<u>תרגיל בית מספר 6 - להגשה עד 10/01/2022 בשעה 23:55</u>

קראו 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 skeleton6.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 hw6_012345678.pdf ו- hw6_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.

שאלה 1 (30 נק׳)

שאלה זו עוסקת בדקדוקים חסרי הקשר.

תזכורת: בהינתן אלפבית Σ, שפה היא קבוצה (אולי אינסופית) של מחרוזות מעל האלפבית Σ, כלומר קבוצה של מחרוזות כך שכל מחרוזת מכילה תווים רק מתוך הקבוצה Σ.
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 $L = \mathcal{L}(G)$.

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. $L_1 = \{0^n 1 2^{2n} : n \in \mathbb{N}\}$

ii. השפה L_2 של כל הביטויים המתמטיים שניתן לייצר עם x, y, z ופעולות $+, *$ וכן **עם סוגריים חוקיים** (בתרגול 11 ראינו את הדוגמה הזו אבל ללא סוגריים). לדוגמה:

$$(x + y) * z \in L_2$$
$$x * (z * (z + y)) \in L_2$$

iii. בהינתן דקדוקים $G_1 = \{V_1, \Sigma_1, R_1, S_1\}, G_2 = \{V_2, \Sigma_2, R_2, S_2\}$ נגדיר את שפת האיחוד שלהם:

$$L_3 = L(G_1) \cup L(G_2)$$

שימו לב שהתשובה צריכה להיות תלויה בפרמטרים של הדקדוקים G_1, G_2.

iv. $L_4 = \{x \in \{0,1\}^n : n \in \mathbb{N}\ and\ \#0\ in\ x = \#1\ in\ x\}$, כלומר שפת כל המחרוזות הבינאריות שבהן מספר האפסים שווה למספר האחדות.

ב.

i. 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 ומרבית המימוש של הפונקציה הרקורסיבית. הורידו את כל סימני ה-# מהקוד בקובץ השלד והשלימו את החסר היכן שמצוין (בכל מקום שצריך להשלים מתאימה שורת קוד אחת).

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**. ניתן להניח שגודל האלפבית הוא קבוע.

ii. סיבוכיות **הזיכרון** 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.

iii. 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.

ג.

i. בקובץ השלד נתונה לכם הפונקציה what(rule_dict, start_var, k) המקבלת מילון 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**. ניתן להניח שגודל האלפבית הוא קבוע.

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

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

<u>שאלה 2 (15 נק')</u>

שאלה זו עוסקת בגנרטורים. בשאלה זו ניתן להתעלם משגיאות הקשורות לאי דיוק של floating point.

א. ממשו את פונקציית הגנרטור get_next_sum(gen) שמקבלת גנרטור אינסופי שמייצר סדרת מספרים מטיפוס float שונים מאפס $\{a_n\}_{n=1}^{\infty}$, ומחזירה גנרטור אשר בקריאת ה-next ה-n-ית שלו מחזיר ערך הסכום:

$$\frac{1}{a_1}+\frac{1}{a_1\cdot a_2}+\cdots+\frac{1}{a_1a_2\cdot\ldots\cdot a_n}$$

<u>הנחיה</u>: על כל קריאת next לגנרטור לרוץ בזמן $O(1)$ במקרה הגרוע ולהשתמש ב-$O(1)$ זיכרון. ניתן להניח שפעולת next על גנרטור הקלט gen רץ בזמן ובמקום $O(1)$.

<u>נקודה למחשבה</u>: כבר נתקלתם בסכום מהצורה הזו <u>בתרגיל בית 3 שאלה 3א'</u>, בהקשר של חישוב נומרי. חשבו כיצד גנרטור כזה היה עוזר לכם במימוש ההוא (רשות – ממשו מחדש את הפונקציה מתרגיל בית 3 עם הגנרטור).

ב. ממשו את פונקציית הגנרטור gen_sequence() כך שיתקיים שאם ניתן אותה כקלט לפונקציית הגנרטור מסעיף א', קריאות next על הגנרטור מסעיף א' יתנו לנו קירוב הולך ומשתפר של המספר e.

<u>רמז</u>: $\sum_{i=0}^{\infty}\frac{1}{i!}=e$, כאשר על פי הגדרה $0!=1$.

דוגמת הרצה:

```
>>> gen1 = gen_sequence()
>>> gen2 = get_next_sum(gen1)
>>> for i in range (20):
      next(gen2)
>>> abs(math.e - next(gen2)) < 0.00001
True
```

שאלה 3 (30 נק')

שאלה זו עוסקת בעצי האפמן.

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 **<u>הדוקים ככל הניתן</u>**. שימו לב שיש לתת תשובות מדויקות, ולא במונחי $O(\cdot)$.

<u>תזכורת</u>: גובה של עץ הוא אורך מסלול ארוך ביותר **<u>בקשתות</u>** 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

ב. נגדיר את ה**רמה ה-i** בעץ כקבוצת כל הצמתים במרחק בדיוק i מהשורש (כלומר הרמה ה-0 כוללת רק את השורש, הרמה ה-1 כוללת רק את בניו של השורש, הרמה ה-2 כוללת את בניו של בניו של השורש וכו').

