תרגיל בית מספר 6 - להגשה עד 10/06/2022 בשעה 23:55

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

הנחיות לצורת ההגשה:

- תשובותיכם יוגשו בקובץ pdf ובקובץ py בהתאם להנחיות בכל שאלה.
- בסה"כ מגישים שני קבצים בלבד. עבור סטודנטית שמספר ת"ז שלה הוא 012345678 הקבצים שיש להגיש הם hw6_012345678.pdf ו- hw6_012345678.py.
- עבור קובץ ה-pdf: מומלץ מאוד להקליד את התשובות בו. ניתן גם לכתוב את התשובות בכתב יד ברור ולסרוק אותן, אבל שימו לב שתשובות שיכתבו בכתב יד לא ברור לא יבדקו, וערעורים בנושא זה לא יתקבלו.
- עבור קובץ ה-py: השתמשו בקובץ השלד skeleton6.py כבסיס לקובץ אותו אתם מגישים.
- לא לשכוח לשנות את שם הקובץ למספר ת"ז שלכם לפני ההגשה, עם סיומת py.

הנחיות לפתרון:

- הקפידו לענות על כל מה שנשאלתם.
- בכל שאלה, אלא אם מצוין אחרת באופן מפורש, ניתן להניח כי הקלט תקין.
- אין להשתמש בספריות חיצוניות פרט לספריות math, random, אלא אם נאמר במפורש אחרת.
- תשובות מילוליות והסברים צריכים להיות תמציתיים, קולעים וברורים.
  א. 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.
- נדרוש שכל הפונקציות שאנו מממשים תהיינה יעילות ככל הניתן. לדוגמה, אם ניתן לממש פתרון לבעיה בסיבוכיות $O(logn)$, ואתם מימשתם פתרון בסיבוכיות $\Theta(n)$, תקבלו ניקוד חלקי על הפתרון.
- בשאלות שבהן ישנה דרישה לניתוח סיבוכיות זמן הריצה, הכוונה היא לסיבוכיות זמן הריצה של המקרה הגרוע ביותר (worst-case complexity).

טבלת גרסאות:

27.05.2022	גרסה ראשונה
28.05.2022	תיקוני **typo** קטנים: 3ב' דוגמת הרצה, 3ג' חתימה בקובץ השלד

שאלה 1 – דקדוקים חסרי הקשר ו-CYK (30 נק')

תזכורת: בהינתן אלפבית סופי Σ, שפה היא קבוצה (אולי אינסופית) של מחרוזות מעל האלפבית Σ, כלומר קבוצה של מחרוזות כך שכל מחרוזת מכילה תווים רק מתוך הקבוצה Σ. בנוסף, נסמן ב-ε את המחרוזת הריקה.
לדוגמה אם $\Sigma = \{0,1,2\}$ אז $L = \{\varepsilon, 0, 01, 02, 22\}$ היא שפה (סופית) מעל Σ. בהינתן שפה L, נאמר שהיא שפה חסרת הקשר אם קיים דקדוק חסר הקשר $G = (V, \Sigma, R, S)$ כך שקבוצת המחרוזות שניתן לגזור באמצעות הדקדוק היא בדיוק L. במקרה זה נאמר ש-L היא השפה של הדקדוק G, ונסמן $L = \mathcal{L}(G)$.

סעיף א'

עבור כל אחת מהשפות הבאות, הראו שהשפה היא שפה חסרת הקשר על ידי כתיבת דקדוק מתאים. הגדירו באופן פורמלי את כל אחד מן הפרמטרים V, Σ, R (השתמשו באות S בתור משתנה ההתחלה שלכם). אין צורך להוכיח את נכונות הפתרון שלכם. הדקדוקים אינם צריכים להיות בצורת CNF.
הערה: בשאלה זו $\mathbb{N}$ היא קבוצת המספרים הטבעיים **כולל 0.**

i. $L_1 = \{0^{2n}12^{3n} : n \in \mathbb{N}\}$.
דוגמאות למחרוזות ששייכות לשפה L_1: $1, 001222, 00001222222 \ldots$

