CS1001.py HW6

שאלה 2

סעיף א'

קוד ההאפמן האופטימלי עבור הקורפוס הנתון (תוך שימוש בקוד מהכיתה, אך ניתן להראות גם ידנית באמצעות עצים כפי שנלמד בכיתה):

{'h': '0', 'g': '10', 'f': '110', 'e': '1110', 'd': '11110', 'c': '111110', 'a': '1111110', 'b': '1111111'}

סעיף ב'

סדרת פיבונאצ'י מקיימת את התכונה שכל איבר בסדרה שווה לסכום שני קודמיו (החל מהאיבר השלישי). ניתן להראות בקלות באמצעות אינדוקציה (לא נראה כי ביקשו הוכחה לא מפורטת) שהטענה הבאה מתקיימת: $\boxed{\forall n \in \mathbb{N}.\ S_n < a_{n+2}}$, כאשר $S_n = \sum_{i=0}^{n} a_i$ היא סדרת הסכומים החלקיים של סדרת פיבונאצ'י. עבור כל קורפוס שמכיל תדירויות שהן n מספרי פיבונאצ'י הראשונים, עץ ההאפמן ייבנה כך שבכל איטרציה $0 \le k \le n-2$ בבניית העץ, הצמתים שייבחרו יהיו תת עץ שמכיל כעלים את האיברים עם התדירויות $a_0, \dots, a_k$, ולכן בעדיפות S_k, והצומת (הבודד) שעדיפותו היא a_{k+1}. באיטרציה בה $k = n-1$ נשארנו עם העץ הסופי – עץ האפמן. הטענה הממוסגרת והטענה שסדרת פיבונאצ'י עולה ממש החל מהאיבר השני, מבטיחות לנו שאף פעם אין חופש בבחירת שני הצמתים עם העדיפות המינימלית, אלא רק בסדר הבנים. ואכן ניתן לראות מסעיף א', שהחל מהאיבר השלישי בסדרת פיבונאצ'י (המיוצג בקורפוס ע"י c), כל מספר פיבונאצ'י עוקב חדש שנוסף לקורפוס, מוסיף את הביט 1 משמאל, לאיברים הקודמים (למעשה יש לנו חופש בבחירת סדר הבנים בבניית עץ האפמן, אך לשם הנוחות נניח שכל איבר חדש שנכנס לעץ, נכנס כבן השמאלי). סה"כ **ההכללה**: עבור קורפוס עם n מספרי פיבונאצ'י הראשונים, הצומת עם התדירות שהיא האיבר $2 \le k \le n-1$ בסדרת פיבונאצ'י תקודד למחרוזת $'0' + (n-k-1) * '1'$, והצמתים עם התדירויות שהן האיברים $k = 0$ ו-$k = 1$ בסדרת פיבונאצ'י יקודדו בדומה לסעיף א': $'0' + (n-2) * '1'$ ו- $'1' * (n-1)$ בהתאמה.

סעיף ג'

<u>טענה</u>: ההפרש הוא 0.

<u>הוכחה</u>: לבניית עץ האפמן של קובץ זה נבחר תחילה כרגיל את שני התווים בעלי התדירויות המינימליות: a_1 ו-a_2. כעת הן מיוצגות במילון העדיפות עם משקל $a_1 + a_2$. מהנתונים מתקיים $a_1 + a_2 > 2a_1 > a_n$, ולכן הצומת שמורכב מ-a_1 ו-a_2, (a_1, a_2), לא ייבחר כמינימום לפני כל השאר $a_3, a_4, \dots, a_{256}$. לאחר מכן נבחר את a_3 ו-a_4 ומאותה סיבה, נמשיך לבחור את כל שאר האיברים כזוגות עוקבים עד לזוג a_{255} ו-a_{256}. קיבלנו 128 צמתים שמייצגים עצים בינאריים מלאים, ובבירור יחס הסדר נשמר בין העדיפויות (העדיפות של (a_9, a_{10}) קטנה יותר משל (a_{11}, a_{12}) למשל). לכן נאחד את (a_1, a_2) ו-(a_3, a_4), לכדי $\big((a_1, a_2), (a_3, a_4)\big)$ שמשקלו $a_1 + a_2 + a_3 + a_4$.

