמן קבוע של (1)O. לבסוף ניצור את ההצבעות ההדדיות בין שלושת מבני הנתונים בזמן קבוע של (1)O.*

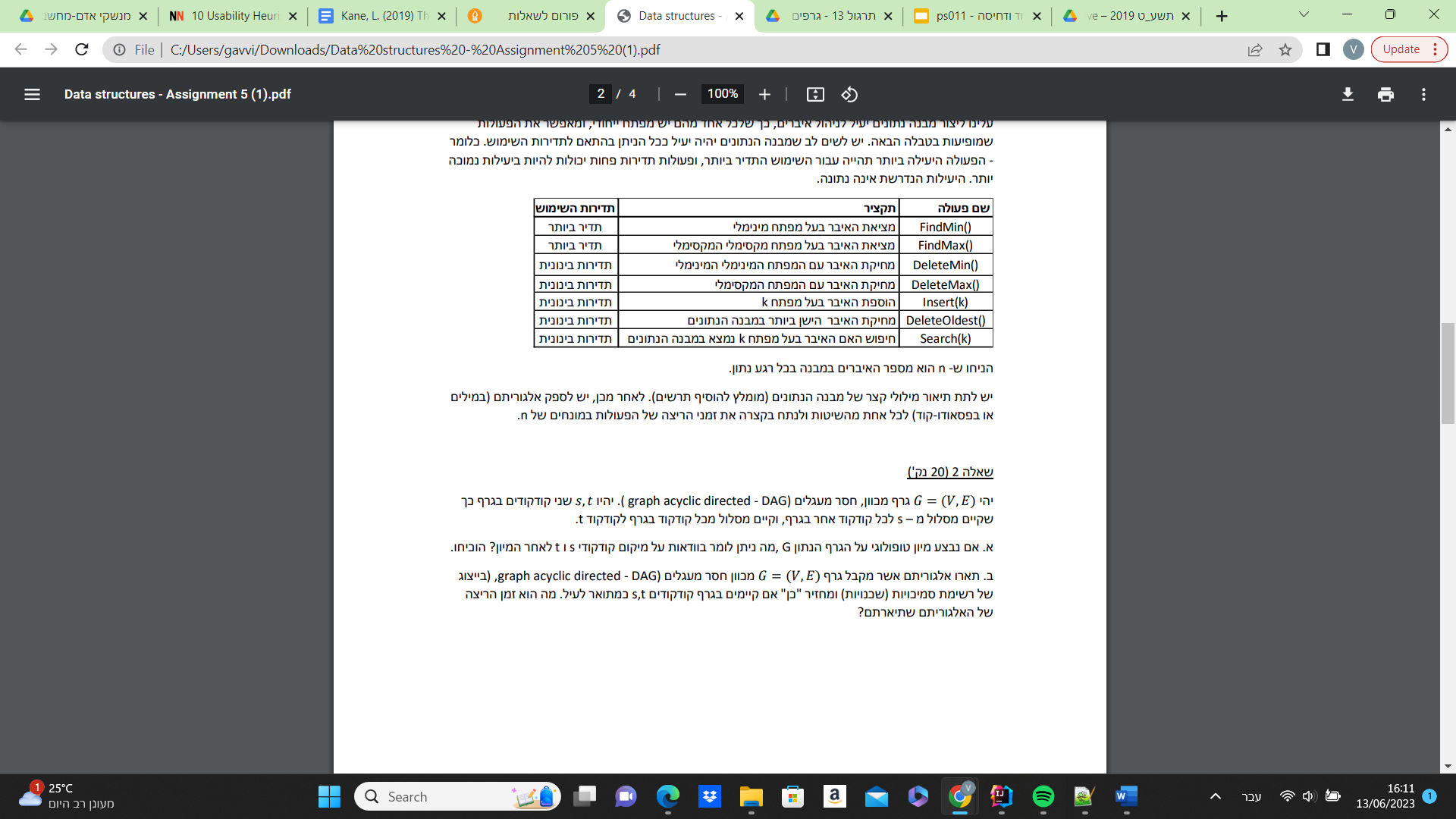
זמן ריצה סה"כ :