ii. השפה L_2 של כל הביטויים המתמטיים שניתן לייצר עם המשתנים x, y, z ופעולות $+,*$ וכן **עם סוגריים חוקיים**.
דוגמאות למחרוזות ששייכות לשפה L_2:
$$(x + y) * z$$
$$x * (z * (z + y))$$
ניתן להגדיר את L_2 פורמלית באופן הבא (זוהי הגדרה אינדוקטיבית):
$x, y, z \in L_2$, וכן לכל $A, B \in L_2$ מתקיים ש- $(A + B), (A * B), A + B, A * B \in L_2$

iii. $L_3 = \{x \in \{0,1\}^n : n \in \mathbb{N} \ and \ \#0 \ in \ x = \#1 \ in \ x\}$, כלומר שפת כל המחרוזות הבינאריות שבהן מספר האפסים שווה למספר האחדות.
דוגמאות למחרוזות ששייכות לשפה L_3: $01, 0110, 11010010 \ldots$

iv. בהינתן דקדוקים חסרי הקשר $G_1 = \{V_1, \Sigma_1, R_1, S_1\}, G_2 = \{V_2, \Sigma_2, R_2, S_2\}$ נגדיר את שפת האיחוד שלהם:
$$L_4 = \mathcal{L}(G_1) \cup \mathcal{L}(G_2)$$
הסבירו מדוע ניתן להניח בלי הגבלת הכלליות ש-$V_1 \cap V_2 = \emptyset$, ובנו את הדקדוק תחת ההנחה הזו. כלומר, פתרו את השאלה תחת ההנחה שהחיתוך ריק והסבירו כיצד היה הפתרון שלכם משתנה אילו קבוצות המשתנים לא היו זרות.

v. בהינתן דקדוק חסר הקשר $G = (V_1, \Sigma_1, R_1, S_1)$ בצורת CNF, נגדיר את השפה:
$$L_5 = \mathcal{L}(G) \cap \{x \in \Sigma_1^* : |x| \in \mathbb{N}_{even}\}$$
כלומר שפת כל המילים ב-$\mathcal{L}(G)$ באורך זוגי.
נקודה למחשבה: שימו לב שגם אם G לא היה נתון בצורת CNF, מכיוון שהשפה L_5 תלויה רק בשפה של G, ולא בדקדוק עצמו, ניתן היה להניח בלי הגבלת הכלליות ש-G בצורת CNF, וזאת מכיוון שכל דקדוק ניתן להמיר באופן יעיל לדקדוק שקול (כלומר, דקדוק המייצר את אותה השפה) בצורת CNF.

סעיף ב'

השלימו את המימוש של הפונקציה generate_language_rec(rule_dict, start_var, k, mem) המקבלת דקדוק G בצורת CNF המיוצג על ידי המילון rule_dict ומשתנה ההתחלה start_var, באותו האופן שבו ייצגנו דקדוק ב-CYK.
על הפונקציה להחזיר set שמכיל את כל המילים באורך k בשפה $\mathcal{L}(G)$.
נתונה לכם פונקציית המעטפה, פונקציית העזר sets_concat וחלק מן המימוש של הפונקציה הרקורסיבית. השלימו את חלקי הקוד החסרים.

בסעיפים הבאים בחרו את התשובה ההדוקה ביותר. למשל, אם סיבוכיות הפונקציה היא לינארית וסימנתם שהיא לכל היותר פולינומיאלית, התשובה אינה הדוקה מספיק. ציינו בקובץ ה-PDF את התשובה שבחרתם **ונמקו בקצרה**. הניחו כי גודל האלפבית וגודל הדקדוק הם קבועים.

i. גודל ה-set המוחזר מ-generate_language במקרה הגרוע הוא:
   1) לכל היותר לינארי ב-k.
   2) לכל היותר פולינומיאלי ב-k.
   3) לכל הפחות אקספוננציאלי ב-k.

ii. סיבוכיות הזמן של הפונקציה generate_language במקרה הגרוע היא:
   1) לכל היותר לינארית ב-k.
   2) לכל היותר פולינומיאלית ב-k.
   3) לכל הפחות אקספוננציאלית ב-k.

<u>סעיף ג'</u>

בקובץ השלד נתונה לכם הפונקציה what(rule_dict, start_var, k) המקבלת דקדוק חסר הקשר בצורת CNF ומספר טבעי k. הסבירו באופן **מדויק ותמציתי** (לא יותר משורה אחת) מה מחזירה הפונקציה.