נוכיח באינדוקציה על k את טענת העזר הבאה: $\boxed{\forall k \in \mathbb{N}.\ \left(1 \le k \le \frac{n}{2}\right) \longrightarrow \left(2\sum_{i=1}^{k} a_i > \sum_{i=n-k+1}^{n} a_i\right)}$, כאשר $n = 256$.

<u>בסיס</u> - עבור $k = 1$: $2a_1 > a_n$ וזה נכון עפ"י הנתון.

<u>צעד</u> - יהי $1 \le k \le \frac{n}{2} - 1$ טבעי, ונניח שמתקיים $2\sum_{i=1}^{k} a_i > \sum_{i=n-k+1}^{n} a_i$. צ"ל $2\sum_{i=1}^{k+1} a_i > \sum_{i=n-k}^{n} a_i$.

$$2\sum_{i=1}^{k+1} a_i = 2a_{k+1} + 2\sum_{i=1}^{k} a_i \underset{\begin{pmatrix}\text{סדרה}\\ \text{מונוטונית}\\ \text{עולה ממש}\end{pmatrix}}{>} 2a_1 + 2\sum_{i=1}^{k} a_i \underset{(\text{מהנתון})}{>} a_n + 2\sum_{i=1}^{k} a_i > a_{n-k} + 2\sum_{i=1}^{k} a_i \underset{(\text{ה"א})}{>}$$

$$> a_{n-k} + \sum_{i=n-k+1}^{n} a_i = \sum_{i=n-k}^{n} a_i$$

לכן, אמנם הסדר נשמר, אך $a_1 + a_2 + a_3 + a_4 > 2(a_1 + a_2) > a_{255} + a_{256}$ ולכן הצומת שמורכב מ-(a_1, a_2) ו-(a_3, a_4), $\big((a_1, a_2), (a_3, a_4)\big)$ לא ייבחר כמינימום לפני כל השאר $(a_5, a_6)\ ,\ \dots\ ,\ (a_{255}, a_{256})$. כך ממשיכים (מאחדים בזוגות בכל שלב את כל הצמתים לפני שמאחדים את זוגות הצמתים) ולפי טענת העזר לא יתבצע איחוד של צמתים המייצגים עצים שאין בהם אותו מספר עלים, ומכיוון ש-256 חזקה שלמה של 2 נקבל בפרט בסוף עץ בינארי מלא ובו כל העלים באותה רמה. לכן ההפרש בין קידודי התווים הוא אפס בין כל זוג תווים, בפרט בין q ו-p.

