תרגיל בית מס' 5 במבוא מורחב למדעי המחשב – 315328963

שאלה 1

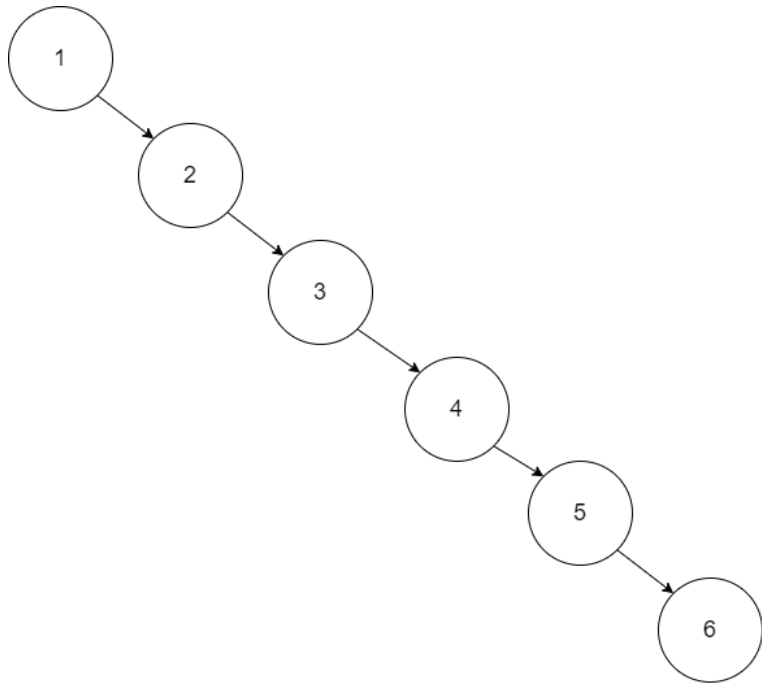
ו. סיבוכיות שמן הריצה של הרכבת הרשימה במונחים אסימפטוטיים היא O(nk). נשים לב כי הפונקציה מבצעת באופן רקורסיבי, הרכבה של שתי הפרמוטציות האחרונות ברשימה (קרי, בשני האינדקסים הימניים ביותר). מעבר לכך, היא מבצעת פעולות בסיבוכיות זמן קבועה. ההרכבה מתבצעת באמצעות הפונקציה compose. מכיוון שיש k איברים, יש צורך ב-k-1 הרכבות, ולכן בסך הכל הפונקציה compose מופעלת k-1 פעמים.

הפונקציה compose נפתחת עם 2 פעולות בסיבוכיות זמן קבועה (יצירת רשימה חדשה ריקה, קבלת אורך של הפרמוטציה הראשונה) ולאחר מכן מופעלת לולאה באורך n. בכל איטרציה של הלולאה נעשות פעולות בסיבוכיות זמן קבועה (הוספה לרשימה, השמה של איבר) ולכן מהלולאה נובעת סיבוכיות של O(n). לאחר מכן הפונקציה מחזירה פרמוטציה של הרשימה שנוצרת, כלומר עוד פעולה בסיבוכיות של O(n) (כפי שנאמר בסעיף א' בין היתר). בסך הכל הסיבוכיות של הפעלת compose היא O(2n)=O(n).