כמו כן נגדיר את **משקל** הרמה ה-i להיות סכום המשקלים של הצמתים ברמה ה-i.

הוכיחו או הפריכו את הטענות הבאות. אם הטענה נכונה, הסבירו בקצרה מדוע היא נכונה. אם הטענה אינה נכונה, ספקו דוגמה נגדית לקורפוס ולעץ שמתקבל ממנו באמצעות האלגוריתם שנלמד בכיתה, והסבר קצר מדוע הדוגמה מפריכה את הטענה.

i. לכל קורפוס corpus ולכל עץ האפמן H שנוצר מהקורפוס, לכל הרמות משקל זהה.
ii. לכל קורפוס corpus ולכל עץ האפמן H שנוצר מהקורפוס, ישנתן שתי רמות שונות שמשקלן שונה.

ג. בתרגול 12 ניתחנו את הסיבוכיות של הפונקציות של קידוד האפמן תחת ההנחה ש-$|\Sigma| = O(1)$. בסעיף זה נרצה להבין מה משתנה בניתוח בכל אחד מהשלבים ללא הנחה זו.

נסמן ב-Σ את האלפבית, ב-n את אורך הקורפוס corpus, ב-m את אורך ההודעה text, וב-b את אורך ההודעה המקודדת bits, בדומה לסימונים מהתרגול.

עבור כל אחת מהפונקציות הבאות שמופיעות במחברת תרגול 12, נתחו בקצרה את סיבוכיות **הזמן הממוצעת,** ללא ההנחה ש-$|\Sigma| = O(1)$. ניתן להניח שפעולות אריתמטיות לוקחות זמן קבוע.

i. char_count(corpus)
ii. build_huffman_tree(char_count_dict)
iii. generate_hcode(htree, prefix="")
iv. compress(text, hcode)
v. decompress(bits, htree)

<u>שאלה 4 (25 נק')</u>

השאלה עוסקת בשינוי באלגוריתם למפל-זיו לדחיסת טקסט.
כזכור, הפונקציה LZW_compress מחזירה את ייצוג הביניים של דחיסת למפל-זיו של המחרוזת text. למשל:

```
>>> LZW_compress("abcdabc")
['a', 'b', 'c', 'd', [4, 3]]
>>> LZW_compress("abab")
['a', 'b', 'a', 'b']
>>> LZW_compress("ababab")
['a', 'b', [2, 4]]
```

בנוסף, ראינו בכיתה את הפונקציה:

```
def inter_to_bin(intermediate, W=2**12-1, L=2**5-1)
```

שבהינתן רשימה lst עם ייצוג ביניים של מחרוזת דחוסה, מחזירה מחרוזת של ביטים, המייצגת את רצף הביטים לאחר הדחיסה. נזכיר, שתו שלא נדחס ייוצג ע"י הביט 0 ואחריו 7 ביטים עבור התו עצמו (סה"כ 8 ביטים, כלומר גם בתרגיל זה אנחנו מניחים לשם פשטות כי אנחנו מטפלים רק בתווי ASCII), ואילו מקטע שנדחס ייוצג ע"י הביט 1 ואחריו 12 ביטים עבור ההיסט אחורה, ו- 5 ביטים עבור אורך המקטע שנדחס (סה"כ 18 ביטים). שימו לב שבחישוב זה לקחנו בחשבון את ערכי ברירת המחדל של הפרמטרים W,L של שתי הפונקציות.

דוגמאות הרצה:

```
>>> inter_to_bin(LZW_compress("abcdabc"))
'01100001011000100110001101100100100000000010000011'
>>> len(inter_to_bin(LZW_compress("abcdabc")))
50      # 4*8 + 18
>>> inter_to_bin(LZW_compress("abab"))
'01100001011000100110000101100010'
>>> len(inter_to_bin(LZW_compress("abab")))
32      # 4*8
```

בעמוד הבא מופיעה הפונקציה `LZW_compress_new` שמציגה מימוש של אלגוריתם שונה במעט עבור דחיסת למפל זיו.

ענו בקובץ ה pdf על הסעיפים הבאים ביחס ל: `LZW_compress, LZW_compress_new`.

א. תנו דוגמא למחרוזת s המקיימת:

`LZW_compress(s) = LZW_compress_new(s)`.

מה יהיה הפלט (ייצוג הביניים) שיתקבל בשתי ההרצות?

ב. טענה: קיימת מחרוזת s שמקיימת:

`len(inter_to_bin(LZW_compress_new(s))) < len(inter_to_bin(LZW_compress(s)))`

תנו דוגמא למחרוזת s כזו בצירוף שני ייצוגי הביניים המתקבלים ע"י הפעלת כל אחת מהפונקציות הנ"ל, או הסבירו מדוע אין מחרוזת s כזו.

