נושאים בביואינפורמטיקה

עדן זיו 208932079, עמית שרגא 208993493

חלק ג'

<u>תיאור האלגוריתם:</u>

<u>שלב א'-</u> הקלט: נריץ את האלגוריתם מסעיף ב' ונשתמש בקובץ תוצאות של אלגוריתם זה, כלומר נשתמש ב-sorted_groups_k וב-output. כאשר sorted_groups_k הוא במבנה נתונים של מילון כך שה-key הוא אורך מילה וה-value הוא מערך ממוין של מילים בסדר יורד על פי שכיחויות עם עד k הכנסות וה-output מכיל פשוט את כל המילים עם עד k הכנסות ממויין רק לפי שכיחויות.

<u>שלב ב'-</u> חיפוש קבוצות המילים שבהן יש לכל היותר d מחיקות: נשלח לפונקציה *d_deletions* את הערך d, את הקבוצות הממוינות שמצאנו בחלק ב' ו- *counter* ריק. עבור כל מילה (W) שהתקבלה בחלק ב', נחפש את המופעים שלה רק בקבוצות המילים באורך $|W| - i$ כך ש- $1 \leq i \leq d$ (קבוצות המילים נמצאו ומוינו בחלק ב').

<u>שלב ג' –</u> חיפוש המילה W בתוך קבוצת מילים מסוימת: מ- d_deletions נקרא לפונקציה *is_find_word_d* ונשלח אליה את המילה W ואת קבוצת המילים בגודל הרלוונטי, נעבור על כל המילים בקבוצה ועבור כל מילה נבדוק האם היא *substring* של W. בגלל שהקבוצות מכילות רק מילים באורך יחיד וספציפי (שתואם את הדרישות), כשנמצא מילה שמהווה תת-רצף של W נוכל לספור אותה כהופעה של W. נעדכן את ה*counter* עבור המילה.

<u>שלב ד'-</u> מיון התוצאות + החזרת הפלט: נאחד את הcounter מסעיף ב' (1 עד k הכנסות) עם הcounter מסעיף ג' (1 עד d מחיקות) בפונקציה update_counter3. נשלח את הcounter לפונקציה שתעדכן בה רק את המילים בהן מספר ההופעות הוא מעל q. בסוף נבצע מיון לcounter המעודכן באופן דומה למיון שבוצע בחלק ב'- קבוצות על פי אורך מילה עולה, כך שכל קבוצה מסודרת בסדר יורד לפי כמות הופעות.

<u>ניתוח זמן ריצה ומקום בזיכרון של האלגוריתם:</u> נתאר רק את הפונקציות החדשות שנוספו בחלק 3.

- $|text| = m \cdot n$, כך שm מייצג את מספר השורות בטקסט ו-n מייצג את אורך השורה.
- def d_deletions(sorted_groups, output, d, counter3):
  1. for word in output:
     a. len_word = len(word)
     b. if len_word > 2:
        i. sum_d = 1
        ii. check_key = len_word - sum_d
        iii. while sum_d <= d and check_key >= 2 and check_key in sorted_groups.keys():
           1. group = sorted_groups.get(check_key)
           2. find_word = is_find_word_d(word, group)
           3. counter3[word] = counter3[word] + find_word
           4. sum_d += 1
           5. check_key -= 1

<u>ניתוח זמן ריצה:</u> מעבר על כל המילים מחלק ב' כאשר עבור כל מילה נרצה לבדוק במילון sorted_groups את קבוצת המילים באורך קטן ב-1 ממנה עד שנגיע במקסימום לקבוצת המילים

באורך $|W| - d$ או לאורך 2.