<u>סעיף ד'</u>

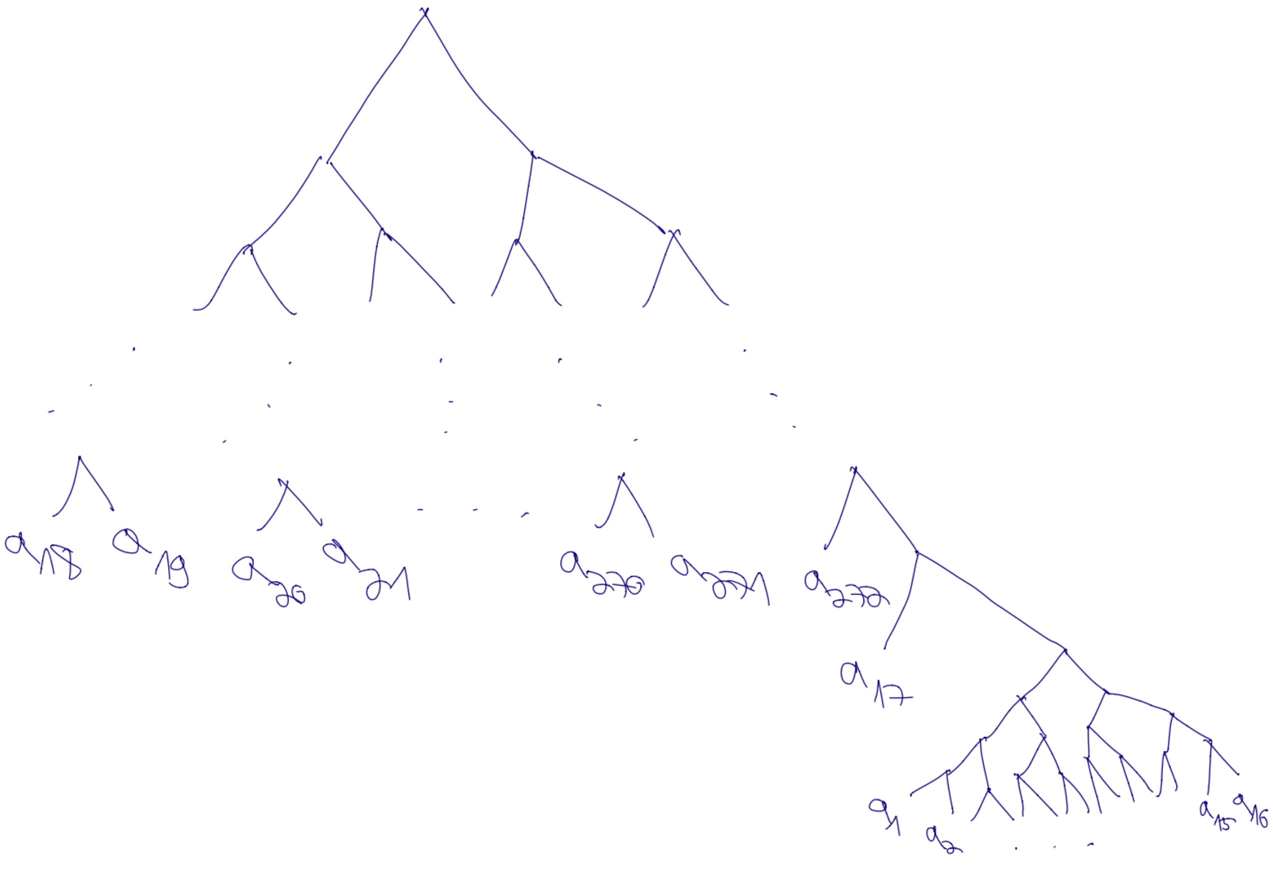
<u>טענה</u> : ההפרש הוא 1.

הפעם 300 אינו חזקה שלמה של 2. תהליך בניית העץ יתרחש כמו מקודם, אך עד שיתקבלו 75 תתי-עצים של 4 עלים. כעת ייבנו תתי עצים של 8 עלים אך בסוף השלב הזה יישאר תת העץ של 4 עלים של a_{297} עד a_{300} (כי המשקל שלו הכי גדול שלא יהיה לו בן זוג. לכן הוא יתאחד עם תת העץ של a_1 עד a_8 (תתי העצים האלה שמתאחדים עכשיו, הם עם המשקל המינימלי ברגע האיחוד). בפרט, זה מספיק לנו לקביעת ההפרש בין העומק של a_1 לעומק של a_{300} בעץ האפמן (העץ הסופי), שמהווה את ההפרש בין הקידודים שלהם. העומק של a_{300} **ביחס לנקודת החיבור** עליה דיברנו הוא 3 (כי הגיע מעץ בינארי מלא עם 4 עלים (2) + האורך של החיבור (1)), ואילו העומק של a_1 ביחס לנקודת החיבור עליה דיברנו הוא 4 (כי הגיע מעץ בינארי מלא עם 8 עלים (3) + האורך של החיבור (1)). לכן **ההפרש הוא 1**.