בסעיפים הבאים בחרו את התשובה ההדוקה ביותר. למשל, אם סיבוכיות הפונקציה היא לינארית וסימנתם שהיא לכל היותר פולינומיאלית, התשובה אינה נכונה. ציינו בקובץ ה-PDF את התשובה שבחרתם **ונמקו בקצרה**. הניחו כי גודל האלפבית וגודל הדקדוק הם קבועים.

i. ערך ההחזרה של הפונקציה what במקרה הגרוע הוא:
   1) לכל היותר לינארי ב-k.
   2) לכל היותר פולינומיאלי ב-k.
   3) לכל הפחות אקספוננציאלי ב-k.

ii. סיבוכיות הזמן של הפונקציה what במקרה הגרוע היא:
   1) לכל היותר לינארי ב-k.
   2) לכל היותר פולינומיאלי ב-k.
   3) לכל הפחות אקספוננציאלי ב-k.

שאלה 2 – גנרטורים (20 נק׳)

הגדרה: גנרטור הוא **בעל השהייה סופית** (finite delay) אם כל קריאה ל-next עליו מסתיימת תוך זמן סופי (לא משנה כמה זמן). כל קריאת next תמיד מחזירה איבר או שגיאת StopIteration בפרק זמן סופי. שימו לב שגם גנרטור שמייצר סדרה אינסופית יכול להיות בעל השהייה סופית.

הגדרה: גנרטור g **מייצר את הקבוצה** S (סופית או אינסופית) אם:

1. g הוא בעל השהייה סופית.
2. $x \in S$ אמ״ם g מייצר את x לאחר מספר סופי של קריאות next.
3. g לא מייצר חזרות (כלומר, כל איבר ש-g מייצר הוא ייחודי).

בהגדרה זו אין חשיבות לסדר החזרת איברי S.

הגדרה: גנרטור g **מייצר את הסדרה** $\{a_n\}$ (סופית או אינסופית) אם g מייצר את קבוצת איברי הסדרה, וכן לאחר i קריאות next יוחזר הערך a_i (או StopIteration לאחר החזרת כל איברי הסדרה). כאן יש חשיבות לסדר החזרת איברי הסדרה.

לכל אחד מהגנרטורים הבאים, אם ניתן לבנות אותו כך שתהיה לו השהייה סופית, השלימו את פונקציית הגנרטור המתאימה בקובץ השלד. אחרת, הסבירו בקצרה מדוע לא ניתן לבנות אותו.

א. gen1() המייצר את הקבוצה $\mathbb{Z}^2$, כלומר כל זוגות המספרים **השלמים**.
תזכורת: ראיתם בתרגול שאלה דומה עבור הקבוצה $\mathbb{N}^2$.

ב. gen2(g) המקבל גנרטור g שמייצר סדרת מספרים כלשהי (g סופי או אינסופי, בעל השהייה סופית), מייצר את סדרת הסכומים החלקיים של איברי g. כלומר אם g מייצר את הסדרה $a_1, a_2, a_3, \ldots$ אז הגנרטור מייצר את הסדרה $a_1, a_1 + a_2, a_1 + a_2 + a_3, \ldots$.

ג. gen3(g) המקבל גנרטור g שמייצר סדרת מספרים כלשהי (g סופי או אינסופי, בעל השהייה סופית), ומייצר את קבוצת איברי g שגדולים מ-0.

ד. gen4(rules_dict, start_var) המקבל דקדוק חסר הקשר G בצורת CNF אשר שפתו $\mathcal{L}(G)$ אינסופית, ומייצר את הקבוצה $\mathcal{L}(G)$. הדקדוק G נתון על ידי המשתנים rules_dict, start_var כפי שראיתם בכיתה.
רמז: ניתן להיעזר בפונקציה שמימשתם בשאלה 1 סעיף ב׳.

ה. gen5(g1, g2) המקבל שני גנרטורים (סופיים או אינסופיים, בעלי השהייה סופית) ומייצר את החיתוך שלהם, כלומר את כל קבוצת האיברים שמיוצרים גם על ידי g1 וגם על ידי g2.

שאלה 3 – דחיסה (40 נק׳)

בשאלה זו הסעיפים אינם קשורים זה לזה.