סה"כ זמן ריצה: כ $\underbrace{O(m \cdot n)}_{|output|} \cdot (\underbrace{d}_{loop\ line\ iii-count} \cdot \underbrace{O(m \cdot n)}_{\text{is\_find\_word\_d()}}) = O(d \cdot m^2 \cdot n^2)$

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

- def is_find_word_d(word, group):
    1. total_sum = 0
    2. for arr in group:
        a. j = 0
        b. for i in range(0, len(word)):
            i. if word[i] == arr[j]:
                1. j += 1
            ii. if j == len(arr):
                1. total_sum += 1
                2. Break
    3. return total_sum

<u>ניתוח זמן ריצה:</u> מעבר על כל איבר בקבוצה group (מילים באורך ספציפי קבוע) כאשר על כל איבר נעבור במקסימום פעם אחת ונבצע השוואה למילה word (זמן קבוע). לכן מקסימום זמן הריצה הוא כגודל ה data - כמות המילים שיכולות להיות בקבוצה - כלומר $O(m \cdot n)$

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

- def update_counter3(counter2, counter3):
    1. for word in counter2:
        a. counter3[word] += counter2[word]

<u>ניתוח זמן ריצה:</u> מעבר על counter2 כלומר סה"כ $O(m \cdot n)$

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

- Main
    ...
    1. counter3 = Counter()
    2. d_deletions(sorted_groups_k, output, d, counter3)
    3. update_counter3(counter2, counter3)
    4. new_counter = bigger_than_q2(q, counter3)
    5. output = []
    6. sorted_groups_d = sort_output(new_counter, output)
    7. print(sorted_groups_d)

<u>ניתוח זמן ריצה:</u> ננתח את זמן הריצה של פונקציית main בהתבסס על זמן הריצה מהאלגוריתם של חלק ב'. כל הפעולות הן פעולות בזמן ריצה של $O(1)$ (שליחה לפונקציה, הדפסה, שינוי ערך של מערך). את זמן הריצה של כל פונקציה ניתחנו למעלה. בנוסף בחלק זה הוספנו פרמטר נוסף לחתימה של

הפונקציה d אשר נשלח על ידי המשתמש ולכן הוא גם יהיה חלק מזמן ריצה בחלק זה.
<u>זמן הריצה הכולל של חלק זה הוא:</u>

$$\underbrace{O(mn\cdot(l\cdot n+k\cdot m\cdot n))}_{part\ B:\ k\_instances}+\underbrace{O(d\cdot m^2\cdot n^2)}_{d\_deletions}+\underbrace{O(m\cdot n)}_{update\_counter3}+\underbrace{O(m\cdot n\cdot log(m\cdot n))}_{sort\_output}$$
$$=max\{O(mn\cdot(l\cdot n+k\cdot m\cdot n)),O(d\cdot m^2\cdot n^2)\}$$