<u>סעיף ה'</u>

ראשית, מנתון המונוטוניות ממש ונתון a, מתקבל מקרה דומה למקרה מסעיף ג', עבור $n = 16$. התדירויות a_1 עד a_{16} הן הקטנות ביותר, אבל עדיין לא ניתן להסיק שייבנה תחילה עץ בינארי מלא עבורם. לשם כך ניעזר בנתון c. נשים לב שמתקיים $\sum_{i=1}^{16} a_i < 16a_{16} < a_{17}$, ולכן אפילו המשקל הכולל של העץ שמורכב מ-a_1 עד a_{16} עדיין מינימלי מבין כל אחד מהאיברים במילון השכיחויות המתעדכן (שממומש תחילה כהעתק מילון השכיחויות של הקורפוס), ולכן בבירור העץ הזה (שמהווה תת עץ של עץ האפמן הסופי) ייבנה תחילה. לאחר מכן, האיברים עם התדירויות המינימליות במילון המתעדכן הם העץ הזה (כאמור) ו-a_{17} לכן נאחד אותם. כעת לא ניתן לדעת בוודאות מי הם שני האיברים המינימליים, כי אנחנו לא יודעים את היחס בין עדיפות העץ שכולל את a_1 עד a_{17}, $\sum_{i=1}^{17} a_i$, לבין שאר הצמתים, a_{18} עד a_{272} (שהם בודדים כרגע). מה שכן ידוע הוא שמתקיים $\sum_{i=1}^{17} a_i < 16a_{16} + a_{17} < 2a_{17} < a_{18} + a_{19}$, ולכן העדיפות של העץ הזה קטנה יותר מכל עדיפות של זוג צמתים מאלה שנותרו (a_{18} עד a_{272}), ובנוסף מתקיים ממונוטוניות ומהנתון: $\sum_{i=1}^{18} a_i > a_{17} + a_{18} > 2a_{17} > a_{272}$, כלומר אם איחדנו את העץ הזה (a_1 עד a_{17}) עם איזשהו איבר, ולו הכי קטן מאלה שנותרנו, a_{18}, העץ שיתקבל לא יהיה המינימלי יותר עד שנצוות בזוגות את כל השאר. לכן ניתן לצפות מהעץ הזה (a_1 עד a_{17}) שיתנהג כמו איזשהו צומת נניח בה"כ a_{17} מבין האיברים a_{17} עד a_{272}. יש בטווח הזה 256 איברים, וכן a_{17} מקיים את נתון b, ולכן באופן דומה לסעיף ג', יתקבל עץ בינארי מלא עם 7 עלים + העץ a_1 עד a_{17} שיושב במקום העלה השמיני (סדר העלים אינו ידוע כי חסרים נתונים, אבל זה לא משנה לשאלה הספציפית פה). לכן בהכרח עומק העלה a_{272} הוא 8, ואילו עומק העלה a_1 הוא 13 (העץ a_1 עד a_{17} יושב בעומק (8) + העץ a_1 עד a_{16} מהווה את אחד הבנים של העלה שהוא בעצם העץ a_1 עד a_{17} (1) + עומק העץ a_1 עד a_{16} הוא 4 (4)). לכן **ההפרש הוא 5**.

סקיצה, עבור המקרה הקיצוני בו $\sum_{i=1}^{17} a_i > a_{272}$ (ולכן יתאחד איתו):



שאלה 3

סעיף א'

דוגמה: $s='abcd'$. הפלט שיתקבל בשתי ההרצות: $['a','b','c','d']$.

סעיף ב'

הטענה נכונה. דוגמה: $s='aaabaaaaa'$.
הפלט של $LZW\_compress$: $['a','a','a','b',[4,3],'a','a']$. אורך הקידוד: $8*6+18=66$.
הפלט של $LZW\_compress\_new$: $['a','a','a','b','a',[1,4]]$. אורך הקידוד: $8*5+18=58$.
$(58<66)$

סעיף ג'

