אוניברסיטת תל אביב - בית הספר למדעי המחשב
מבוא מורחב למדעי המחשב, אביב 2015

תרגיל בית מספר 6 (אחרון!) - להגשה עד 21 ביוני (יום ראשון) 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 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 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.

הערות:

1. כל השאלות בתרגיל זה מבוססות על שאלות ממבחנים משנים קודמות, עם שינויים מסויימים.
2. לייד כל שאלה מצויין מספר הנקודות שהיא מקנה בתרגיל הבית (לא בהכרח מספר הנקודות שהשאלה היתה שווה במבחן). ניתן להגיע למקסימום של 110 נקודות.
3. שאלה 4 איננה להגשה, ותקבלו עליה אוטומטית 30 נק' (כלומר הציון מתחיל מ- 30).

שאלה 1 – קארפ-רבין (40 נק')

בשאלה זו נרצה לענות על השאלה, האם תמונה נתונה מכילה תת-תמונה ריבועית בגודל נתון שחוזרת על עצמה יותר מפעם אחת, כאשר שני מופעים של תת תמונה יכולים לחפוף אחד את השני באופן חלקי. נכנה את תת-התמונה "חלון", ונניח שגודלו k x k פיקסלים. התמונה השלמה היא בגודל n שורות על m עמודות, ומתקיים k<= min(n,m). התמונה וכן החלון ייוצגו באמצעות המחלקה Matrix שראינו בקורס. כל פיקסל מייצג ערך אפור בין 0 (שחור) ל- 255 (לבן).

פתרון יעיל אפשרי מתבסס על הרעיון של אלגוריתם Karp-Rabin, בו השתמשנו על מנת לחפש מחרוזת תבנית בתוך מחרוזת טקסט: מחשבים מעין טביעת אצבע של כל החלונות בגודל k x k אשר מוכלים בתמונה הגדולה. מדווחים על חזרה אם נמצאו שתי טביעות אצבע שוות.

לשם פשטות ניתוח הסיבוכיות, בכל הסעיפים נניח כי פעולות חיבור וחיסור והשוואה בין מספרים שלמים רצות בזמן קבוע O(1) (כלומר ללא תלות בגודל המספר).

נגדיר אם כן פונקציה fingerprint, אשר בהינתן מטריצה ריבועית k x k מחזירה מספר, שנקרא לו "טביעת אצבע" של המטריצה:

```
def fingerprint(mat):
   assert isinstance(mat,Matrix)
   k,makesure = mat.dim()
   assert k == makesure

   return sum(mat[i,j] for i in range(k) for j in range(k))
```

לצורך פתרון יעיל, נזדקק לפונקציה move_right אשר מקבלת (בסדר זה) תמונה mat (כלומר אובייקט מסוג Matrix), אינדקסי שורה i ועמודה j של פיקסל בתוכה, גודל חלון k, ואת טביעת האצבע fp של החלון בגודל kxk, אשר הפינה השמאלית העליונה שלו ממוקמת mat[i][j]. הפונקציה מחזירה את טביעת האצבע של החלון אשר מתקבל על ידי הזזת החלון **ימינה** בפיקסל אחד. הפונקציה תניח כי החלון מימין אכן קיים (כלומר שלא הגענו לגבול הימני של התמונה).

לדוגמה, לאחר רצף הפקודות

```
fp = fingerprint(mat[0:k,0:k])
right_fp = move_right(mat,0,0,k,fp)
```

מתקיים

```
right_fp == fingerprint(mat[0:k,1:k+1])
```

א. השלימו בקובץ השלד את מימוש הפונקציה move_right, בסיבוכיות זמן ריצה O(k).

ב. השלימו בקובץ השלד את מימוש הפונקציה move_down, בסיבוכיות זמן ריצה O(k). ההבדל בין פונקציה זו לזו מסעיף א' הוא ש move_down מחזירה את טביעת האצבע של החלון אשר מתקבל על ידי הזזת החלון

המקורי **מטה** בפיקסל אחד. גם כאן הפונקציה מניחה כי החלון שלמטה אכן קיים (כלומר שלא הגענו לגבול התחתון של התמונה).