סעיף א׳

נתון קורפוס corpus באורך n מעל א״ב בן $t \geq 2$ תווים $\{a_1, \dots, a_t\}$ כאשר כל תו מופיע בקורפוס לפחות פעם אחת.
בנוסף, נתון עץ האפמן H שנוצר כתוצאה מהרצת האלגוריתם של האפמן על הקורפוס corpus.
עבור כל אחד מארבעת הפריטים שלפניכם, ציינו בקובץ ה-PDF את ערכו כפונקציה של t, n, והסבירו בקצרה את תשובתכם. אם ישנו ערך יחיד, ציינו אותו במפורש. אם ישנו טווח ערכים אפשרי, תנו ערך תחתון וערך עליון **הדוקים** **ככל הניתן**. שימו לב שיש לתת תשובות מדויקות, ולא במונחי $O(\cdot)$ או $\Theta(\cdot)$.

תזכורת: גובה של עץ הוא אורך מסלול ארוך ביותר **בקשתות** מהשורש לעלה כלשהו בעץ. כמו כן משקל של עלה בעץ הוא שכיחות התו המתאים בקורפוס (היזכרו כיצד הוגדר בכיתה משקל של צומת פנימי בעץ).

i. מספר העלים בעץ H
ii. משקל השורש של העץ H
iii. גובה העץ H
iv. מספר הצמתים בעץ H

סעיף ב׳

במימוש שראיתם בכיתה של LZW_compress, עבור הפרמטרים $L = 2^5 - 1$ ו-$W = 2^{12} - 1$ ראיתם שכל חזרה ניתן לקודד על ידי 18 ביטים בעוד שכל תו ניתן לקודד על ידי 8 ביטים, ולכן ״לא משתלם״ לקודד חזרות באורך קטן מ-3 (תזכורת: ניתן לקודד 2 תווים תו-תו בעזרת 16 ביטים בעוד שקידודם כחזרה עולה 18 ביטים). שימו לב שעבור ערכי W, L שונים, יתכן שדווקא נרצה לקחת חזרות באורך קטן מ-3, או לחילופין לא ישתלם לנו לקחת חזרות באורך 3.
ממשו את הפונקציה repetition_threshold(W, L), שבהינתן W, L מחזירה את האורך המינימלי של חזרה שמשתלם לנו לקחת באלגוריתם למפל-זיו.
לדוגמה:

```
>>> repetition_threshold(2**12-1, 2**5-1)
3
```

סעיף ג׳

באלגוריתם למפל-זיו שראינו בכיתה, כל תו בודד קודד על ידי הביט 0 ואחריו 7 ביטים עבור ייצוג ה-ASCII שלו. בסעיף זה, נרצה לקודד תווים בודדים באמצעות קוד האפמן, במקום קוד ASCII. את החזרות נמשיך לקודד באותו האופן, וכן נמשיך להשתמש בביט אינדיקטור כדי להבדיל בין בלוק המייצג תו לבין בלוק המייצג חזרה.

תזכורת: נאמר שקוד c הוא קוד האפמן אם קיים טקסט corpus כלשהו כך שהרצת האלגוריתם של האפמן על corpus תניב את הקוד c. הקוד c מיוצג ע״י מילון (של פייתון) כפי שראינו בכיתה.

i. ממשו את הפונקציה LZW_compress_v2(text, c, W, L), המקבלת בנוסף קוד האפמן c, ומחזירה את ייצוג הביניים המתאים.
שימו לב – בדומה למימוש המקורי, לכל תו בטקסט נמצא חזרה מקסימלית שמתחילה בתו זה. נקודד את החזרה באמצעות תת הרשימה $[m, k]$ אמ״ם אורך הקידוד הבינארי של החזרה **קטן ממש** מאורך הקידוד הבינארי תו-תו לפי קוד האפמן c.
דוגמאות הרצה: (שימו לב כי ערכי W, L שונים מהערכים הדיפולטיבים שראיתם בכיתה)

```
>>> c = {'a':'0', 'b':'10', 'c':'110', 'd':'1110', 'e':'1111'}
>>> LZW_compress_v2("abcdeabccde", c, 2**5-1, 2**3-1)
['a', 'b', 'c', 'd', 'e', 'a', 'b', 'c', [6, 3]]
>>> LZW_compress_v2("ededaaaaa", c, 2**5-1, 2**3-1)
['e', 'd', [2, 2], 'a', 'a', 'a', 'a', 'a']
```