הטענה אינה נכונה. הפונקציה $LZW\_compress$ היא תאוותנית ($greedy$): כל חזרה באורך 3 ומעלה (שנמצאה ע"י הפונקציה $maxmatch$) נלקחת בחשבון מיד ונכנסת לייצוג הביניים שלנו. לעיתים זה לא אופטימלי: ייתכן שיש לנו חזרה $[m_1,k_1]$ באיזשהו אינדקס p וחזרה אחרת עם $[m_2,k_2]$ באינדקס $p+1$ או $p+2$, למשל, כאשר $k_1 \ll k_2$, אבל בגלל שהיא תאוותנית, הפונקציה $LZW\_compress$ תפספס את החזרה $[m_2,k_2]$ (החסכנית יותר) ותממש את החזרה $[m_1,k_1]$. הפונקציה $LZW\_compress\_new$, לעומת זאת, הממומשת באופן רקורסיבי, פועלת שונה כאשר נמצאה חזרה עם $k\geq 3$ עבור איזושהי תת-מחרוזת (עם אותה סיפא) של המחרוזת המקורית. היא מחשבת (רקורסיבית) שתי אופציות: ייצוג הביניים של תת-המחרוזת הנוכחית, כאשר התו הראשון מיוצג כתו בודד, וייצוג הביניים של תת-המחרוזת הנוכחית כאשר החזרה שנמצאה ממומשת, ולוקחת בחשבון (מחזירה) את האפשרות המשתלמת ביותר מבניהן (מבחינת מספר הביטים של רצף הביטים שהתקבל מהפונקציה $inter\_to\_bin$). לכן היא בעצם מפעילה את $maxmatch$ על כל תו במחרוזת, לעומת המקורית שמדלגת על רצף התווים באורך החזרה וכך עשויה לפספס שם תו שעבורו קיימת חזרה יותר טובה, ולכן היא עשויה להניב דחיסה משופרת (אם כי עשויה לעלות יותר זמן), ואכן קיימים קלטים כאלה כפי שראינו (סעיף ב'). לכן גם לא קיימת מחרוזת שהפונקציה $LZW\_compress$ תידחס טוב יותר מ-$LZW\_compress\_new$ כי האחרונה לוקחת בחשבון גם את האפשרויות ש-$LZW\_compress$ הייתה מממשת באופן מידי, ובוחרת מבין אפשרויות נוספות את האפשרות שתיתן את הדחיסה היעילה יותר.

שאלה 4

חלק ראשון

סעיף א'

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

סעיף ב'

טענה: $\boxed{d=4}$.

חסם עליון: למשל המילה $w_1=(0,0,0,0,0,0,0)$ המתקבלת מההודעה $(0,0,0)$ והמילה $w_2=(0,0,1,0,1,1,1)$ המתקבלת מההודעה $(0,0,1)$. הן נבדלות באינדקסים: 2, 4, 5 ו-6 ולכן מרחק ההמינג שלהן הוא 4.

חסם תחתון: תהיינה שתי הודעות $y\neq y'$. נחלק לשלושה מקרים זרים ומשלימים:

- אם $\Delta(y,y')=1$, נניח בה"כ ש-x_1 הוא הביט בו הן נבדלות. אזי מילות הקוד שלהן נבדלות בביט x_1, בביט x_1+x_2 (הסכומים x_1+x_2 מודולו 2 של שניהם שונים, כי סכום אחד זוגי והשני אי-זוגי עקב מספר אי-זוגי של 1-ים באחד ומספר זוגי של 1-ים בשני), בביט x_1+x_3 (מאותה סיבה), ובביט $x_1+x_2+x_3$ (מאותה סיבה).
- אם $\Delta(y,y')=2$, נניח בה"כ ש-x_1 ו-x_2 הם הביטים בהם הן נבדלות. אזי מילות הקוד שלהן נבדלות בביט x_1, בביט x_2, בביט x_1+x_3 (מהסיבה שלעיל) ובביט x_2+x_3 (מאותה סיבה).
- אם $\Delta(y,y')=3$, מילות הקוד שונות בכל שלושת הביטים, x_1, x_2 ו-x_3. לכן בשלשה אחת כזאת יש מספר זוגי של 1-ים ובשנייה יש מספר אי-זוגי, ולכן גם הביט $x_1+x_2+x_3$ שונה בשניהם (מאותה סיבה שלעיל). סה"כ בכל מקרה אפשרי, מילות הקוד נבדלות בלכל הפחות 4 ביטים שונים בין מילות הקוד.

סעיף ג'

הטענה נכונה. דוגמה: $y=(0,1,0,1,0,0,0)$, $w_1=(0,1,1,1,1,0,0)$, $w_2=(0,1,0,1,0,1,1)$ (המרחק של שתיהן מ-y הוא 2, וניתן להשוות עם כל שמונה מילות הקוד ולראות שזה אכן המרחק המינימלי).