- בזמן קבוע של (1)O ניגש לאיבר האחרון בQ (כאמור יש לנו הצבעה אליו לפי איך שהגדרנו את מבני הנתונים), לאחר מיכן ניגש לכל שאר מבני הנתונים ע"י ההצבעות ההדדיות ונמחק את אותו איבר מהם, הראנו בסעיפים הקודמים של המחיקות ממבנים אלו יקחו זמן של *. לבסוף נמחק את האיבר מ-Q ע"י הזזת ההצבעה מהאיבר האחרון לקודם לו. (אם הגודל 1 אז פשוט יצביע על NULL).*

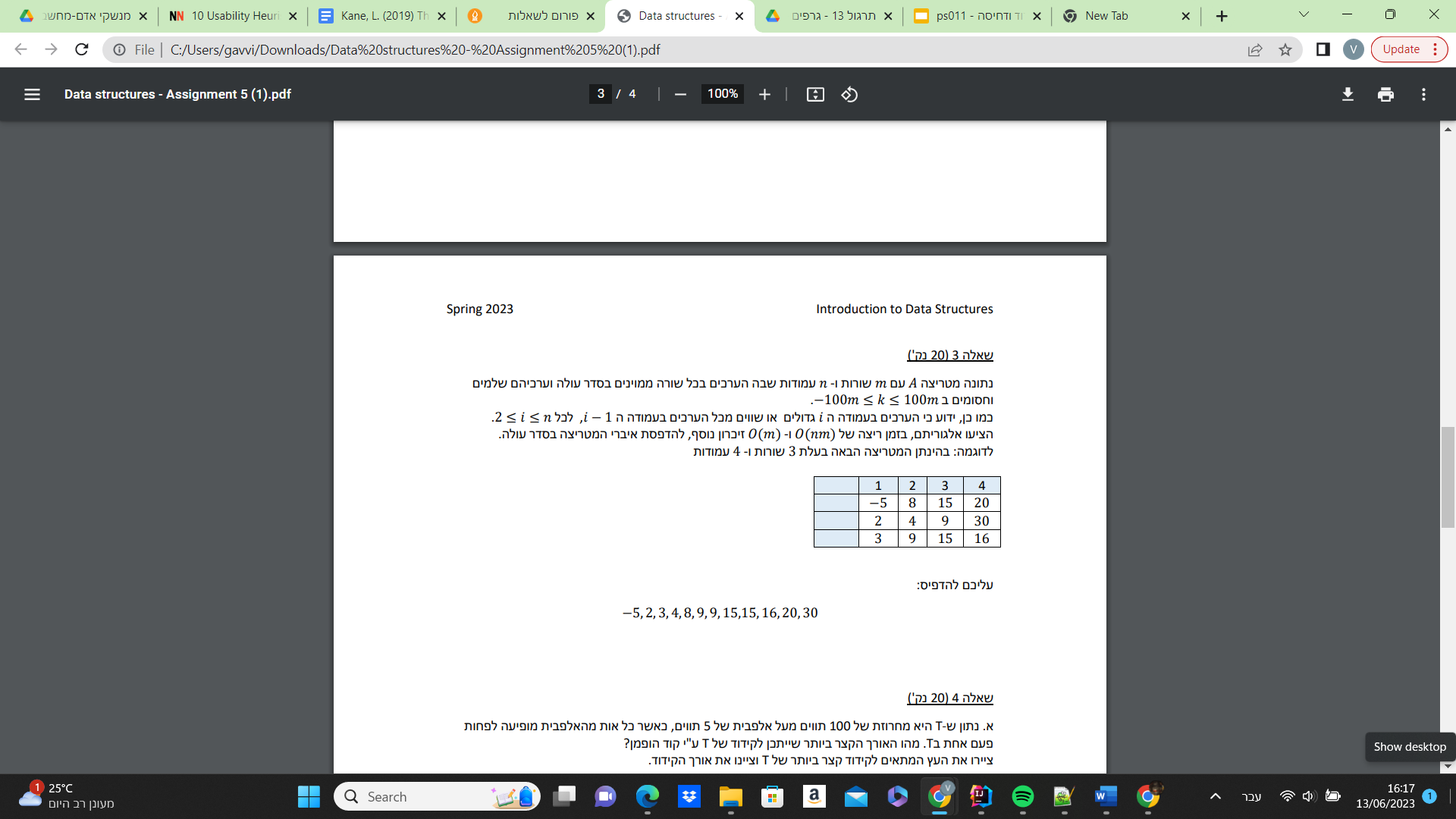
זמן ריצה סה"כ :