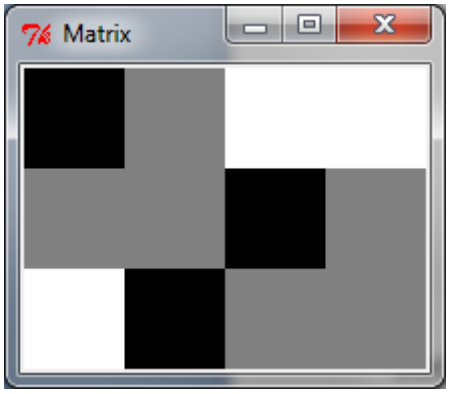
ג. עתה נממש את הפונקציה has_repeating_subfigure, שמקבלת מטריצה mat שמייצגת תמונה, וגודל צלע k של חלון ריבועי. הפונקציה תחזיר True אם יש בתמונה תת-תמונה ריבועית בגודל kxk שמופיעה בה יותר מפעם אחת, אחרת False. כאמור, מותרות חפיפות בין תת-תמונות.

הנחיות: (1) מותר שהפונקציה תחזיר תשובה שגויה, אם לשני "חלונות" שונים יש אותה טביעת אצבע. (2) חישוב טביעות האצבע ייעשה ע"י הפונקציות מהסעיפים הקודמים. (3) המקרה הגרוע ביותר מבחינת סיבוכיות הריצה הוא כאשר התמונה אינה מכילה תת-תמונה חוזרת (מדוע?). במקרה זה סיבוכיות הזמן הדרושה לחישוב כל טביעות האצבע תהיה O(mnk), ואילו סיבוכיות הזמן הדרושה לכלל הבדיקות האם יש טביעות אצבע חוזרות (בהינתן טביעות האצבע) תהיה O(mn) בממוצע (חישבו באיזה מבנה נתונים של פייתון יש לאחסן את טביעות האצבע כדי לעמוד בדרישה האחרונה).

ד. ציינו את חסרונה העיקרי של הפונקציה fingerprint שהופיעה בתחילת השאלה, ביחס לבעיה אותה אנו מנסים לפתור בשאלה זו. תארו במילים שיפור אפשרי לפונקציה, שיסייע להתגבר על חסרון זה.

דוגמאות הרצה (שחור – 0, לבן – 255, אפור - 128):

```
>>> im = Matrix.load("./sample.bitmap")
>>> im.display(zoom = 50)
>>> k=2
>>> fingerprint(im[:k,:k])
384
>>> fingerprint(im[1:k+1,1:k+1])
256
>>> move_right(im, 0, 0, k, 384)
511
>>> move_down(im, 0, 1, k, 511)
256
>>> has_repeating_subfigure(im, k)
True
>>> has_repeating_subfigure(im, 3)
False # there is no repeating subfigure of size 3x3
```



<u>שאלה 2 – דחיסת האפמן (20 נק')</u>

א. נסמן ב- a_i את המספר ה-i בסדרת פיבונאצ'י, כאשר האיבר הראשון הינו $a_1 = 1$ והאיבר השני הינו $a_2 = 1$. הוכיחו כי מתקיים:

$a_n < \sum_{i=1}^{n-1} a_i < a_{n+1}$ לכל $n \geq 2$ טבעי.

ב. נתון קורפוס (corpus) שבו תדירויות התווים השונים הן n מספרי פיבונאצ'י הראשונים (1,1,2,3,...).
מהו אורך קידוד האפמן הקצר ביותר ומהו אורך קידוד האפמן הארוך ביותר של תו כלשהו ע"פ קורפוס זה? נמקו תוך שימוש בטענה המוצגת בסעיף א'.

ג. נתון קורפוס עם k=128 תווים שונים, בעלי התדירויות: $0 < a_1 < a_2 < \cdots < a_k$. בנוסף מתקיים:
$a_k < 2a_1$.

יהי p התו בעל התדירות המינימלית (a_1), ו- q התו בעל התדירות המקסימלית (a_k).

יהיו $C(p)$, $C(q)$ קודי ההאפמן שמתקבלים עבור התווים q, p בהתאמה.

מהו ההפרש בין $|C(p)|$ (מספר הביטים שדרושים כדי לקודד את התו p) לבין $|C(q)|$ (מספר הביטים שדרושים כדי לקודד את התו q)?

על תשובתכם להיות מנומקת.