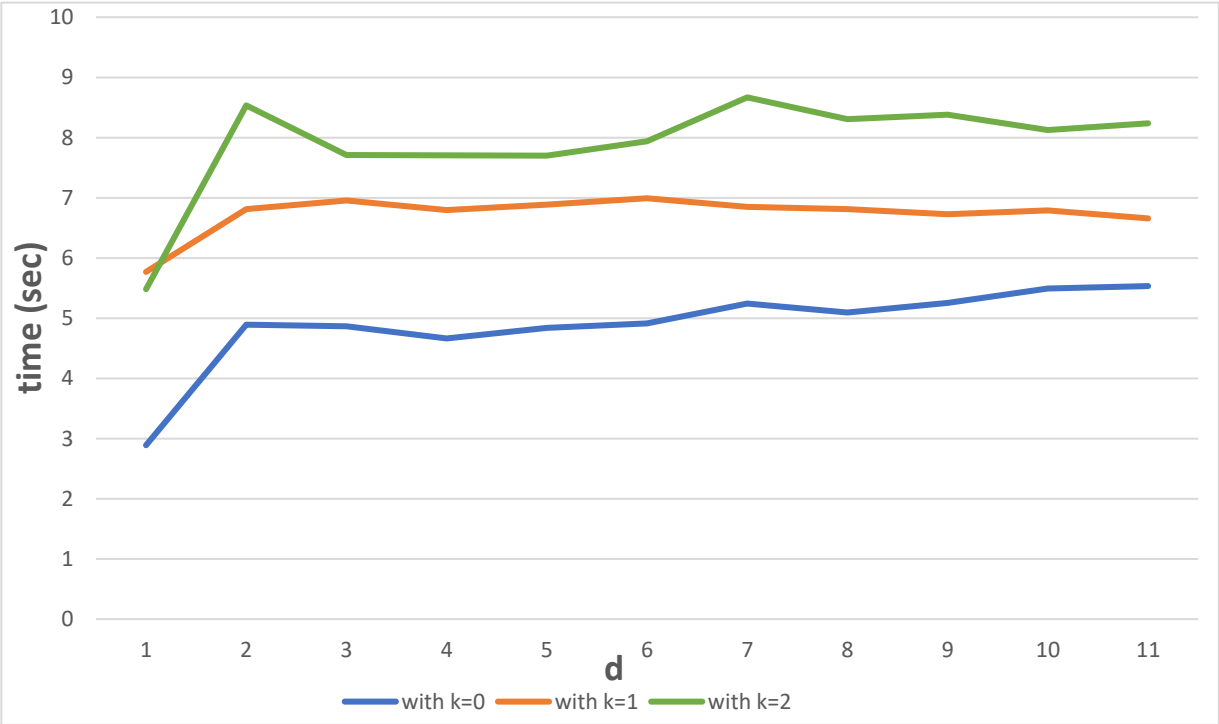
<u>סיבוכיות מקום:</u> בחלק זה הוספנו מקום בזיכרון עבור המשתנה counter3. משתנה זה, בדומה לcounter2 תופס בזיכרון מקום של $O(m\cdot n)$.

לסיכום, הרחבת הבעיה כך שניתן למחוק עד d אותיות לתבנית של מילה לא פוגעת בזמן הריצה. זמן הריצה של חלק ב' הוא $O(|text|\cdot(l\cdot|line|+k\cdot|text|))$. כלומר, בחלק ב' זמן הריצה היה תלוי בריבוע גודל הטקסט. הפונקציות שהוספנו בחלק ג' רצות בזמן ריצה שגם תלוי בריבוע גודל הטקסט - $O(d\cdot m^2\cdot n^2)=O(|text|^2)$. לכן סה"כ זמן הריצה עדיין תלוי בריבוע גודל הטקסט.

לגבי סיבוכיות מקום – בחלק ב' השתמשנו ב$O(4\cdot|text|)=O(|text|)$ מקום בזיכרון. בחלק ג' הוספנו counter ב -$O(|text|)$ מקום, לכן סה"כ השתמשנו ב- $O(5\cdot|text|)=O(|text|)$ מקום. לכן, סיבוכיות המקום לא השתנתה.

<u>תלות של d על זמן הריצה</u>: הרצנו את התוכנית עם l=10,q=20, שינינו את d ובדקנו זמן ריצה.

with k=2	with k=1	with k=0	d
5.4817033	5.769248	2.8889407	0
8.5364022	6.8133602	4.88964	1
7.7094996	6.9557873	4.8676875	2
7.7054097	6.7985755	4.6608134	3
7.6986595	6.8841296	4.838927	4
7.9385802	6.9927188	4.9154608	5
8.668912	6.8504204	5.2455602	6
8.3062986	6.812769	5.0938347	7
8.3806574	6.7264284	5.2551997	8
8.1248084	6.788494	5.4909736	9
8.23517117	6.657191	5.5334368	10



לפי הגרף ניתן לראות שככל שK גדל - זמן הריצה עולה. עבור ערכי K גדולים צריך לבדוק יותר קבוצות אורכים של מילים, לבצע יותר השוואות ולכן זמן הריצה עבור k=2 גבוה מזמן הריצה עבור k=1 שגבוה מk=0.