- ניגש לעץ AVL שלנו, ופשוט נחפש את K חיפוש רגיל כפי שנלמד בהרצאות. כאמור, חיפוש איבר בעץ AVL לוקח זמן , נחזיר אמת אם מצאנו ושקר אם לא.

זמן ריצה סה"כ :



פתרון השאלה בסוף הקובץ !!



תחילה נעבור על כל עמודות המטריצה ונמספר אותן מ 0 ועד n, תוך כדי המעבר נוסיף לכל איבר 100m כך שכל האיברים יהיו בטווח . כעת, נעבור עמודה עמודה וכל עמודה נמיין לפי *מיון מניה*, כך שמיון כל עמודה יקח זמן של:

כיוון שבכל עמודה יש לנו m איברים (כמספר השורות). כך נעבור על כל המטריצה ונמיין עמודה עמודה.

סה"כ נצטרך למיין n עמודות. לכן סה"כ זמן הריצה של מיון כל המטריצה הינו:

*לבסוף, נעבור כל כל המטריצה פעם אחרונה ומכל ערך נוריד 100m כדי להחזיר את הערכים של המטריצה להיות כפי שהיו לפני המיון (כיוון שבהתחלה הוספנו* 100m לכולם)

כמו כן נציין שהשתמשנו כל פעם בזכרון של (m)O כיוון שבכל פעם מיינו עמודה אחת שמכילה כאמור m איברים. לכן סה"כ זמן ריצה הינו

A screenshot of a computer

Description automatically generated

*סעיף א:*

*סעיף ב:*

טענה זו איננה נכונה. קל להפריך אותה ע"י הדוגמא הנגדית הבאה:

יהיו התווים הבאים : , נתבונן בעץ האופמן המתאים:

קל לראות שאורך מילת הקוד של b ארוכה ממש מאורך מילת הקוד של a, אך השכיחות שלהם שווה זו לזו.

*סעיף ג:*

טענה זו גם אינה נכונה. דוגמא נגדית לכך : . נתבונן בעץ האופמן הבא:

בעץ הנ"ל לשני התווים יש אותו אורך של מילת קוד, אך שהשכיחות שלהן שונה !

שאלה 5:

נניח ומערך הקלט מתחיל מאינדקס 0.

א. מיון מנייה (Counting Sort):

לגבי מיון מניה, אנו צרכים שערכי מערך הקלט יהיו חסומים בטווח מסוים מהצורה של [1…k] . אצלנו, עבור n רכיבים קיימים לכל היותר2n אפשרויות לקידוד ( לדוג : עבור n=3 נקבל את האפשרויות הבאות : {000, 100,110,111,101,010,001,011} שזה סה"כ 8 אפשרויות לקידוד, ז"א 23=8). לכן, ערכי מערך הקלט יהיו חסומים בטווח שבין [0….2n].

המיון יתבצע באופן הבא: קודם כל, נעבור על כל המערך ונמיר את הקידודים לערכים דצימליים. לאחר מיכן, ניצור את מערך העזר שלנו C אשר גודלו יהיה 2n. לאחר מיכן נריץ את אלגוריתם מיון המניה (כפי שנלמד בהרצאה) על מערך הקלט שלנו עם k= 2n . הפלט שנקבל יהיה מערך B חדש עם הערכי הקידודים באופן דצימלי ממוינים, לכן נעבור על מערך הפלט שלנו, ונמיר בחזרה את הערכים למספריים בינריים, ונחזיר את המערך הסופי הממוין.

