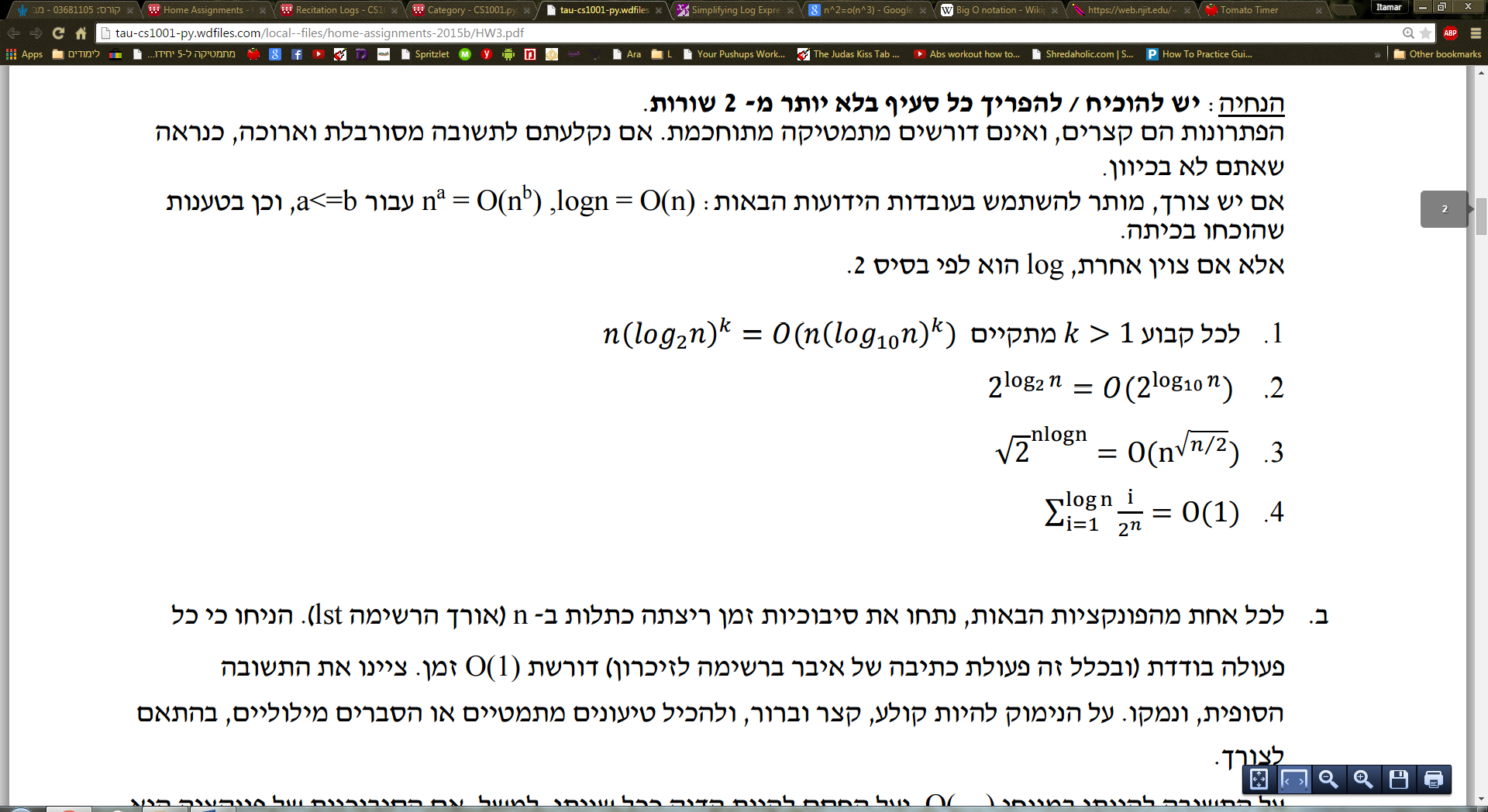
**מבוא מורחב למעדי המחשב - תרגיל בית 3**

1. א. (1) נכון. . כעת נבחר קבוע c= ונקבל =2 ולכן זה נכון.

(2) לא נכון. נוכיח בשלילה: . נקבל:

=> . זוהי סתירה כיוון שהביטוי שואף לאינסוף כש-n שואף לאינסוף.

(3) לא נכון. נוכיח בשלילה: => => <c = . שוב סתירה מאותה סיבה.

(4) נכון. = < c. בביטוי זה המכנה שואף לאינסוף מהר יותר מהמונה, לכן הביטוי שואף ל-0 באינסוף וניתן לחסום אותו על ידי קבוע c.

ב. n=len(lst) היא זניחה (O(1)) לכן לא נתייחס אליה בניתוח הסיבוכיות.