ג. טענה: קיימת מחרוזת s שמקיימת:

```
len(inter_to_bin(LZW_compress_new(s))) >  len(inter_to_bin(LZW_compress(s)))
```

תנו דוגמא למחרוזת s כזו בצירוף שני ייצוגי הביניים המתקבלים ע״י הפעלת כל אחת מהפונקציות הנ״ל, או הסבירו מדוע אין מחרוזת s כזו.

```
def LZW_compress_new(text, start=0, W=2**12-1, L=2**5-1):
    n = len(text)
    if start >= n:
        return []

    #find the maximal length matching
    m,k = maxmatch(text, start, W, L)

    res1 = [text[start]] + \
          LZW_compress_new(text, start+1, W, L)
    res1_len = len(inter_to_bin(res1, W, L))

    if k < 3:
        return res1

    res2 = [[m,k]] + LZW_compress_new(text, start+k, W, L)
    res2_len = len(inter_to_bin(res2, W, L))

    if (res2_len < res1_len):
        return res2
    return res1
```

שאלה 5 (20 נק')

שאלה זו עוסקת בקודים לתיקון שגיאות.

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 **<u>לא טריוויאלי</u>** אם $d > 1$.
- 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 $d = 1$.

v. אם C הוא קוד (n,n,d) אזי בהכרח $d = 1$.

ב. בסעיף זה נתייחס לקודים מהצורה $C:\{0,1\}^k \to \{0,1,2,3\}^n$, כלומר C היא פונקציה חח״ע שממירה מחרוזות בינאריות באורך k למחרוזות בבסיס 4 באורך n.

עבור כל אחד משני הקודים הבאים:

- תארו את פונקציית הפענוח D המתאימה ל-C, כלומר פונקציה $D:\{0,1,2,3\}^n \to \{0,1\}^k$ כך שלכל $x \in \{0,1\}^k$ מתקיים השוויון $D\big(C(x)\big) = x$.
- אם הקוד הוא טריוויאלי – תנו דוגמה של שתי מילות קוד במרחק 1 זו מזו. אם הקוד אינו טריוויאלי, הוכיחו שמרחק ההאמינג של הקוד הוא לפחות 2.

i. $C:\{0,1\}^n \to \{0,1,2,3\}^n$ מקבלת מחרוזות בינארית באורך n, נסמנה $b_0 b_1 \dots b_{n-1}$ כאשר $b_i \in \{0,1\}$.
נסמן ב-t את המספר $b_0 b_1$ בבסיס 4, פורמלית $t = b_0 \cdot 2^1 + b_1 \cdot 2^0$, כלומר $t \in \{0,1,2,3\}$.
נסמן ב-p את ה-parity bit של מחרוזת הקלט, פורמלית $p = \sum_{i=0}^{n-1} b_i \ mod\ 2$, כלומר $p \in \{0,1\}$.
ניתן להניח כי $n \ge 2$. הפונקציה C מוגדרת על ידי:
$$C(b_0 b_1 \dots b_{n-1}) = t b_2 \dots b_{n-1} p$$
לדוגמה עבור $n = 6$:
$$C(110010) = 300101$$
$$C(011000) = 110000$$

ii. $C:\{0,1\}^n \to \{0,1,2,3\}^n$ אותו דבר כמו בסעיף i אבל הפעם p מוגדר להיות $p = \sum_{i=0}^{n-1} b_i \ mod\ 3$, כלומר $p \in \{0,1,2\}$.

לדוגמה עבור $n = 6$:
$$C(110010) = 300100$$
$$C(011000) = 110002$$

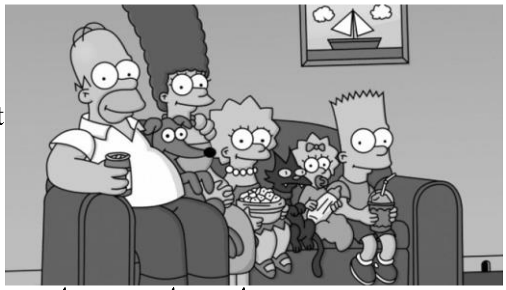
<u>שאלה 6 (10 נק׳)</u>

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

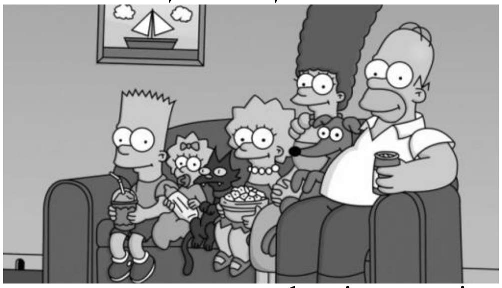
א. השלימו בקובץ השלד את שלוש השורות החסרות בפונקציה right_left, שמקבלת מטריצה שמייצגת תמונה ומחזירה מטריצה חדשה שמייצגת את התמונה שמתקבלת ע״י שיקוף התמונה על הציר האנכי.

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.

ב. לפניכם תמונה לפני ואחרי שהפעלנו עליה פעולה כלשהי.

לפני:

what2

אחרי:

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

<u>סוף.</u>