ניתוח סיבוכיות זמן ריצה ומקום :

* סיבוכיות מקום : מיון מניה לוקח סיבוכיות מקום של במקרה הגרוע כאשר אצלנו k= 2n  לכן:
* סיבוכיות זמן ריצה : מיון מניה לוקח סיבוכיות מקום של במקרה הגרוע כאשר אצלנו k= 2n  לכן:

*כפי שניתן לראות, מיון זה איננו יעיל מכיוון שגם סיבוכיות זמן הריצה וגם סיבוכיות המקום הם אקספוננט.*

ב. מיון בסיס (Radix Sort):

עבור מיון בסיס, הבסיס שלנו הוא 2, ז"א b = {0,1}, אולם כמו הספרות שלנו לכל היותר n, ז"א d = n.

המיון יתבצע באופן הבא:

נעבור בצורה סדורה על כל ספרות הקידודים מימין לשמאל, ונכניס את כל הספרות למערך. את המערך נמיין במיון יציב כלשהו ( לדוג' merge\_sort אשר לוקח סיבוכיות מקום של O(n)).

לכן עבור מיון זה סיבוכיות זמן הריצה הוא :

ועבור סיבוכיות זמן המקום זה תלוי בסוג המיון היציב שנבחר לטובת מיון כל עמודה של ספרות.

עבור מיון מניה נקבל כמו בסעיף מעלה, , ועבור מיון כדוגמת merge\_sort נקבל במקרה הגרוע O(n)

*ג. מיון מבוסס השוואה במקרה הגרוע:*

*עבור מיון זה, הכי נוח יהיה לבחור את* merge\_sort.

המיון יתבצע באופן הבא:

נעבור בצורה סדורה על כל אברי מערך הקלט, ונעביר אותם לבסיס דצימלי. לאחר מיכן, נפעיל את אלגוריתם merge\_sort (כפי שנלמד בהרצאה), כך שבסיומו נקבל מערך חדש ממוין, כך שכל מה שישאר לעשות זה לעבור שוב על המערך בצורה סדורה ולהמיר את אבריו לבסיס בינרי מחדש. לבסוף נקבל מערך פלט בעל ערכים בינריים ממוין כפי שנדרש.

ניתוח סיבוכיות מקום וזמן ריצה:

* *עבור סיבוכיות מקום אני נדרש להשתמש במערך נוסף שגודלו כגודל הקלט, ז"א סה"כ* O(n) במקרה הגרוע.
* עבור סיבוכיות זמן ריצה, כפי שידוע merge\_sort ימיין בזמן ריצה של במקרה הגרוע.

**לסיכום,** לאחר ניתוח שלושת המיונים, נראה שלא ניתן להוריד את סיבוכיות זמן הריצה לזמן ריצה לינארי ( במידה וכמו הספרות במיון הבסיס,d , הייתה קבוע כל שהוא, היינו יכולים להגיע לזמן ריצה לינארי) , לכן, המסקנה היא שעדיף למדענים למיין את הרכיבים ע"י מיון מבוסס השוואות במקרה

סעיף ב':

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Coding | Cn | … | Ck+1 | Ck | … | C4 | C3 | C2 | C1 |  |
|  | 1..10..1000 | 1 | … | 0 | 1 | … | 0 | 0 | 0 | 1 | C1 |
|  | 0..01..0010 | 0 | … | 1 | 0 | … | 0 | 1 | 0 | 0 | C2 |
|  | 0..10…1100 | 0 | … | 0 | 1 | … | 0 | 0 | 1 | 1 | C3 |
|  | 0..10..1010 | 0 | … | 0 | 1 | … | 0 | 1 | 0 | 1 | C4 |
|  | … | … | … | … | … | … | … | … | … | … | … |
|  | 0..00..1110 | 0 | … | 0 | 0 | … | 0 | 1 | 1 | 1 | Ck |

עבור :

א. מיון מנייה (Counting Sort):

עבור מיון מנייה זמן הריצה יהיה:

O(n+k)={k=O(n)}=O(2n)=O(n)

ב מיון בסיס (Radix Sort):

עבור מיון בסיס זמן הריצה יהיה:

O(d(n+k))={ k=O(n), d=n}=O(n(2n)=O(n^2)

*ג. מיון מבוסס השוואה במקרה הגרוע:*

גם אם( k=O(n זמן הריצה של מיון מיזוג נשאר . הסיבה לכך היא שהאלגוריתם עדיין צריך לעבור את אותו תהליך של חלוקה, מיון ומיזוג האלמנטים. זמן הריצה אינו תלוי ב k אלא במספר האיברים שיש למיין.