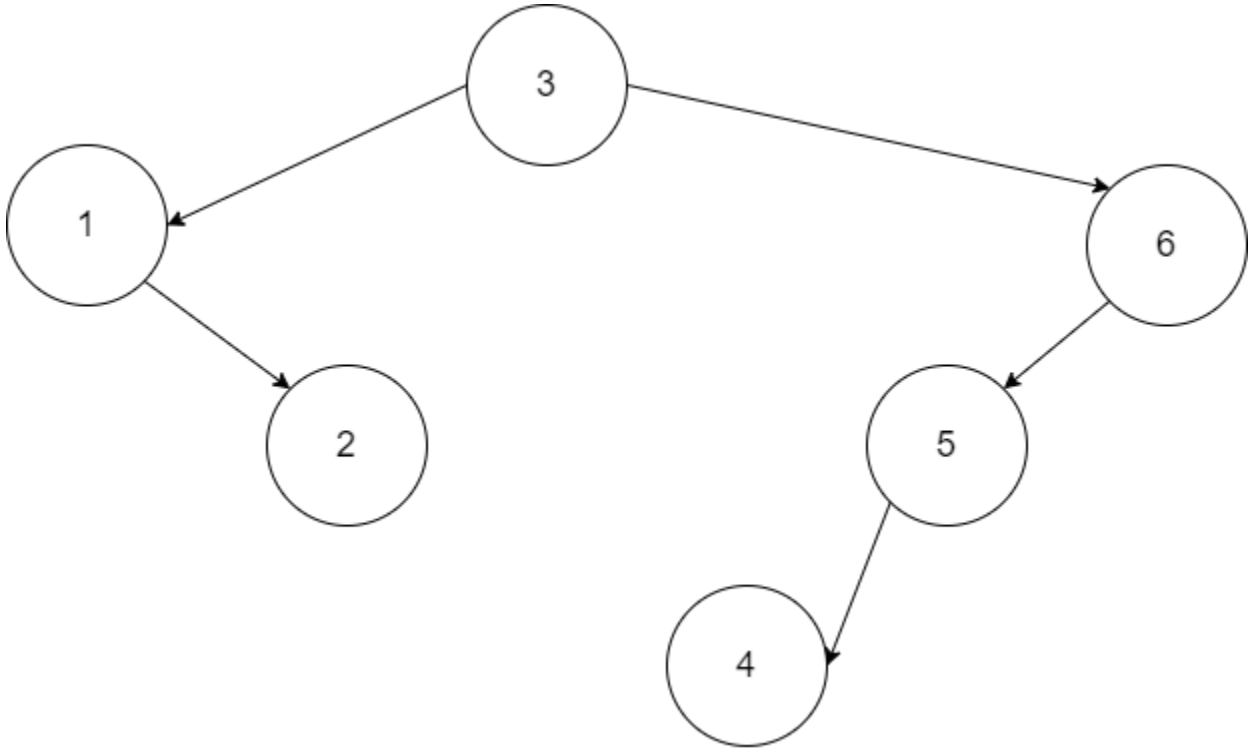
אנו מפעילים פונקציה זאת k-1 פעמים, לכן הסיבוכיות של הרכבת הרשימה הכוללת היא $O\big(n \cdot (k-1)\big) = O(nk)$.

שאלה 3

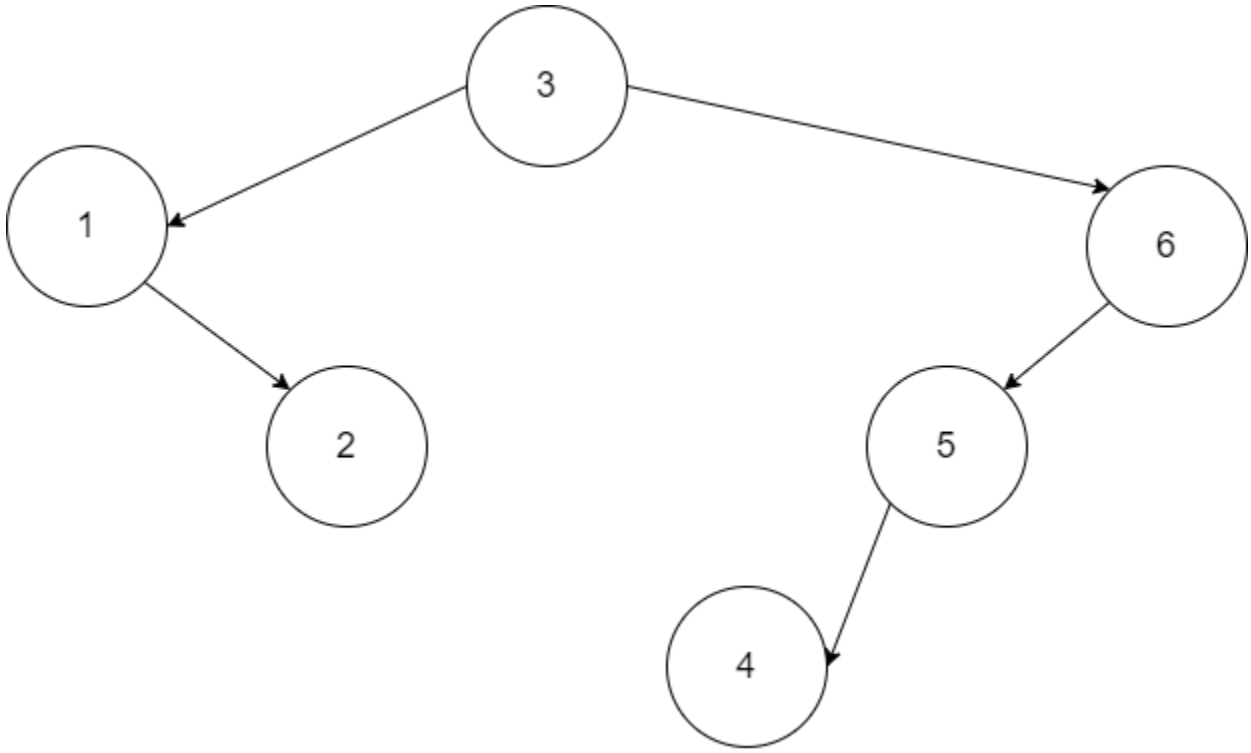
א. lst1 = [1,2,3,4,5,6]:



lst2 = [3,1,2,6,5,4]:



ב. הרשימות הן [3,1,2,6,5,4] ו-[3,6,5,4,1,2] ושתיהן יוצרות את העץ הבא:



ד. סיבוכיות זמן הריצה במקרה הגרוע היא $O(n^2)$. ראשית, הפונקציה בודקת את האיברים הראשונים של הרשימות ומשווה אותם – סיבוכיות זמן קבועה.

לאחר מכן, היא מפעילה על כל אחת מהרשימות את פונקציית העזר הרקורסיבית help_same (החל מהאיבר השני של הרשימה). פונק' זאת מחלקת את האיברים בסופו של דבר לשתי רשימות – האחת של מספרים הגדולים מהראשון ברשימה המקורית, והשנייה של הקטנים ממנו. הפונ' מבצעת הוספה לרשימה בכל איטרציה (מלבד הסופית, בה היא מחזירה את הרשימות), פעולה בסיבוכיות זמן קבועה. עם זאת, בכל איטרציה (ויש n-1 כאלה לפי החוקיות), נעשית פעולה של סלייסינג על הרשימה, שעלותה כגודל הרשימה שנותרה. בתחילה מדובר ברשימה בגודל n-1 (בקריאה לפונקציה הרקורסיבית), אח"כ בגודל n-2 וכן הלאה עד 0 (רשימה ריקה). אם כן זהו סכום סדרה חשבונית, ולכן בסה"כ הפעלת הפונקציה גוררת סיבוכיות של $O(n^2)$.

את פונקציית העזר אנחנו מפעילים על שתי הרשימות, ולאחר מכן משווים בין איברי תתי-הרשימות (רשימות בהן איברים קטנים מהאיבר הראשון, וכאלה שבהם גדולים). מדובר בסיבוכיות $O(n)$ במקרה הגרוע (למשל שיש רק איברים קטנים מהראשון).

נסכום את הסיבוכיות ונראה שאכן הסיבוכיות הכללית של הפעלת הפונקציה same_tree היא $O(n^2)$.

ה. לא ניתן לשפר את סיבוכיות זמן הריצה בעזרת ממואיזציה, וזאת משום שפונקציית העזר הרקורסיבית help_same מוסיפה איברים מרשימת הקלט לשתי רשימות פלט באופן בלתי תלוי באיברים הקודמים, אלא רק באיבר הראשון של הפונקציה (שנרשם כמפתח בקלט). אם כן, אין תלות בין הקריאות הרקורסיביות השונות ולכן גם אין חזרה על קריאות שונות ופעולות דומות, ולכן ממואיזציה לא תשפר את זמן הריצה.

שאלה 5

א. נכתוב את a כסכום הבא: $a = \left\lfloor \frac{a}{b} \right\rfloor \cdot b + a \bmod b$. כלומר a שווה למספר השלמים שבו b נכנס בו כפול b (כלומר כל מה שהוא לא שארית החלוקה ב-b), ועוד שארית החלוקה. לצורך נוחות נסמן $k = \left\lfloor \frac{a}{b} \right\rfloor \cdot b, r = a \bmod b$. כלומר $a = k + r$

כעת נסתכל על הביטוי שרשום מימין: $a^c \bmod b$. נציב את הדרך השנייה לכתיבה של a:

$$(k + r)^c \bmod b$$

לפי נוסחת הבינום:

$$(k + r)^c \bmod b = (r^c + c \cdot kr^{c-1} + \cdots + c \cdot k^{c-1}r + k^c) \bmod b \qquad (1)$$

נשים לב כי מתקיים: $\mathrm{k \bmod b} = \left\lfloor \frac{a}{b} \right\rfloor \cdot \mathrm{b \bmod b} = 0$, וזאת משום ש-k הוא ביטוי שאחד מגורמיו הוא b, לכן שארית החלוקה שלו במספר זה תהיה אפס.