1. O(N^2). בכל סבב של הלולאה נוספים i (שרץ מ-0 עד N-1) איברים. סך הכל: הוספות איברים. (הביטוי חסום על ידי n^2 כיוון שזוהי החזקה הגבוהה ביותר בפולינום). כל הוספה היא O(1). לכן הסיבוכיות היא O(N^2)

2. O(n\*(2^n)). בכלל סבב נוספים n\*(2^i) איברים כאשר i רץ בין 0 ל-n-1. מסכום סדרה הנדסית, נקבל n\*(2^n)-n הוספות של איברים. כלומר O(n\*(2^n)) כיוון שזהו האיבר הדומיננטי בביטוי.

3. O(n\*(2^n)). בכל סבב נוצרת רשימה חדשה בגודל n\*(2^(i+1)) כאשר i רץ מ-0 עד n-1. מסכום סדרה הנדסית, נקבל n\*(2^(n+1))-2n<c\*(n\*(2^n)) פעולות, ואם נבודד את c נקבל ביטוי ששואף באינסוף לקבוע. כלומר O(n\*(2^n)).

4. O(n). בכל סבב נוספים לרשימה 10,000 איברים בדיוק. ישנם n סבבים. נקבל 10,000n הוספות של איברים, 10,000n<c\*n אם נבחר c>10,000. כלומר סיבוכיות של O(n).

ג. בקטע הקוד נוסף איבר חדש לרשימה בכל סבב של הלולאה. הלולאה רצה על כל איבר של הרשימה ומכיוון שבכל סבב נוסף איבר חדש היא לא תסתיים לעולם.

1. א. הפונקציה מחזירה None כי היא חיובית עבור -10 ו-10.

ב. 0.823974609375-

ג. x= 12345678901 זהו שורש של הפונקציה, לכן a ו-b מתכנסים סביבו. אולם כאשר נגדיל או נקטין את x ולו ב- 10^(-6) נקבל ערכים גדולים ביותר בסדרי גודל של 10^35.  
אחרי כמה סבבים התכנית מחשבת את הפונקציה עבור 12345678901.0 שמיוצג כ-float, כלומר בזיכרון של פייטון ישנה סטייה קלה מהערך המוצג (בגלל floating point). בשל התנהגות הפונקציה, מתקבל מספר מסדר גודל גבוה ביותר. אז, בגלל רמת הדיוק המוגבלת שנובעת מ-floating point, התכנית שוב מגיעה לאותו מספר על ידי החישוב: ((12345678901.0+12345678901.000002)/2=12345678901.0), ואף פעם לא מגיעה לקירוב טוב, לכן נוצרת לולאה אינסופית.

1. א. **O((m\*n)^2).** על מנת לחשב את המקרה הגרוע נניח כי בכל הרשימות n איברים, כלומר מקסימום איברים סה"כ.  
   לבנות את all מצריך m\*n פעולות, כלומר סיבוכיות של O(m\*n).

נתעלם מפעולות זניחות.  
min כל פעם עוברת על כל הרשימה שקטנה ב-1 בכל סבב. כלומר m\*n-i פעולות כל פעם כאשר i רץ מ-0 עד m\*n-1. נקבל בערך ((m\*n)^2)/2 פעולות.

O(m\*n) זניח לעומת O((m\*n)^2) לכן נקבל סה"כ: O((m\*n)^2)

ד. הסיבוכיות של merge במקרה הכי גרוע היא O(n1+n2) כאשר אלו הם ארכי שתי הרשימות. במקרה הכי גרוע הלולאה רצה n1+n2 פעמים ומבצעת בכל סבב פעולות ב-O(1), בנוסף להכנת הרשימה בהתחלה שלה סיבוכיות של O(n1+n2). ביחד נקבל O(n1+n2).

נניח כי כל הרשימות באורך n. בכל סבב הסיבוכיות היא O(i\*n) כאשר i רץ בין 2 ל-m. סה"כ לאחר חישוב סדרה חשבונית נקבל O((m^2)\*n) כמו בסעיף הקודם.

1. א. 10,000.   
   log100=x. (הערכת זמן) log(n)=2x. (כאילו המחשב האיטי פעל פי 2 זמן).=> n=10,000.

ב. 200.

100=x. n=2x. n=200.

ג. sqrt(20,000)

100^2=x. n^2=2x. n=sqrt(20,000)

ד. 101.

2^100=x. 2^n=2x. n=101.

1. ב. אם המספרים לא היו שלמים, לא היה ניתן להתבסס על העובדה שההפרש בין כל שני איברים הוא לכל הפחות ההפרש בין האינדקסים שלהם. [0 ,1 ,1.1 ,2 ,3]

ג. אם היו חזרות של מספרים, שוב לא היינו יכולים להתבסס על אותה עובדה. [4,4,4,4,4]

1. ב. אם k=nlogn נקבל סיבוכיות של O(k+n)=O(nlogn), כמו של המקרה הממוצע של quicksort.

לכן נדרוש O(k)<O(nlogn). כלומר k מסדר גודל כלשהו אשר נמוך מסדר הגודל של nlogn, למשל: O(n), O(logn), O(