שאלה 3 – זיו למפל – (20 נק')

א. כזכור, באלגוריתם Lempel-Ziv דוחסים חזרות באורך לפחות 3 (מתעלמים מחזרות באורך 1,2 משום שדחיסתן אינה משתלמת). אם נסמן ב-L את אורך החזרה המינימלי שהאלגוריתם דוחס, אז L=3.
האם תיתכן מחרוזת שדחיסתה עם L=4 תהיה יעילה יותר מאשר עם L=3? כלומר האם ייתכן שגם אם גילינו חזרה באורך 3, ישתלם לא לדחוס אותה? אם לדעתכם כן, רשמו דוגמה למחרוזת כזו, וכן את את ייצוג הביניים* של הדחיסה, עבור L=3 ועבור L=4. אם לדעתכם לא, הסבירו מדוע.
* דוגמה לייצוג ביניים: ייצוג הביניים של המחרוזת 'abcabcdededе' הוא ['a','b','c',(3,3),'d','e',(2,4)]

ב.

i. לפניכם מוצג קוד עבור הפונקציה genString(n) שמייצרת מחרוזת באורך n מתוך התפלגות ידועה של שכיחות אותיות (הנתונה ע"י המחרוזת freq בקוד).

```
def genString(n):
    freq = 'a'*25+'bcdefghijklmnopqrstuvwxyz'
    randLetters = [random.choice(freq) for i in range(n)]
    return ''.join(randLetters)
```

תהי s=genString(100000). איזו דחיסה צפויה לתת יחס דחיסה טוב יותר עבור s: Huffman או Lempel-Ziv? הסבירו את תשובתכם בצירוף מספר דוגמאות הרצה שיתמכו בה. אין צורך בהוכחה מתמטית פורמלית.
הבהרה: קידוד Huffman כאן ישתמש ב-s הן בתור corpus והן בתור text.

ii. נחליף את המחרוזת freq במחרוזת הבאה: freq = 'a'*2500+'bcdefghijklmnopqrstuvwxyz'
האם לדעתכם התשובה תשתנה? הסבירו.

ג. נניח שעבור טקסט באורך n, מאפשרים לאורך החזרה המקסימלי באלגוריתם Lempel-Ziv להיות n-1 (במקום 31 כפי שמופיע בערכי ברירת המחדל של האלגוריתם שהוצג בהרצאה). שאר פרטי האלגוריתם ללא שינוי. רוצים לדחוס באופן זה את המחרוזת '01010101...' באורך n.
כיצד נראה ייצוג הביניים של הדחיסה? מהו יחס הדחיסה (=מספר הביטים במחרוזת הדחוסה חלקי מספר הביטים במחרוזת ללא שימוש בדחיסת למפל-זיו) כתלות ב- n? תנו תשובה בסדר גודל במונחים של O(...).

טיפ: כדאי לבדוק את התשובות בשאלה זו ע"י הרצות...

שאלה 4 – קודים לאיתור ולתיקון שגיאות (לא להגשה, 30 נק')

חלק ראשון

הקוד לתיקון טעויות המתואר כאן מעתיק 3 ביטים של אינפורמציה למילות קוד בנות 7 ביטים, על פי הסכמה הבאה:

$$(x_1, x_2, x_3) \rightarrow (x_1, x_2, x_3, x_1+x_2, x_1+x_3, x_2+x_3, x_1+x_2+x_3)$$

כאשר הסכומים מחושבים מודולו 2.

א. בטבלה הבאה, השלימו בכל שורה את מילת הקוד המתקבלת מ- 3 הביטים הרשומים בה.

(x_1, x_2, x_3)	$(x_1, x_2, x_3, x_1+x_2, x_1+x_3, x_2+x_3, x_1+x_2+x_3)$
(0 , 0 , 0)	
(0 , 0 , 1)	
(0 , 1 , 1)	
(1 , 1 , 1)	

ב. מהו המרחק המינימלי, d, של הקוד? רשמו שתי מילות קוד שונות w_1, w_2, שהמרחק ביניהן הוא d.

ג. <u>טענה</u>: קיימת מילה $y \in \{0,1\}^7$ כך שיש שתי מילות קוד שונות w_1, w_2, המקיימות: המרחק של שתיהן מ- y שווה, ומרחק זה הוא המרחק המינימלי מ- y למילת קוד כלשהי.

החליטו אם הטענה הנ"ל נכונה. אם לדעתכם הטענה נכונה – תנו דוגמה ל- y, w_1, w_2 כאלו. אחרת – הסבירו מדוע לא.

חלק שני

להלן פונקציית קידוד עבור קוד חדש בשם bad_coding , המקבלת רשימת ביטים x ומוציאה רשימת ביטים.

```
def bad_coding(x):
        z = (x[0]+x[1]) % 2
        return (x+[z])*4
```

תזכורת: קוד $C : \{0,1\}^k \rightarrow \{0,1\}^n$ עם מרחק מינימלי d נקרא קוד מטיפוס $[n, k, d]$.

האורך של x יסומן כרגיל ב- |x|.

השלימו את המשפט הבא: bad_coding הוא קוד מטיפוס [n=____ , k=_____ , d=_____].

סוף