ii. ממשו את הפונקציה inter_to_bin_v2(intermediate, c, W, L), המקבלת גם היא את קוד ההאפמן c (שאיתו ייצרו את ייצוג הביניים intermediate), ומחזירה את מחרוזת הביטים הדחוסה המתאימה לייצוג זה.
<u>דוגמת הרצה:</u>

```
>>> c = {'a':'0', 'b':'10', 'c':'110', 'd':'1110', 'e':'1111'}
>>> inter_to_bin_v2(['e', 'd', [2, 2]], c, 2**5-1, 2**3-1)
0111101110100010010  # indicator bits are colored in red for better readability
```

iii. ממשו את הפונקציה bin_to_inter_v2(bits, htree, W, L) המקבלת **<u>עץ האפמן</u>** (לפי הייצוג שראיתם בכיתה) המתאים לקוד האפמן c שלפיו נדחסה המחרוזת, ומחזירה את ייצוג הביניים המתאים לה.
<u>דוגמת הרצה:</u>

```
>>> htree = ('a', ('b',('c',('d', 'e'))))
# This is the huffman tree corresponding to the c defined previously
>>> bin_to_inter("0111101110100010010", htree, 2**5-1, 2**3-1)
['e', 'd', [2, 2]]
```

<u>רמז</u>: השינויים הדרושים הם קצרים ומקומיים, ומרבית הקוד נשאר זהה לקוד שראיתם בכיתה. אין צורך להסתבך. ניתן להיעזר בקוד שראיתם בכיתה.

<u>הערה</u>: השתכנעו שהפונקציה LZW_decompress הממירה את ייצוג הביניים לטקסט המקורי תעבוד ללא שינוי (ובפרט היא אינה תלויה בקוד האפמן c שאיתו קודדנו את המחרוזת).

iv. <u>הערה</u>: בסעיף זה נניח לשם פשטות כי W=2**5-1 ו-L=2**3-1 קבועים.
בזכות תכונת ה-prefix free של קוד האפמן, האלגוריתם שמימשתם יכול לפענח כל מחרוזת בינארית דחוסה באופן יחיד. בפרט, מתקיימת הטענה הבאה:
לכל קוד האפמן c ולכל text1 != text2 הדחיסה תניב שתי מחרוזות בינאריות שונות.
וודאו שאתם מבינים מדוע טענה זו נכונה עבור קודים שהם prefix-free (אין צורך לכתוב זאת ב-PDF).
נרצה לבחון האם הטענה נכונה גם עבור קודים שאינם prefix-free.
לפניכם שני תנאים על הקוד c:

a. c הוא קוד חח"ע <u>שאינו</u> prefix-free (ובפרט אינו קוד האפמן) וגם <u>ואינו</u> uniquely decodable.
b. c הוא קוד חח"ע <u>שאינו</u> prefix-free (ובפרט אינו קוד האפמן) אבל <u>כן</u> uniquely decodable.

עבור כל אחד משני התנאים a ו-b (בנפרד), הוכיחו / הפריכו את הטענה הבאה. אם הטענה נכונה, הסבירו בקצרה כיצד לפענח ממחרוזת בינארית לייצוג הביניים המתאים, ואם היא לא נכונה תנו דוגמה נגדית.

<u>טענה</u>: לכל קוד c המקיים את התנאי, ולכל text1 != text2 הדחיסה מניבה שתי מחרוזות בינאריות שונות.
כלומר, לכל c, text1, text2 כנ"ל מתקיים:

```
>>> bin1 = inter_to_bin_v2(LZW_compress_v2(text1, c), c)
>>> bin2 = inter_to_bin_v2(LZW_compress_v2(text2, c), c)
>>> bin1 != bin2
True
```

שאלה 4 – קודים לתיקון שגיאות (20 נק׳)

סעיף א׳

לכל אחת מהטענות הבאות יש לסמן האם היא נכונה או לא. אם הטענה נכונה, יש להסביר בקצרה מדוע. אם היא לא נכונה, יש לספק דוגמא נגדית.