חלק שני

bad_coding הוא קוד מטיפוס $[n=12,\ k=2,\ d=8]$ עבור $|x|=2$, ו-$[n=4\cdot(|x|+1),\ k=|x|,\ d=4]$ עבור $|x|>2$ (עבור $0\le|x|<2$ הקוד לא מוגדר (תיזרק שגיאת זמן ריצה $IndexError$)).

שאלה 5

סעיף א'

```
def fill_cell(table, i, j, rule_dict):
    for k in range(i+1, j): # non trivial partitions of s[i:j]      (j-i-1 איטרציות)
        for lhs in rule_dict: # lhs is a single variable            (O(|R|) איטרציות)
            for rhs in rule_dict[lhs]:                              (O(1) איטרציות)
                if len(rhs) == 2: # rule like A -> XY (not A -> a)  O(1)
                    X, Y = rhs[0], rhs[1]  O(1)
                    if X in table[i][k] and Y in table[k][j]:  O(1) בממוצע | O(|R|) גרוע
                        table[i][j].add(lhs)  O(1) בממוצע
                                                                    (c(j-i-1)·|R| איטרציות)
```

עבור מחרוזת באורך m ישנן $m-1$ חלוקות לא טריוויאליות, כי שאלה שקולה קומבינטורית היא בכמה מקומות ניתן לשים חוצץ במחרוזת כזאת, ובבירור ניתן בין כל שני תווים במחרוזת, ויש $m-1$ כאלה. אצלנו מחרוזת בטבלה מיוצגת כ-$st[i:j]$, ולכן עבור i ו-j שניתנות כקלט לפונקציה $fill\_cell$, ישנן $j-i-1$ חלוקות לא טריוויאליות. כל חלוקה כזאת משרה k (או הפוך) באופן חח״ע. לכן במקום n איטרציות של k (הנחה לא הדוקה), נניח שהוא רץ $j-i-1$ איטרציות (הנחה הדוקה). בהנחה ששאר התנאים לא משתנים ולפי הניתוח מהתרגול, העבודה של $fill\_cell$ היא $W_{fill\_cell}=c\cdot(j-i-1)\cdot|R|$ בממוצע. נתבונן במקטע הקוד בו נעשה השינוי:

```
    # Fill the table cells representing substrings of length >=2
    for length in range(2, n+1):                 (n-1 איטרציות)
        for i in range(0, n-length+1):           (n-length+1 איטרציות)
            j = i+length  O(1)
            fill_cell(table, i,j, rule_dict)  c·(j-i-1)·|R| בממוצע
                                                 (O(n²) איטרציות)
```

עבור כל ערך של $lenth$, שמייצג אלכסון בטבלה, i רץ $n-length+1$ איטרציות. ועבור כל איטרציה של i, j נקבע באופן חח״ע ונשים לב שהעבודה של $fill\_cell$ היא $W_{fill\_cell}=c\cdot(j-i-1)\cdot|R|=c\cdot(length-1)\cdot|R|$, כלומר לא תלויה בכלל ב-$i$ ו-j (הגיוני, כי כל אלכסון קובע מספר קבוע של חלוקות). לכן, איטרציה פנימית עולה $(n-length+1)\cdot W_{fill\_cell}$. לשם נוחות נקרא ל-$length$ בשם l, ונסכום עליו:

$$\sum_{l=2}^{n}\big((n-l+1)\cdot c\cdot(l-1)\cdot|R|\big)=c|R|\cdot\left[(n+2)\cdot\sum_{l=2}^{n}l-(n+1)\cdot\sum_{l=2}^{n}1-\sum_{l=2}^{n}l^2\right]=$$

$$\underbrace{=}_{\text{(סכום ריבועים וסדרה חשבונית)}}\quad c|R|\cdot\left[(n+2)\cdot\frac{(n+2)\cdot(n-1)}{2}-(n+1)(n-1)-\frac{2n^3+3n^2+n-6}{6}\right]=\frac{c}{6}|R|(n^3-n)$$

