נושאים בביואינפורמטיקה

עדן זיו 208932079, עמית שרגא 208993493

חלק א'

<u>תיאור האלגוריתם:</u>

<u>שלב א'- מיפוי הקלט:</u> המרנו את בסיס הנתונים למערך דו מימדי כאשר כל תא במערך מכיל שורה מהקובץ המיוצגת כמערך בפני עצמה. כל הנתונים בשורה הופרדו על פי סולמית (#) ועל פי טאב (\t).

<u>שלב ב'- חיפוש X:</u> כדי לחפש את X עברנו על כל השורות של בסיס הנתונים ובתוך כל שורה עברנו על כל תא, החל מהתא החמישי, שכן רק החל מתא זה נמצא רצף הגנום.

<u>שלב ג'- מציאת השכנים של X:</u> במידה ומצאנו את X, נשלח אותו יחד עם הגנום שהוא מופיע בו לפונקציה find_words המוצאת את כל שכניו (באותה השורה). הפונקציה מוצאת את כל המילים באורכים l = 2 עד l = 10 כך שX מופיע בהן ומכניסה אותן לטבלת גיבוב.
נשתמש בפונקציה find_words עבור כל שורה בבסיס הנתונים בה X מופיע לפחות פעם אחת.

- הפונקציה find_words תחילה מחלקת את הגנום שבשורה לכל זוגות השכנים האפשריים. עבור כל זוג, נכניס אותו לטבלת הגיבוב רק אם X מופיע בו. באותה דרך נעבור על השלשות, רביעיות וכן הלאה, עד מילים באורך 10.
- טבלת גיבוב זו מכילה בתור key את כל המילים (כל מילה היא מסוג tuple) באורכים 2-10 שבהן X נמצא. ו- value מסוג array עם שני תאים. התא הראשון הוא מטיפוס int שמכיל את כמות המופעים של אותה מילה בבסיס הנתונים והתא השני הוא קבוצה של שמות של אורגניזמים. בחרנו לתחזק שדה של כמות המופיעים עבור ניתוח ועיבוד הנתונים בסוף הרצת התוכנית. בחרנו לשמור את שמות האורגניזמים בקבוצה כדי שנוכל לדעת בדיוק מי האורגנזימים בהם המילה מופיעה ולהשתמש בזה בניתוח המידע בסוף. לפי גודל הקבוצה נוכל לדעת בכמה אורגניזמים שונים המילה מופיעה. כמו כן, מכיוון שמדובר בקבוצה, אם המילה כבר הופיעה באותו האורגניזם אז שם האורגניזם כבר שמור בקבוצה של המילה ולכן לא יתווסף שוב לקבוצה.

<u>שלב ד'- מציאת מילים עם יותר מq הופעות:</u> לאחר שטבלת הגיבוב עודכנה, נשלח אותה ואת q לפונקציה bigger_than_q כדי לבדוק אילו מילים מופיעות בלפחות q גנומים שונים. אם גודל הקבוצה הינו לפחות q אז נוסיף את המילה לאובייקט מסוג Counter. ונעדכן את הערך של כל מילה בCounter להיות גודל קבוצת האורגניזמים.

<u>שלב ה'- מיון המילים</u>: נשלח את הCounter לפונקציית sort_output הפונקציה תבצע most_common על הCounter. פעולה זו ממיינת את הCounter לפי הערך של כל אלמנט בסדר יורד – כלומר נקבל סידור מחדש של הCounter לפי מספר הגנומים השונים בהם מופיעות.

כעת נבצע מיון נוסף על פי אורכי מילים. תחילה, נשרשר כל מילה מהCounter הממוין לרשימה שכעת ממוינת גם כן על פי מספר הופעות יורד. נשתמש בפונקציה שמקבלת רשימה ומקבצת על פי אורך האלמנטים לתתי רשימות.

<u>שלב ו' – החזרת הפלט:</u> הפלט של התוכנית הוא רשימה של רשימות המסודרות לפי אורך מילה עולה, כך שברשימה הראשונה יש מילים מאורך 2, ברשימה השנייה מילים מאורך 3 וכן הלאה. כמו כן, כל תת רשימה ממוינת על פי מספר הופעות יורד של המילה בגנומים שונים.