כלומר, עבור כל איבר בסוגריים בביטוי שקיבלנו ב-(1), שכולל את k, תוצאת שארית החלוקה ב-b תהיה 0. אז נישאר עם הביטוי הבא:

$$r^c \bmod b = (a \bmod b)^c \bmod b$$

ובסה"כ קיבלנו $a^c \bmod b = (a \bmod b)^c \bmod b$, כנדרש.

ב. נשים לב, כי לפי סעיף א' מתקיים:

$$(g^a \bmod p)^b \bmod p = g^{ab} \bmod p \qquad (1)$$

באופן דומה עבור a′:

$$\left(g^{a'} \bmod p\right)^b \bmod p = g^{a'b} \bmod p \qquad (2)$$

לפי הנתון מתקיים: $g^{a'} \bmod p = g^a \bmod p$, לכן השוויון יישמר גם אם נעלה בחזקה ונבצע פעולת שארית חלוקה ב-p:

$$(g^{a'} \bmod p)^b \bmod p = (g^a \bmod p)^b \bmod p$$

ממשוואות (1) ו-(2) (טרנזיטיביות) נקבל בסה"כ:

$$g^{ab} \bmod p = g^{a'b} \bmod p \qquad (3)$$

ובכדי להגיע ל-$g^{a'b} \bmod p$, נוכל לקחת את $g^b \bmod p$ שנתון לנו, להעלות אותו בחזקת a′ ולבצע שארית חלוקה ב-p (שניהם נתונים לנו). ניעזר שוב בסעיף א':

$(g^b \bmod p)^{a'} \bmod p = g^{a'b} \bmod p$, כלומר לפי משוואה (3) קיבלנו את $g^{ab} \bmod p$.

שאלה 6

ב. ראשית, נשים לב שהמקרה הגרוע יתקבל כאשר המחרוזות זהות, ואז ההשוואה ביניהן לא 'תיפסק' לפני שיעברו על כל אורך המחרוזות O(k).
אם כן, סיבוכיות הזמן של הפתרון במקרה הגרוע תהיה $O(n^2k)$. נשים לב כי עבור כל מחרוזת ברשימה, עלינו לבדוק את הרישא שלה, כלומר עלינו לעבור על n רישות (O(n)). עבור כל רישא, אנו בודקים את הסיפא של כל שאר המחרוזות, n-1 כאלה (שמנו תנאי על מנת לא לבדוק רישא וסיפא של אותה המחרוזת), כלומר O(n). אם כן, עבור כל מחרוזת אנו מוציאים את הרישא שלה והסיפא של השנייה, שתיהן באורך *k*, ולכן מדובר בסה"כ ב-O(k), ואז מבצעים את ההשוואה, שכפי שנאמר בשאלה, מתבצעת גם בזמן של O(k) (בייחוד במקרה הגרוע בו אנו עוברים על איברי כל הרשימה). אם, כמו במקרה הגרוע, אכן יש שוויון בין קטעי המחרוזות, נוסיף את האינדקסים של הרשימות לרשימה המאגדת – פעולה בסיבוכיות זמן קבועה. כלומר בסה"כ מתבצעות בדיקות בסיבוכיות זמן של O(k), והן נעשות $O(n^2)$ פעמים (כפי שתואר לעיל, לכל המחרוזות והאיברים הרלוונטיים). אז הסיבוכיות במקרה הגרוע תהיה $O(n^2k)$.

ה. הסיבוכיות של הפתרון בממוצע היא O(nk). ראשית נפרט את הסיבוכיות של הפונקציות בהן נעשה שימוש במסגרת הפתרון:

אתחול מילון חדש – באתחול של מילון נוצרת רשימה באורך m (לפי פרמטר שאנו בוחרים), ולכן הסיבוכיות של פעולה זאת היא O(m). נסביר בהמשך כיצד נבחר את m

כיצד נבחר את m? בפועל, m מייצג את מספר הרשימות בטבלה, אליהן מסווגים כל הערכים. אם נבחר את m בגודל של מספר האיברים במילון (n), אז בממוצע כל אחת מרשימות בטבלה יכילו איבר אחד בלבד. בכדי להקטין את הסיכוי להתנגשויות נבחר את m כפי שלוש ממספר האיברים במילון (3n)

הכנסה של פעולה למילון (insert) – בפעולה זאת ישנו חישוב בעזרת פונקציית ההאש של פייתון, כלומר אם הקלט שאנו מכניסים הוא באורך k, אז החישוב יעלה O(k). מעבר לכך, שאר הפעולות הן בסיבוכיות זמן קבועה (הוספה לרשימה, השמה, חישוב אריתמטי) ולכן בסה"כ סיבוכיות הזמן של פעולה זאת היא O(k)