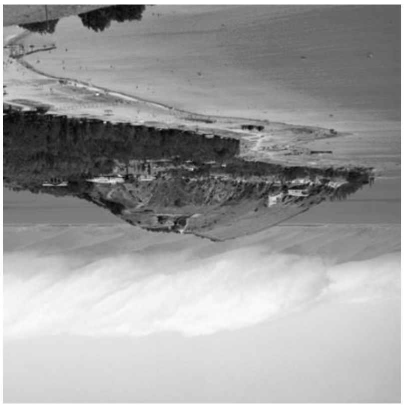
בבירור $\frac{c}{6}|R|(n^3-n)=O(n^3\cdot|R|)$, שכן $\frac{c}{6}|R|(n^3-n)\le\frac{c}{6}|R|\cdot n^3$ (הקבוע הוא $\frac{c}{6}$ וזה נכון לכל $n,|R|\ge0$). גם בכיוון השני $n^3\cdot|R|=O\left(\frac{c}{6}|R|(n^3-n)\right)$, שכן $n^3\cdot|R|\le2|R|\cdot n^3-2|R|\cdot n=\frac{12}{c}\cdot\frac{c}{6}|R|(n^3-n)$ (הקבוע פה הוא $\frac{12}{c}$ וזה נכון לכל $n\ge1$ ו-$|R|\ge0$). לכן העבודה פה היא $\Theta(n^3\cdot|R|)$. וזה בדיוק כמו מה שקיבלנו בכיתה בחלק זה (במונחי O עם ההנחה הלא הדוקה). בשאר חלקי האלגוריתם שניתחנו בתרגול ההנחות שלנו היו הדוקות, ולכן נסיק שסיבוכיות האלגוריתם במקרה הממוצע היא $\boldsymbol{\Theta(n^3\cdot|R|)}$ ולא רק $O(n^3\cdot|R|)$.

שאלה 6

סעיף ב׳

התמונה המקורית[1]:

התמונות המתקבלות:



תמונה 1: לפני השינוי

תמונה 2: אחרי השינוי

בתמונה 1 ניתן לראות כי התמונה אכן עברה היפוך בציר האנכי כרצוי, ואילו תמונה 2 לא התהפכה ביחס למקורית. זאת משום שתמונה 1 התקבלה מהפונקציה לפני השינוי. בפונקציה זו יצרנו העתק של התמונה המקורית ובמטריצה של התמונה המועתקת עברנו על כל העמודות, כאשר בכל עמודה עברנו על כל הפיקסלים מלמעלה למטה, ועבור כל פיקסל כזה הפעלנו את פונקציית הלמבדא שבשאלה שמחשבת את ערך הפיקסל (בטווח 0-255) של הפיקסל הנגדי $(x, h-y-1)$ (זה שנמצא במרחק זהה מקצה אותה עמודה אבל מהקצה השני (עבור מספר אי-זוגי של שורות, הפיקסל הנגדי של האמצעי הוא זה שבאמצע)) שנמצא במטריצה שבתמונה המקורית. תמונה 2 לעומת זאת, התקבלה מהפונקציה לאחר השינוי. בפונקציה זו לא הפעלנו את פונקציית הלמבדא על הפיקסלים באיזושהי תמונת העתק, אלא על הפיקסלים שבמטריצה של התמונה המקורית. לכן בכל עמודה במעבר שלנו, מהרגע שהגענו לפיקסל בשורה שבאינדקס $h//2+1$ (השורה שאחרי ה״אמצע״), בעצם ערכי הפיקסלים שהתקבלו מפונקציית הלמבדא אינן של המחצית העליונה של אותה עמודה מהתמונה המקורית, כי אם של המחצית העליונה של העמודה של התמונה שביצענו עליה שינוי בריצה זו (שהיא קיבלה את הערכים של המחצית התחתונה של העמודה בתמונה במקורית במן סימטריה ביחס לאמצע האופקי של התמונה). לכן לכל עמודה x ערך הפיקסל באיזושהי שורה $h//2+1 \le y \le h-1$ הוא של השורה $h-y-1$, כלומר נקבל $mat[x, h-(h-y-1)-1] = mat[x,y]$. לכן קיבלנו בטווח שורות זה בכל עמודה את אותם ערכי פיקסלים כמו של התמונה המקורית וזו הסיבה לצורה של תמונה 2 (בחצי העליון סימטריה בציר האנכי ובתחתון סימטריה כפולה (ולכן אין שינוי)).

[1] מקור: https://armeniadiscovery.com/en/articles/armenian-landscape