באותו אופן, עבור כל גרף, ניתן לראות שככל שd גדל – זמן הריצה עולה. עבור ערכי d גדולים נבדוק יותר קבוצות אורכים של מילים.

עם זאת, ככל שערכי d גדלים השיפוע של הגרף נעשה פחות תלול. ככל ש-d גדול יותר, מאפשרים יותר מחיקות, וצריך לחפש מילים באורך $|W| - i$ כך ש$1 \leq i \leq d$. כאשר $|W| - i \leq 2$, נפסיק את החיפוש ולכן משלב מסוים לא משנה כמה גדול יהיה d לא נוכל למחוק יותר מאורך המילה, זמן הריצה יתקבע והגרף יתייצב. נצפה שנגיע לנקודה בה הגרף יתיישר כמעט (מגמה שניתן לזהות בגרף).

<u>דיון בהרחבת המשימה:</u>

כפי שלמדנו בכיתה, במהלך האבולוציה התרחשו מוטציות ברצפי הדנ"א. חלק מהמוטציות גרמו להוספת נוקליאוטידים, חלק למחיקת נוקליאוטידים וחלק לשחלוף. בפרויקט שלנו נעסוק במקרים של הוספה(חלק ב), מחיקה והשילוב ביניהם (חלק ג). על מנת להגדיל את הsensitivity של האלגוריתם, כלומר את מספר הרצפים הדומים לcog שלנו נוסיף גם מחיקות לרצף. עבור cog מסוים נקבל רצפי אותיות שמכילות אותו במעל q גנומים עם עד d מחיקת אותיות. נשים לב שכאשר אנחנו מעלים את ה sensitivity אנחנו מורידים את ה specificity ולכן נרצה לעלות את ה sensitivity רק במידה מסוימת.

בחלק ב' חקרנו את cog0586. בחלק זה המשכנו לחקור אותו וניסינו להבין כיצד משפיעות המחיקות על דירוג הרצפים השונים. כעת נתמקד ברצף (0136, 0101, 0586, 0777, 0285) . כאשר הרצנו את הקוד עם k=0 וערכי d משתנים. קיבלנו שכאשר d=0 רצף זה מופיע 104 פעמים ודירוגו לפי מספר ההופעות הוא 5. כאשר d=1 הרצף הופיע 106 פעמים ודירוגו שונה ל4 וכאשר הרצנו עם d=2 הרצף הופיע כבר 109 פעמים ודירוגו שונה ל3. כלומר ניתן להבחין באופן מגמתי כי המחיקות בלבד משנות באופן מבוהק את דירוג הcog הנ"ל.
באופן דומה ביצענו בדיקה אך הפעם בתוספת הכנסה כלומר כאשר k=1. כאשר הרצנו עם d=0 קיבלנו שמספר ההופעות של רצף זה הוא 105 דירוגו הוא 4 וכאשר הרצנו עם d=2 קיבלנו שהרצף כבר מופיע 110 פעמים ודירגו הוא 3.

נתייחס לנתון האחרון ונרחיב עליו מעט. כאשר ההרצה הייתה עם k=1, d=2 מיקום הרצף אומנם שלישי אך מבחינתו הוא ממוקם ראשון מכיוון ששני הרצפים שנמצאים לפניו בדירוג
[(0586, X):192, (X, 0586):188)] הם רצפים מאורך 2 שלא ניתן לבצע בהם מחיקות ומכילים cog מסוג X שלא ניתן להסיק עליו מידע – ולכן התעלמנו מרצפים אלו בניתוח פעילות הcog שלנו. כלומר בעצם הרצף שבחרנו הפך להיות הרצף הראשון שניתן להסיק ממנו מידע אודות פעילות הcog שלנו 0586.

היות והרצף מכיל cog'ים הקשורים לפעילות מטבולית (cog0285, cog 0777, cog0136) מתחזקת השערתנו מחלק ב' שגם הcog הלא ידוע 0586 קשור לפעילות מטבולית.