הערות:

- אם A, B הן קבוצות, $A \cap B$ הוא החיתוך שלהן ו-$|A|$ הוא מספר האיברים ב-A.
- נאמר ש-C הוא קוד (n,k,d) אם $C:\{0,1\}^k \to \{0,1\}^n$ היא פונקציה חח״ע, ומרחק ההאמינג של C הוא $\Delta C = d$.
- קוד נקרא **טריוויאלי** אם $d \le 1$.
- כפי שראיתם בהרצאה, עבור $y \in \{0,1\}^n$ אנחנו מגדירים את הכדור ברדיוס r סביב y על ידי
$$B(y,r) = \{z \in \{0,1\}^n | \Delta(y,z) \le r\}$$
כאשר Δ הוא מרחק האמינג.

i. אם C הוא קוד (n,k,d) לא טריוויאלי אזי לכל $y \in \{0,1\}^n$ מתקיים $\left|B\left(y,\left\lfloor\frac{d-1}{2}\right\rfloor\right) \cap \mathrm{Im}C\right| \le 1$

ii. אם C הוא קוד (n,k,d) לא טריוויאלי אזי לכל $y \in ImC$ מתקיים $|B(y,\mathrm{d}-1) \cap \mathrm{Im}C| = 1$

iii. אם C הוא קוד (n,k,d) לא טריוויאלי ו-ℓ מספר שלם המקיים כי $\ell > \left\lfloor\frac{d-1}{2}\right\rfloor$,
אזי לכל $y \in \{0,1\}^n$ מתקיים $|B(y,\ell) \cap \mathrm{Im}C| > 1$.

iv. אם C הוא קוד $(n,n-1,d)$ אזי בהכרח הוא טריוויאלי.

v. אם C הוא קוד (n,n,d) אזי בהכרח C הוא טריוויאלי.

סעיף ב׳

בכל אחד מתתי הסעיפים הבאים מתואר קוד C לתיקון שגיאות. עבור הודעה msg באורך k, ציינו מהם ערכי n, d של הקוד. כתבו את הערכים המדויקים, או טווח ערכים מצומצם ככל האפשר אם אין ערך מדויק יחיד. שימו לב כי ערכים אלו יכולים להיות תלויים בפרמטרים נוספים הנתונים בסעיף. הסבירו את תשובתכם.

i. נתון קוד חזרה R עם פרמטר חזרה t. כלומר, בהינתן הודעה $a = a_1 a_2 \dots a_k$ מילת הקוד $R(a)$ מורכבת מ-k בלוקים של תווים כאשר לכל $1 \le i \le k$ הבלוק ה-i מכיל t עותקים של התו a_i.
נסמן ב-Par את הפונקציה שמקבלת מחרוזת בינארית ומחזירה את ביט הזוגיות של המחרוזת.
כעת נגדיר את הקוד C על ידי:
$$C(msg) = R(msg) + Par(R(msg))$$

ii. C_1 הוא קוד (n_1,k,d_2) ו-C_2 הוא קוד (n_2,k,d_2). נגדיר את הקוד C על ידי:
$$C(msg) = C_1(msg) + C_2(msg)$$

iii. C_1 הוא קוד (n_1,k,d_1), ו-t מספר חיובי קבוע כך ש-$n_1 > t$. נגדיר את הקוד C על ידי:
$$C(msg) = C_1(msg)_{1\dots n_1-t}$$
כאשר $C_1(msg)_{1\dots n_1-t}$ היא הרישא באורך $n_1 - t$ של מילת הקוד $C_1(msg)$, כלומר המחרוזת המתקבלת ממחיקת t התווים האחרונים של $C_1(msg)$

שאלה 5 – עיבוד תמונה (20 נק׳)

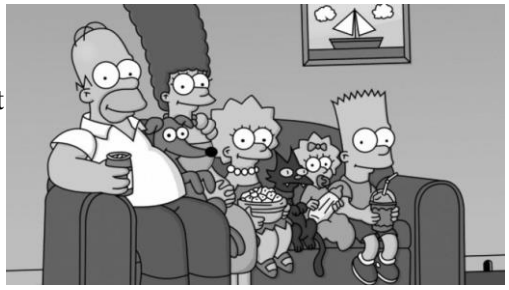
שימו לב שעל מנת להריץ את הפונקציות בשני הסעיפים הבאים עליכם להתקין את הספרייה PILLOW על ידי הרצת הפקודות הבאות ב-command prompt (אם זה לא עובד, נסו להחליף את המילה python ב-python3. אם אתם עדיין נתקלים בבעיות, היעזרו בפיאצה ובשעות החונכות):

```
python -m pip install --upgrade pip
python -m pip install --upgrade Pillow
```