חיפוש לפי מפתח במילון (find) – בפעולה זאת גם כן זאת ישנו חישוב בעזרת פונקציית ההאש של פייתון, כלומר אם הקלט שאנו מכניסים הוא באורך k, אז החישוב יעלה O(k). אחרי זה מתבצע מעבר על הרשימה בתוך המיקום בטבלה (באינדקס אותו חישבנו). **במקרה הממוצע** יהיו ברשימה זאת $\frac{n}{m}$ איברים, ובכל מעבר שכזה אנו מבצעים (או שלא) פעולות בסיבוכיות זמן קבועה – הוספה לרשימה אחרת. לפי הבחירה שלנו של m (ראו למעלה), נקבל כי בממוצע יהיה איבר אחד ברשימה. לכן, בסה"כ הסיבוכיות של find תהיה לכל איבר ברשימה O(k), ומכיוון שבממוצע יש אחד כזה, הסיבוכיות של כל החיפוש תהיה O(k)

כעת נסתכל על הפתרון שהצענו.

ראשית, ישנו אתחול של המילון, שהוא בסיבוכיות $O(m) = O(n)$.

לאחר מכן, ישנה לולאה שרצה על כל איבר ברשימת הקלט (ישנם n איברים כאלה), ועבור כל אחד מהם, מכניסה את הרישא שלו (ע"י סלייסינג באורך k, O(k)) למילון יחד עם האינדקס של האיבר. פעולת ההכנסה כפי שאמרנו היא גם כן O(k), כלומר לכל איבר ברשימת הקלט נעשות פעולות בסיבוכיות של O(k) בסה"כ. (ישנם n איברים כאלה), ולכן בסה"כ הסיבוכיות של הלולאה היא O(nk)

הלולאה השנייה רצה גם כן על כל כל איבר ברשימת הקלט (ישנם n איברים כאלה), ועבור כל אחד מהם, לוקחת את הסיפא שלו (ע"י סלייסינג באורך k, O(k)) ובודקת האם היא קיימת במילון, כפי שאמרנו פעולה בסיבוכיות O(k). מכיוון שיש בממוצע איבר אחד בכל רשימה בטבלה, אז שאר הפעולות בלולאה תהיינה בסיבוכיות זמן קבועה (ישנה לולאה באורך קבוע, שנובע מהתוצאה של פעולת החיפוש, ובעקבותיה הוספה לרשימת הפלט של האיבר ברשימת הקלט ושל ה-מאץ' שלו). כלומר בסה"כ מדובר בפעולה בסיבוכיות O(k) לכל איבר ברשימת הקלט, ובסה"כ בסיבוכיות O(nk)

אם כן, נסכום את כל מה שקיבלנו לעיל ונראה כי הסיבוכיות הכללית של הפתרון היא O(nk)

ז. מבדיקת הזמנים נשים לב כי הפונקציה הראשונה שכתבנו (ללא המילון) היא האיטית ביותר, אחריה הפונקציה השנייה (עם המחלקה *Dict* שלנו) ולבסוף הפונקציה השלישית (עם המחלקה של פייתון), בהפרש לא גבוה מאוד.

ניתן להסביר את ההבדלים בכך שהפונקציה הראשונה היא בסיבוכיות גבוהה יותר מהשתיים האחרות - $O(n^2k)$, ולכן עושה פעולות בסיבוכיות זמן גבוהה יותר וייקח לה יותר זמן לרוץ.

נסביר את ההבדל בין הפונקציה השנייה לשלישית, למרות שהמימוש שלהן עובד בצורה דומה יחסית, בכך שהפונקציה השלישית משתמשת בטיפוס מילון של פייתון, שם החיפוש הוא בסיבוכיות זמן קבועה, ולא בסיבוכיות O(k) כמו בפונקציה השנייה. כלומר למרות שהפונקציות פועלות באותה סיבוכיות (O(nk)), הפונקציה השלישית עושה פעולות שלוקחות פחות זמן.

כמו כן, בפונקציה השנייה נוצר מילון באורך m מלכתחילה, ואילו בפונקציה השלישית נוצר מילון בהתאם לקלטים שמתקבלים, כלומר לאו דווקא באורך m (כלומר אין "תאים ריקים").

בנוסף, במימוש של הפונקציה השנייה, תתכנה התנגשויות בשל השימוש בפונקציית ההאש, ולכן יש לבצע חיפוש בתוך ה'תא' כדי לוודא שוויון אמיתי. עם זאת, בפונקציה השלישית הגדרנו כל תא מראש כך שיכיל את כל הרשימות המקוריות שהן הרישות מופיעות.