<u>ניתוח זמן ריצה ומקום בזיכרון של האלגוריתם:</u>

$|text| = \boldsymbol{m} \cdot n$, כך שm *מייצג את מספר השורות בטקסט ו-*n *מייצג את אורך השורה.*

ננתח את הזמן הריצה והמקום בזיכרון של כל פונקציה בנפרד ונסכום הכל בסוף.

- def find_cog(data, cog_map, cog):
    1. for row in data:
    2. for i in range(5, len(row)):
        a. if cog == row[i]:
            i. find_words(cog_map, row, i)
            ii. break

<u>ניתוח זמן ריצה:</u> שורה 1 – מעבר על כל שורות הdataBase

שורה 2- עבור כל שורה בdataBase נעבור על כל אורכה (החל מתא 5) ונבצע בדיקות ב $O(1)$ ושליחה לfind_words (שמבוצעת ב$O(n)$ – הסבר בהמשך)

<u>סה"כ</u> : $\underbrace{m}_{number\ of\ rows} \cdot (\underbrace{O(n)}_{length\ of\ row} \cdot O(n)) = O(m \cdot n^2)$

<u>סיבוכיות מקום:</u> בפונקציה השתמשנו בפרמטרים שנשלחו מפונקציית ה *Main* ללא הקצאת מקום נוספת.

- *def find_words(cog_map, row, i):*
    1. *for l_param in range(2, 11):*
    2. *if l_param <= len(row) - 5:*
        a. *for j in range(5, len(row) - l_param + 1):*
            i. *word = tuple(row[j: j + l_param])*
            ii. *if row[i] in word:*
                1. *if word in cog_map:*
                a. *value_of_word = cog_map[word]*
                b. *value_of_word[0] += 1*
                c. *organism = row[3]*
                d. *value_of_word[1].add(organism)*
                e. *cog_map[word] = value_of_word*
                2. *else: cog_map[word] = [1, {row[3]}]*

<u>ניתוח זמן ריצה:</u> שורה 1 – מתבצעת זמן קבוע של פעמים ולכן לוקחת $O(1)$

שורה 2- מבצעת בדיקה ב $O(1)$

שורה a – מבצעת מעבר על כל הגנום- מעדכנת את ה-hashMap או מוסיפה ערך חדש ולכן לוקחת $n \cdot O(1) = O(n)$

<u>סה"כ</u> : $O(1) \cdot O(n) = O(n)$

<u>סיבוכיות מקום:</u> בפונקציה השתמשנו בפרמטרים שנשלחו מפונקציית ה *Main* ללא הקצאת מקום נוספת.

- *def bigger_than_q(cog_map, q, counter):*
    1. *for key in cog_map:*
        a. *value = cog_map[key]*
        b. *if len(value[1]) >= q:*
            i. *counter[key] = len(value[1])*

<u>ניתוח זמן ריצה:</u> שורה 1 – מבצעת מעבר על כל המילים בhashMap ועל כל מילה מבצעת פעולות ב $O(1)$

<u>סה"כ</u> : $\underbrace{l \cdot O(m \cdot n)}_{\substack{the\ max\ words\ in\ the\ hashMap \\ l\ is\ const}} \cdot O(1) = O(m \cdot n)$

<u>סיבוכיות מקום:</u> בפונקציה השתמשנו בפרמטרים שנשלחו מפונקציית ה *Main* ללא הקצאת מקום נוספת.

- *def sort_output(counter):*
    1. *output = []*
    2. *for k, v in counter.most_common():*
        a. *output.append(k)*
    3. *group = defaultdict(list)*
    4. *for c in output:*
        a. *group[len(c)].append(c)*
    5. *return group*

<u>ניתוח זמן ריצה:</u> שורה 2 – ממיינת ומבצעת מעבר על כל המילים בcounter ומוסיפה למערך output

$$\underbrace{O(m \cdot n \cdot \log(m \cdot n))}_{most\_common()} \cdot \underbrace{O(1)}_{append\ element} = O(m \cdot n \cdot \log(m \cdot n)) -$$

שורה 4- עוברת על *output* ומוסיפה לתתי רשימות על פי אורך –$O(m \cdot n)$

<u>סה"כ</u> : $O(m \cdot n \cdot \log(m \cdot n)) + O(m \cdot n) = O(m \cdot n \cdot \log(m \cdot n))$

<u>סיבוכיות מקום:</u> הקצנו מקום למערך *Output* ו- *group*. גודלם הוא $O(m \cdot n)$.