א. מיון מנייה (Counting Sort):

עבור מיון מנייה זמן הריצה יהיה:

O(n+k)={k=O(log(n))}=O(n+log(n))=O(n)

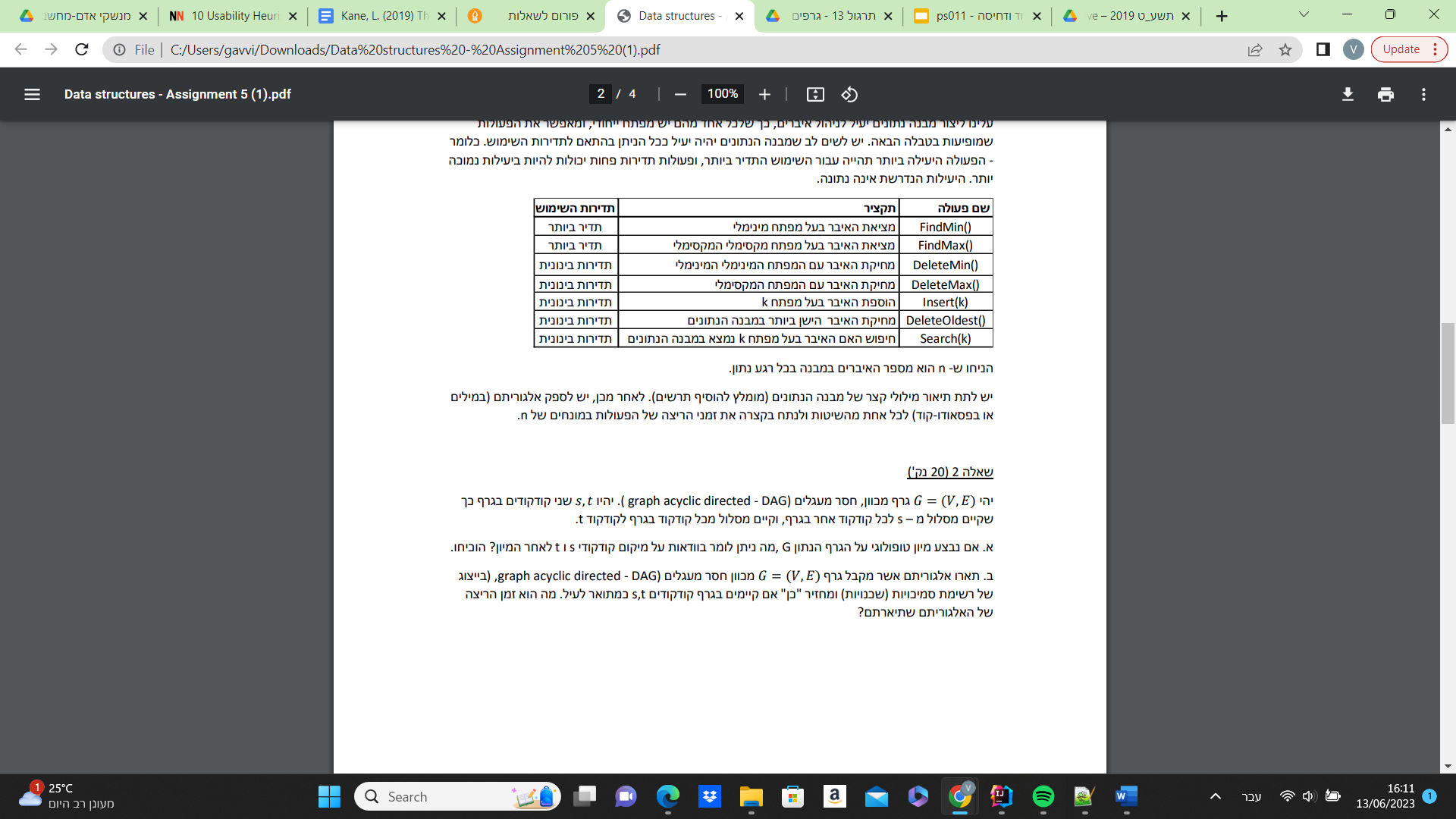
ב. מיון בסיס (Radix Sort):

