תרגיל בית מספר 4 - להגשה עד 21 במאי (יום חמישי) בשעה 23:55

קיראו בעיון את הנחיות העבודה וההגשה המופיעות באתר הקורס, תחת התיקייה assignments. חריגה מההנחיות תגרור ירידת ציון / פסילת התרגיל.

הגשה:

- תשובותיכם יוגשו בקובץ pdf ובקובץ py בהתאם להנחיות בכל שאלה.
- השתמשו בקובץ השלד skeleton4.py כבסיס לקובץ ה py אותו אתם מגישים.

לא לשכוח לשנות את שם הקובץ למספר ת"ז שלכם לפני ההגשה, עם סיומת py.

- בסה"כ מגישים שני קבצים בלבד. עבור סטודנטית שמספר ת"ז שלה הוא 012345678 הקבצים שיש להגיש הם 012345678.pdf ו- 012345678.py.
- הקפידו לענות על כל מה שנשאלתם.
- תשובות מילוליות והסברים צריכים להיות תמציתיים, קולעים וברורים.

להנחיה זו מטרה כפולה:

1. על מנת שנוכל לבדוק את התרגילים שלכם בזמן סביר.
2. כדי להרגיל אתכם להבעת טיעונים באופן מתומצת ויעיל, ללא פרטים חסרים מצד אחד אך ללא עודף בלתי הכרחי מצד שני. זוהי פרקטיקה חשובה במדעי המחשב.

דאגו שהפונקציות אותן אתם מממשים ירוצו בזמן סביר (פונקציות שירוצו יותר מדקה יאבדו ניקוד)

שאלה 1

בשאלה זו ננתח ונשווה את הסיבוכיות של מספר פונקציות רקורסיביות לחישוב מקסימום ברשימה.

את שתי הגרסאות הבאות, max1 ו max2, פגשנו בתרגול 6, ואף ניתחנו את סיבוכיות הזמן ואת עומק הרקורסיה שלהן:

```
 def max1(L):
   if len(L)==1:
        return L[0]
    return max(L[0], max1(L[1:]))

def max2(L):
    if len(L)==1:
        return L[0]
    l = max2(L[:len(L)//2])
    r = max2(L[len(L)//2:])
    return max(l,r)
```

א. להלן הפונקציה max11, שהיא גרסה הדומה ל- max1, אך הפעם ללא slicing. פונקציה זו מקבלת בנוסף לרשימה L שני אינדקסים אשר תוחמים את אזור החיפוש הרלבנטי. הקריאה הרקורסיבית הראשונה מתבצעת מתוך פונקצית המעטפת max_list11:

```
def max11(L,left,right):
    if left==right:
        return L[left]
    return max(L[left], max11(L,left+1,right))

def max_list11(L):
    return max11(L,0,len(L)-1)
```

השימוש במנגנון כזה של פונקצית מעטפת מקובל מאוד, ומאפשר למשתמש להעביר לפונקציה אך ורק את הקלט הדרוש (הרשימה עצמה), ולא פרמטרים נוספים שקשורים לאופן מימוש רקורסיבי כזה או אחר (גבולות שמאלי וימני).

נתחו את סיבוכיות הזמן ואת עומק הרקורסיה של max11. על הניתוח לכלול:

1. ציור סכמטי של עץ הקריאות הרקורסיביות עבור רשימה באורך n. העץ יכיל צומת עבור כל קריאה רקורסיבית. בתוך הצומת רישמו את אורך הרשימה עליה בוצעה הקריאה, ולצד הצומת רישמו חסם אסימפטוטי הדוק במונחי O(…) על סיבוכיות הזמן של אותו צומת כפונקציה של n.
2. ציון עומק עץ הרקורסיה כפונקציה של n.
3. ציון סיבוכיות הזמן (חסם אסימפטוטי הדוק במונחי O(…)) כפונקציה של n.

ב. השלימו בקובץ השלד את הפונקציה max22, שתעבוד באותו אופן כמו max2 (כלומר תבצע חלוקה של הבעיה בדיוק לאותן תתי בעיות), אבל הפעם ללא slicing. הקריאה הראשונה ל- max22 תהיה מתוך max_list22, כפי שמופיע בקובץ השלד:

```
def max_list22(L):
    return max22(L,0,len(L)-1)
```

ג. נתחו את סיבוכיות הזמן ואת עומק הרקורסיה של max22 בדומה לניתוח שביצעתם בסעיף א' 3-1 עבור max11.

ד. השלימו את הטבלה הבאה, המציינת את זמני הריצה של ארבע הפונקציות max1, max2, max_list11, max_list22, עבור רשימות בגודל n=1000, 2000, 4000. את זמני הריצה מדדו באמצעות מנגנון ה"סטופר" שהוצג בתרגיל בית 1, או באמצעות הפונקציה elapsed מההרצאות. זכרו שכדי לקבל תוצאות אמינות עדיף להריץ מספר גדול של הרצות ולמצע. מאחר שכברירת מחדל עומק הרקורסיה המקסימלי הינו 1000, חלק מהפונקציות לא יצליחו לרוץ. לכן יהיה עליכם לשנות את עומק הרקורסיה המקסימלי ל- 5000 באמצעות הפקודה sys.setrecursionlimit(5000) (עקבו אחר ההנחיות בשקפים של הרצאה 10).

Function	n = 1000	n = 2000	n = 4000
max1			
max2			
max_list11			
max_list22			

ה. הסבירו **<u>בקצרה</u>** כיצד תוצאות המדידה מסעיף ד' מתיישבות עם סיבוכיות זמני הריצה מסעיפים א3, ג3 ומהתרגול (של max1 ו- max2). התייחסו לקצב הגידול של זמני הריצה כתלות בגודל הקלט.

<u>שאלה 2</u>

בתרגול ראינו את בעיית העודף ואת פתרונה הרקורסיבי.

א. ציירו את עץ הרקורסיה אשר מתקבל מהרצת הפקודה הבאה (על הפונקציה change מהתרגול):

```
change(3, [1,2])
```

ב. השלימו בקובץ השלד את הפונקציה change_fast, שתפתור גם היא את בעיית העודף באופן דומה, אך הפעם עם מנגנון של memoization לחיסכון בזמן ריצה. הפונקציה, כמו קודמתה ללא memoization , מקבלת שני פרמטרים – amount ו- coins.

ג. ציירו את עץ הרקורסיה אשר מתקבל מהרצת הפקודה הבאה :

```
change_fast(3, [1,2])
```

ציירו רק את הקריאות שיווצרו בפועל, לא כולל אלה שנחסכו על ידי ה memoization.

שאלה 3

בכיתה הוצג המשחק זלול! (Munch!), משחק לוח לשני שחקנים אשר גורעים (זוללים) בזה אחר זה קוביות מתוך טבלת שוקולד על פי חוקים מוסכמים מראש. זהו, בין השאר, גם משחק עם מסר חינוכי אקטואלי: הזלילה אינה מומלצת! השחקן המפסיד הוא זה אשר נאלץ לבלוע את קובית השוקולד האחרונה (השמאלית התחתונה).
בהרצאה הראנו שקיימת "אסטרטגית ניצחון" עבור השחקן הפותח. פירוש הדבר הוא שאם השחקן הפותח (זה שמשחק ראשון) משחק היטב, מובטח כי ינצח, ללא תלות במהלכי המשחק של השחקן היריב. יש לציין שההוכחה מבטיחה קיום אסטרטגית ניצחון כזו, אך לא מהי (למשל מהו הצעד הראשון שעל השחקן הפותח לבצע על מנת לזכות).

בנוסף, הוצגה בכיתה הפונקציה הרקורסיבית win, אשר מקבלת כקלט את גודל הלוח n,m וכן ייצוג קומפקטי של מצב הלוח הנוכחי hlst, ומחזירה True אם זו קונפיגורציה מנצחת (או במלים אחרות, אם לשחקן שתורו לשחק כעת יש אסטרטגית נצחון), ו False אחרת.

א. ציירו את עץ הרקורסיה אשר מתקבל בעת הרצת הפונקציה win מהכיתה על טבלה מלאה בגודל 2x2. בכל צומת רשמו את הערך של הפרמטר hlst איתו בוצעה הקריאה הרקורסיבית, וכן האם קונפיגורצית הלוח שמקודדת על ידי hlst היא קונפיגורציה מנצחת.

ב. הריצו את הפונקציה win מהכיתה על קלטים שונים. בדקו ורשמו מהו ערך n הגדול ביותר עבורו הקריאה הבאה לפונקציה:

```
win(n, n+3, [n]*(n+2)+[n-1])
```

(כלומר קריאה על טבלה מלאה בגודל n x (n+3 אשר הפינה הימנית העליונה שלה כורסמה) מחזירה True או False תוך דקה לכל היותר על המחשב שלכם.

ג. שפרו את הפונקציה מהכיתה כך שתכלול memoization. ממשו את השכלול בפונקציה שחתימתה win_fast(n, m, hlst, show=False).

ד. בדומה לסעיף א', ציירו את עץ הרקורסיה אשר מתקבל בעת הרצת הפונקציה המשופרת win_fast על טבלה מלאה בגודל 2x2. ציירו רק את הקריאות שיווצרו בפועל, לא כולל אלה שנחסכו על ידי ה memoization.

ה. בדומה לסעיף ב', בדקו ורשמו מהו ערך n הגדול ביותר עבורו הקריאה הבאה לפונקציה המשופרת win_fast:

```
win_fast(n, n+3, [n]*(n+2)+[n-1])
```

מחזירה True או False תוך דקה לכל היותר על המחשב שלכם.

שאלה 4

בשאלה זו עליכם לכתוב את הפונקציה הרקורסיבית choose_sets(lst, k).

הפונקציה מקבלת רשימה של איברים lst ומספר שלם k, ומחזירה רשימה המכילה את כל הרשימות השונות באורך k שניתן ליצור מאיברי lst, ללא חשיבות לסדר האיברים. כלומר, את כל האפשרויות לבחור k איברים מתוך הרשימה lst, ללא חשיבות לסדר הבחירה. ניתן להניח שאיברי הרשימה lst יחודיים, כלומר, שאין איברים שחוזרים על עצמם.

שימו לב:

- כאן אנו מעוניינים לייצר ממש את כל האפשרויות השונות לבחור k איברים, ולא רק למצוא כמה אפשרויות כאלו יש.
- הערך המוחזר הוא רשימה של רשימות, וכל אחת מהרשימות הללו הינה בדיוק באורך k.
- סדר הרשימות בתוך רשימת העל אינו חשוב.
- כאמור, הסדר הפנימי בכל תת-רשימה אינו חשוב, ואסור שיהיו כפילויות. לדוגמא, הרשימה [1,2,3] שקולה לרשימה [3,2,1].

הנחיה:

- ניתן לקבל את כל תת-הרשימות באורך k ע"י איסוף של כל תת-הרשימות שמכילות את האיבר הראשון ברשימה וכל תת-הרשימות שאינן מכילות את האיבר הראשון ברשימה.
- שימו לב לערך ההחזרה של מקרה הבסיס (תנאי ההתחלה).
- ניתן להניח כי הקלט תקין – אין חזרות של איברים ברשימת הקלט ו-$0 \le k \le n$ הוא מספר שלם, כאשר n הוא אורך הרשימה lst.
- אין להשתמש בחבילות חיצוניות של פייתון בפתרון.

דוגמאות הרצה:

```
>>> choose_sets([1,2,3,4], 0)
[[]]
>>> choose_sets([1,2,3,4], 2)
[[4 ,3] ,[2 ,4] ,[2 ,3] ,[1 ,4] ,[1 ,3] ,[1 ,2]]
>>> choose_sets([1,2,3,4], 4)
[[4 ,3 ,2 ,1]]
>>> choose_sets(['a','b','c','d','e'], 4)
[['d', 'c', 'b', 'a'], ['e', 'c', 'b', 'a'], ['d', 'e', 'b', 'a'],
['c', 'd', 'e', 'a'], ['b', 'c','d', 'e']]
```

הנחיות הגשה:

השלימו את מימוש הפונקציה בקובץ השלד.

שאלה 5

א. כתבו פונקציה בשם density_primes שמקבלת שני פרמטרים: מספר שלם חיובי n ומספר שלם חיובי times.

```
density_primes(n, times=10000)
```

הפונקציה אומדת את השכיחות של מספרים ראשוניים בני בדיוק n ביטים (תזכורת: המספר היחיד שהביט השמאלי שלו הוא 0 הינו המספר 0) מבין כל הטבעיים בני בדיוק n ביטים באופן הבא: תדגום באקראי times מספרים טבעיים בני n ביטים, ותוציא כפלט את שכיחות הראשוניים שנמצאה (כלומר מספר הראשוניים שנמצאו, חלקי times). השתמשו בקוד לבדיקת ראשוניות מההרצאה (הפונקציה is_prime) שמצורף לקובץ השלד.
צרפו את המימוש לקובץ השלד.

בקובץ ה pdf הגישו את תוצאת הפעלת הפונקציה על times=10000 וחמש האפשרויות הבאות ל-n:
100, 200, 300, 400, 500.
בנוסף הסבירו בקצרה (לא יותר משלוש שורות) האם השכיחות שאמדתם מתנהגת כפי שניתן לצפות מתוך משפט המספרים הראשוניים?

ב. דני פצחני יודע שחוזקן של מערכות הצפנה רבות תלוי בקושי למצוא פירוק של מספר טבעי N, שידוע שהוא מכפלת שני ראשוניים, לגורמים הראשוניים שלו. כלומר עבור N=pq למצוא את p ואת q שידוע שהם ראשוניים. פתרונות נאיביים בסגנון trial division שראיתם בכיתה אינם יעילים מספיק (אקספוננציאליים), ולכן עבור N גדול, פתרונות כאלו אינם מעשיים. אך לדני יש רעיון אחר. הוא זומם להשתמש בפונקציה is_prime כדי לפצח הצפנות שכאלה.
להלן תוכניתו הזדונית:

בהינתן מספר N כזה – הוא יקרא לפונקציה is_prime(N,True).
הפונקציה תדפיס את העד (witness) שנמצא לפריקותו של N.
כעת דני יחלק את המספר N ב-witness שהודפס, והוא יקבל את הגורם השני של המספר N.

כידוע, ההסתברות ש- is_prime תטעה היא זניחה, וזו פונקציה שרצה בזמן ריצה פולינומי. בהנחה שהפונקציה is_prime אכן איננה טועה, האם תוכניתו של דני תצליח לפרק מספר גדול N=pq לגורמיו? אם לדעתכם לא, נמקו.

שאלה 6

יעל ומיכל החליפו ביניהן מפתח סודי באמצעות פרוטוקול Diffie-Hellman, כפי שניתן לראות בהרצות הבאות (הפונקציות find_prime ו- DH_exchange הן אלו שראינו בתרגול):

```
>>> p = find_prime(10)
>>> p
593
>>> g, a, b, x, y, key = DH_exchange(p)
>>> g, a, b, x, y, key
(9, 530, 147, 574, 527, 156)
```

סטודנט מהקורס מבוא מורחב מנסה לגלות את המפתח המשותף הנ"ל. לשם כך הוא מאזין לתשדורת בין מיכל ויעל, ולכן יש ברשותו את x ואת y ששלחו זה לזו. בנוסף, כמובן, יש ברשותו את p ואת g הפומביים. כעת, מנסה הסטודנט להריץ את הפונקציה crack_DH מהתרגול, שמטרתה לפתור את בעיית ה- Discrete log.

להלן ההרצה שביצע:

```
>>> crack_DH(p,g,x)
234
```

כפי שניתן לראות, הפונקציה לא גילתה את הסוד המקורי a=530, אלא ערך אחר, a'=234. אך כמובן שהסטודנט הזדוני אינו יודע זאת.

האם יכול הסטודנט, באמצעות המידע שברשותו כעת, לחשב את הסוד המשותף של יעל ומיכל (156 במקרה זה)? אם לדעתכם כן, הוכיחו זאת באופן מתמטי, עבור a'≠a כללי כלשהו, ועבור p, g, x, y, b כלליים כלשהם. אם לדעתכם לא, הסבירו מדוע לא.

סוף