- *def main(cog, q):*
    1. *with open(WORDS_PLASMID_TXT, "rb") as f:*
        a. *data1 = f.readlines()*
    2. *split1 = [x.split(b"\t")[:-1] for x in data1]*
    3. *data1 = [list([*x[0].split(b"#"), *x[1:]]) for x in split1]*
    4. *counter = Counter()*
    5. *cog_map = {}*
    6. *find_cog(data1, cog_map, cog)*
    7. *with open(WORDS_BAC_TXT, "rb") as f:*
        a. *data2 = f.readlines()*
    8. *split2 = [x.split(b"\t")[:-1] for x in data2]*
    9. *data2 = [list([*x[0].split(b"#"), *x[1:]]) for x in split2]*
    10. *find_cog(data2, cog_map, cog)*

11. bigger_than_q(cog_map, q, counter)
12. sorted_groups = sort_output(counter)
13. print(sorted_groups)

<u>ניתוח זמן ריצה:</u> שורה 6+ 1-3 ושורות 7-10– פתיחת קובץ טקסט והכנסת כל שורה למערך לוקחת $O(m \cdot n)$. פיצול כל שורה לפי תווים לוקחת $O(m \cdot n)$ ושליחה לפונקציה *find_cog* ב- $O(1)$. ולכן:
$O(m \cdot n) + O(m \cdot n) + O(1) = O(m \cdot n)$

שורות 11-13 – ביצוע פעולות ב$O(1)$

<u>סה"כ</u> : $O(m \cdot n) + O(1) = O(m \cdot n)$

<u>ניתוח מקום:</u> בפונקציה זו הקצנו מקום ל *data1* כגודל ה*dataBase* *cog_words_plasmid.txt* ו*data2* כגודל ה *dataBase* ה*cog_words_bac.txt* . בנוסף הקצנו מקום למשתנים *cog_map*, *sorted_groups* ו- *counter* בגודל $O(m \cdot n)$ כל אחד.

סה"כ: $O(m \cdot n) + O(m \cdot n) + 3 \cdot O(m \cdot n) = O(m \cdot n)$

<u>סה"כ זמן ריצה של התוכנית:</u>

$$O(m \cdot n^2) + O(m \cdot n) + O(m \cdot n \cdot \log(m \cdot n)) + O(m \cdot n) = O(m \cdot n^2) = O(|text| \cdot |line|)$$

<u>סה"כ מקום של התוכנית:</u> $O(m \cdot n) + O(m \cdot n) = O(m \cdot n) = O(|text|)$

<u>השערת תפקיד הCOG שבחרנו: COG0121</u>

מצאנו שהמקטע (0121, 0279) הוא המקטע הנפוץ ביותר ונמצא ב128 אורגניזמים שונים.

תחילה נתמקד בזוג (0121, 0279).

Cog0279 – אחראי על Carbohydrate transport כלומר, פירוק סוכרים. הוא שייך לקבוצת הגנים phosphoheptose isomerase. קבוצה זו היא חלבון שמתפקד כאנזים המזרז פירוק של גלוקוז ופרוקטוז.

רביעייה שמופיעה מספר גבוה של פעמים ביחס לשאר הרצפים היא: (3572, 1262, 0121, 4301) הרביעייה מופיעה ב61 אורגניזמים שונים.

COG3572 הוא חלק מאנזים שמזרז העברה של קבוצות פונקציונאליות ממולקולות של glutathione (אנטי אוקסידנט), והופך אותן לגלוטומט.

COG1262 חלק מאנזים הדרוש לזירוז ההמרה מציסטאין לפורמילגליצין(קבוצת סולפט).

מכאן ניתן להסיק שככל הנראה COG0121 גם הוא חלק מאנזימים האחראיים לפעילות מטבולית של פירוק מולקולות. על פי מספר הופעות גדול יחסית במקטע (0121, 0279). נסיק שהפעילות העיקרית שלו קשורה בפירוק סוכרים. מידע קודם כל תא צריך ATP כדי לבצע פעולות בסיסיות ובפרט פעילות אנזימטית. ומכאן השערתנו היא שגם ברצפים אחרים הCOG0121 אחראי על פירוק סוכרים, זאת כדי שפעילות האנזימים בשאר הרצפים שבהם הקוג מופיע תתרחש.