סעיף א׳

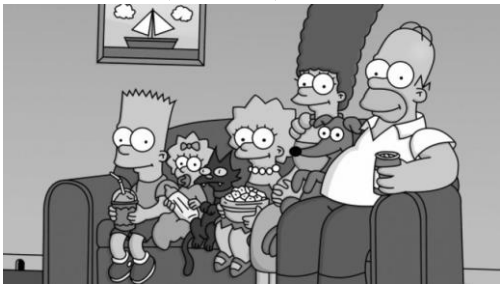
השלימו בקובץ השלד את שלוש השורות החסרות בפונקציה right_left, שמקבלת מטריצה שמייצגת תמונה ומחזירה מטריצה חדשה שמייצגת את התמונה שמתקבלת ע״י שיקוף התמונה על הציר האנכי.

לדוגמה עבור התמונה:



right_left ←

לאחר הפעלת הפונקציה תתקבל התמונה:

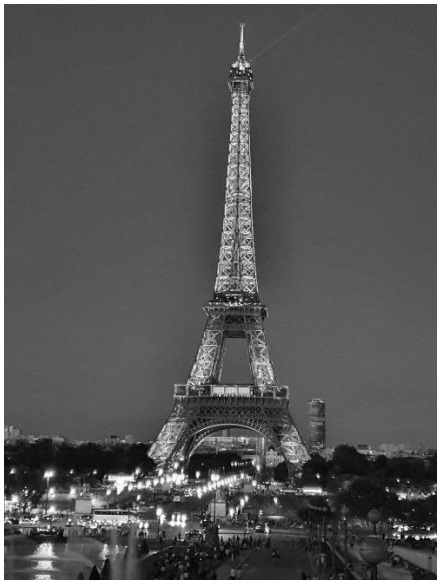


בדקו את הפונקציה שלכם על ידי הפעלתה על התמונה the-simpsons.jpg המצורפת בין קבצי התרגיל.

סעיף ב׳

לפניכם תמונה לפני ואחרי שהפעלנו עליה פעולה כלשהי.

לפני:



what2 ←

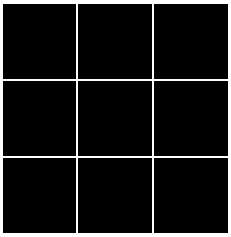
אחרי:

השלימו בקובץ השלד את הפונקציה what2 כדי להגיע לאותה תוצאה.
בדקו את הפונקציה שלכם על ידי הפעלתה על התמונה eiffel-tower.jpg המצורפת בין קבצי התרגיל.

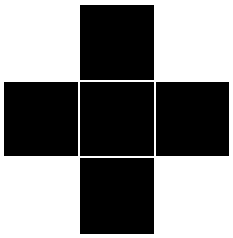
<u>סעיף ג'</u>

בכל תת סעיף נתונה שורת קוד, וסביבה של פיקסלים שחורים (ערך 0 greyscale).
הסביבה הנתונה היא סביבה **<u>פנימית</u>** בתוך התמונה img, המוקפת בפיקסלים לבנים (ערך 255 greyscale). הניחו כי יש מסביב לסביבה לפחות 3 פיקסלים לבנים בכל כיוון.
לכל תת סעיף, כתבו בטבלה בקובץ ה-PDF את ערכי ה-greyscale של הסביבה לאחר הרצת הפקודה הבאה:

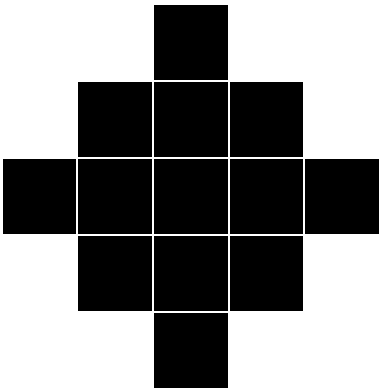
i. local_means(img, kx=1, ky=1)



ii. local_medians(img, kx=1, ky=1)



iii. local_medians(img, kx=2, ky=2)



<u>סוף.</u>