עבור מיון בסיס זמן הריצה יהיה:

O(d(n+k))={d=n, k=O(log(n))}=O(n(n+log(n))=O(n^2)

*ג. מיון מבוסס השוואה במקרה הגרוע:*

גם אם ( k=O(logn זמן הריצה נשאר .



*סעיף א:*

לפי החומר שנלמד בהרצאה אנו יודעים שמיון טופולוגי הינו מיון כך שאם בגרף G קיים

אז u מופיע איפה שהוא לפני v ברשימה שמייצגת מיון זה. נתון לנו שקיים מסלול מכל קודקוד בגרף אל עבר קודקוד t, לכן קודקוד זה יהיה **אחרון** ברשימה שמייצגת את המיון הזה, כמו כן נציין שאין ממנו אף צלע לאף קודקוד אחר, אחרת היה נוצר לנו מעגל בגרף (כיוון שיש מסלול אליו מכל קודקוד, ומספיק שהיה מסלול ממנו לקודקוד אחר היה נוצר מעגל) וזה בסתירה לנתון שמדובר על DAG.

כמו כן, ידוע לנו שקיים מסלול מקודקוד s אל עבר כל הקודקודים בגרף, לכן ברשימה שמייצגת את המיון הטופולוגי הוא יהיה בהכרח במקום **הראשון,** ז"א בתחילת הרשימה. (נסיים עם s רק לאחר שנסיים עם כל שאר הקודקודים בגרף, כיוון שיש מסלול אליהם ממנו)

*סעיף ב:*

תחילה נבצע סריקת DFS אחת מקודקוד שרירותי בגרף G. לאחר שנסיים עם הסריקה, נעבור על כל קודקוד בגרף, ונחפש את הקודקוד שזמן הסיום שלו מקסימלי מבין שאר הקודקודים בגרף, נסמן קודקוד זה ב-s.

כעת נבצע סריקת DFS נוספת, אך הפעם לא מקודקוד שרירותי אלה מקודקוד s. בסיום הסריקה, נרוץ על כל הקודקודים בגרף ונבדוק האם קיים קודקוד שלא ביקרו בו, אם קיים קודקוד כזה נחזיר שקר כיוון שלא קיים מסלול מ s לשאר הקודקודים בגרף G. אחרת קיים מסלול לכולם.

זמן ריצה הינו שתי סריקות DFS ושתי מעברים על כל הקודקודים בגרף G:

כדי לבדוק האם קיים קודקוד t כמתואר בתנאי השאלה. ניצור גרף חדש בדרך הבאה:

נרוץ על כל הקודקודים בגרף ולכל נהפוך את כיוון החיצים ממנו אל השכנים המיידים שלו בדרך הבא: ברשימה הסמיכויות של גרף G, כל קודקוד כעת יופיע ברשימת השכנים של אלו שהיו השכנים שלו לפי השינוי (פה נכנס החלק של "היפוך" החיצים בגרף), חלק זה ייקח לנו כיוון שאני עוברים על כל רשימת השכנויות של כל הקודקודים בגרף. כעת נוכל לבצע על הגרף החדש 'G את האלגוריתם שביצענו לחיפוש s, כיוון שאם קיים קודקוד t כזה בגרף 'G אז בהכרח יהיה חץ (מסלול) ממנו לכל קודקוד בגרף.

אם לא, נחזיר שקר.

לבסוף נחזיר אמת כי אם הגענו לשלב זה אכן מצאנו שני קודקודים s,t אשר עומדים בתנאי השאלה.

זמן ריצה לחלק זה יהיה גם , וכן סה"כ זמן הריצה של כל האלגוריתם יהיה: