המחלקה להנדסת תוכנה

Paralleling BWA-Aligner

מקבול חיפוש מיקומם של מקטעי DNA על פני הגנום האנושי.

חיבור זה מהווה חלק מהדרישה לקבלת

תואר ראשון בהנדסה

מאת

אבי טרנר

תמוז תשע"ה
2015

יולי

המחלקה להנדסת תוכנה

Paralleling BWA-Aligner

מקבול חיפוש מיקומם של מקטעי DNA על פני הגנום האנושי.

חיבור זה מהווה חלק מהדרישה לקבלת

תואר ראשון בהנדסה

מאת

אבי טרנר

מנחה: ד"ר חסין יהודה

אישור המנחה: **תאריך:**

תקציר

הפרויקט הוא פרויקט מחקרי המתמקד בהבנת אלגוריתם BWA ובאפשרויות המקבול שלו.

אלגוריתם BWA הוא אלגוריתם לחיפוש מחרוזות קצרות בתוך מחרוזת ארוכה וקבועה. השימוש הנפוץ היותר לאלגוריתם BWA הוא בתחום הביואינפורמטיקה והתאמת דגימות DNA למיקומן על פני הגנום האנושי.

ה-DNA, שנלקח מאדם בריא, מיוצג ע"י מחרוזת ארוכה של תוים (לדוגמא: GCATGCTAGCTAGCTCGATGCTACTGACCGTCAG....) ומשמש כבסיס להשוואה לדגימות עתידיות (דבר זה מתאפשר בזכות העובדה שכ- 99.9% מה-DNA של כל בני האדם משותף).

כאשר אדם מגיע לאבחון גנטי, נלקחות ממנו דגימות DNA קצרות (35-200bp) אך בכמות מאוד גדולה (מספר בודד של מיליארדים) ועל מנת למצוא מוטציות יש להתאים כל אחת מהדגימות למיקומה על פני הגנום.

חלק מהאתגר במשימה זו, הוא העובדה שהדגימות לא זהות לחלוטין לגנום המושווה (מוטציות / טעויות בקריאת המכונה הדוגמת...)

עקב הכמות הגדולה של הדגימות, אורכו של הגנום ותלות זמן הריצה באורך הגנום, אלגוריתמים "מסורתיים" לחיפוש לא מצליחים להתמודד עם משימה זו בזמנים פרקטיים. לדוגמא :

1. האלגוריתם הנאיבי: $\theta(\mathrm{n}^2 \cdot |w|)$
2. KMP : $\theta(n^2)$

כאשר:
$|w|$ - אורך של דגימה.
n – אורך הגנום
ובהנחה (הנכונה) שמספר הקריאות הוא בסדר גודל של אורך הגנום.
בנוסף, אלגוריתמים אלו אינם מתמודדים עם בעית חוסר ההתאמה המדויקת בין הדגימות לגנום.

לעומת אלגוריתמים אלו, אלגוריתם BWA מוצא התאמות לדגימות בזמן ריצה של $\theta(|w|)$ (ללא תלות באורך הגנום!) ויודע להתמודד עם אי התאמות בין הדגימות לגנום.

במסגרת הפרויקט זה מימשנו את האלגוריתם ומיקבלנו אותו כך שכל דגימה מעובדת ע"י תהליכון נפרד. בנוסף, בנינו תכנה שמדמה דגימות הכוללות טעות קריאה / מוטציה ובדקנו את המיקבול עם פרמטרים שונים.

על מנת לבחון את ייעול שלב זה התמקדנו בבחינת:

1. השפעות אורך הדגימות על זמן הריצה.
2. השפעות מספר הדגימות על זמן הריצה.
3. השפעות של מיון הדגימות על זמן הריצה.
4. השפעת מקבול התהליך על זמן הריצה.

לאחר בדיקה והשוואות הממצאים, נראה שהתוצאות הטובות ביותר מתקבלות ע"י מקבול התהליך, באופן שבו כל דגימת DNA מקבלת תהליכון (בניגוד למקבול פנימי של התהליך).

ייעול מסוג זה נשאר יציב ללא תלות באורך / מספר / מיון הדגימות.

הצהרה

ההעבודה נעשתה בהנחיית ד"ר חסין יהודה, המכללה האקדמית להנדסה ירושלים – עזריאלי המחלקה להנדסת תוכנה.

החיבור מציג את עבודתי האישית ומהווה חלק מהדרישות לקבלת תואר ראשון בהנדסה.

תודות

ברצוני להביע תודות לאנשים שבהם נעזרתי בפרויקט.

תודה מיוחדת לאשתי ומשפחתי, שעזרו ותמכו בי לאורך כל מהלך הלימודים בכלל והפרויקט בפרט.

תודה לעמיתי עז ידגר, מבצע הפרויקט המקביל, שעמו התייעצתי במהלך הפרויקט.

ולבסוף תודה מיוחדת למנחה הפרויקט ד"ר יהודה חסין על העזרה המקצועית, יחסו האישי, העצמאות והתמיכה שניתנה לי במהלך ביצוע עבודת הגמר.

תודה!

תוכן העניינים

מילון מונחים

w – קריאה – קטע מה- DNA של חולה שנדגם לצורך בדיקה.

X – רפרנס - הגנום שנשתמש בו להשוואה לכל הקריאות.

bp – base pair - זוג בסיסים המהווים שלב ב"סולם ה-DNA". משמש כמידת אורך של קריאה.

$|w|$ - אורך של קריאה. באופן טיפוסי הקריאות הן באורך של 100 bp- 200 bp אך ה"לקוח" הספציפי של תוצר הפרויקט, משתמש בקריאות באורך 35bp.

X_i – המחרוזת על גבי X באורך $|w|_{bp}$, המתחילה באינדקס i.

w_i – הקריאה שהחלה במקום ה i ב X

n – אורכו של X ביחידות bp - $|X|$

m – מספר הקריאות הנדגמות מחולה. סדר גודל של $\sim n$

Illumina – מכונה הדוגמת קריאות.

מרחק עריכה – מרחק עריכה בין שתי מחרוזות מוגדר כמספר המינימלי של פעולות עריכה שיש לבצע על מחרוזת אחת כדי להגיע למחרוזת השנייה, כאשר פעולות העריכה המותרות הן: הוספת אות, מחיקת אות או שינוי אות לאות אחרת.

שגיאה בקריאה – תו שנקרא כתו אחר, תו שהוסף, או תו שהושמט בקריאה שנקראה ע"י Illumina,

תיאור מסגרת הפרויקט

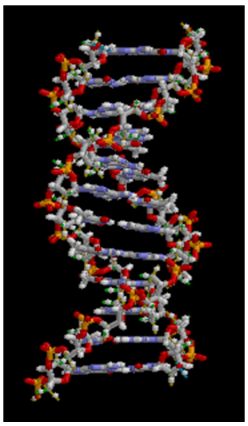
לאחר שב-2003 מופה הגנום האנושי, תהליכים רפואיים הקשורים לגנום הפכו לאפשריים.

אבחון מוטציות היה אחד היישומים המבטיחים, בפרט בתחום אבחון הסרטן.

האתגר שבא לפתחם של המהנדסים היה היכולת לעבד כמויות מידע בסדר הגודל של הגנום האנושי, תוך שמירה על זמן ריצה מעשי.

נפתח בתיאור קצר של הרקע ושל מונחים הקשורים לפרויקט.

DNA: ה-DNA היא מולקולת ענק שמצויה בכל אחד ואחד מתאי הגוף שלנו ובה מצוי כל המידע התורשתי לבניית החלבונים בתא אצל כל האורגניזמים הידועים, מחיידקים ועד בני אדם.



תמונה 1 –

ה- DNA בנוי כמעיין "סולם" שמסתלסל סביב עצמו

המבנה של ה- DNA בנוי כמעיין "סולם" שמסתלסל סביב עצמו, כאשר ה"שלבים בסולם" מורכבים, כל אחד, מזוג בסיסים המתחברים זה לזה ומסומנים באותיות הלטיניות A, G, T, C.

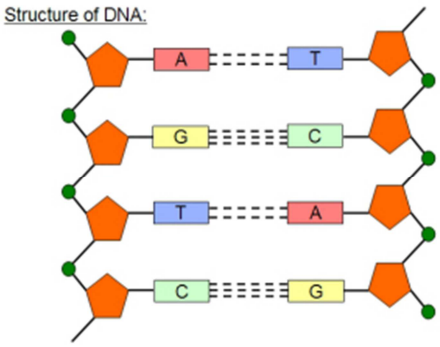
בכל "שלב" מתחברים הבסיסים עם בן זוג קבוע – A עם T ו C עם G, כך שאם ידוע לנו רק צד אחד של ה"סולם" אנו יכולים לשחזר ממנו במדויק גם את הצד השני.

המדהים הוא שכ- 99.9% מה-DNA של כל בני האדם משותף למרות אבני הבניין המועטות והפשוטות– וזוהי תכונה קרדינלית לפרויקט זה.

אם כן, את המידע המגולם ב-DNA ניתן לייצג כמחרוזת של הבסיסים המרכיבים אותו, וכך נתייחס אליו בפרויקט זה (לדוגמא: TGACCGTCAG....)

ה- DNA מורכב מכ-$6\ X10^9$ שלבים. במונחים דיגיטלים, המידע הגלום ב- DNA שווה ערך למעט יותר מ 1.6 GB (או כ-2 דיסקים...)

כמו כן, DNA יכול לעבור מוטציה, שינוי. רוב המוטציות אינן מזיקות אך אם הן מופיעות במקומות מסוימים על גבי רצף ה-DNA הן יכולות לגרום לבעיות גנטיות וביניהן לנטיה למחלות גנטיות ובפרט לסרטן.



תמונה 2 –

הבסיסים מתחברים עם בן זוג קבוע – A עם T ו G עם C

כיום, כאשר חולה מגיע לאבחון לראות האם הוא חולה בסרטן, תהליך הבדיקה הוא ארוך ומסורבל:

1. לקיחת דגימת DNA.
2. השוואת ה-DNA ל-DNA של אדם בריא.
3. איתור מוטציות ב-DNA.
4. חיפוש ידני במאגרי מידע האם המוטציות הן במקום שידוע כגורם לסרטן.
5. חיפוש ידני האם קיימת תרופה שעוזרת לסוג המסוים של הסרטן הנ"ל.

בין הבעיות בתהליך זה ניתן לציין את תהליך ההשוואה (2) שלוקח זמן ארוך במיוחד. כיום, בתי החולים שוכרים חוות שרתים על מנת ליעל את החישוב הארוך של ההשוואה.

עם זאת, גם כיום שלב זה לוקח כיום שלם. פרויקט זה יתמקד בבחינת ייעול שלב ההשוואה.

כחלק מתהליך אבחון סרטן אצל חולים, נדגם ה-DNA שלהם ונבדק על מנת למצוא מוטציות שעלולות לגרום לסרטן.

תהליך הדגימה מתבצע באמצעות מכונה הנקראת Illumina החותכת מספר גדול של מקטעי DNA באורך קצר יחסית (כ 35-200bp) במקומות אקראיים, קוראת אותם ומיצאת את תוכן הקריאות לקובץ בפורמט של הבסיסים החנקניים של המקטעים שנדגמו (לדוגמא: ...ATCGCA).

הIlluminaה עובדת במתודולוגיה הנקראת NGS (ראה מסגרת להרחבה).

פרוייקט זה מבוצע תחת הנחייתו של דר' חסין יהודה.

פרוייקט זה מהווה פרוייקט גמר לתואר "הנדסת תוכנה" במכללת "עזריאלי – המכללה האקדמית להנדסה" בירושלים.

סטודנטים נוספים העובדים על פרוייקטים הקשואים לפרוייקט זה: עז ידגר.

הרחבה: ***NGS – Next Generation Sequencing – ריצוף מהדור החדש***

באופן כללי, Sequencing היא שיטה לקביעת הסדר המדויק של נוקלאוטידים במולקולת DNA נתונה, או בהפשטה – בהינתן מולקולת DNA, מהו סדר אבני הבסיס שבה.

בעשור האחרון השימוש ב-Sequencing גדל בצורה אקספוננציאלית בעיקר בשירות מעבדות קליניות ומחקר בכל רחבי העולם, כשהמתשמש הבולט מביניהם היה פרויקט מיפוי הגנום האנושי שהסתיים ב-2003.

בפרויקט מיפוי הגנום האנושי, נעשה שימוש ב-Sanger Sequencing שפותח ב-1977 והיה שייך לדור הראשון של הSequencing (ריצוף).

מאז השלמת רצף הגנום האנושי הראשון, חל גידול בביקוש לשיטות Sequencing זולות ומהירות יותר. דרישה זו האיצה את פיתוח Next Generation Sequencing (NGS).
פלטפורמות NGS מבצעת ריצוף מסיבי באופן מקבילי, שבמהלכו מיליוני שברי DNA ממדגם בודד מרוצפים בתיאום. כחלק מהתהליך, שברי ה-DNA נחתכים "אורך" בגלל המבנה של ה-DNA, ניתן לשחזר כל צד מה- DNA ע"י בן זוגו שנחתך. מקטעי DNA אלו הם הקלט של פרויקט זה.

טכנולוגיית ה- NGS מאפשרת תפוקה גבוהה מהדור הקודם, וריצוף של כל הגנום בפחות מיום אחד.

בעשור האחרון, מספר פלטפורמות NGS פותחו והן מספקות ריצוף תוך עלות נמוכה, וקצב תפוקה גבוה. פיתוח פלטפורמות אלו הנגיש את ה- NGS עבור מספר גדל והולך של מעבדות, ובעקבות כך גדל המחקר והאבחון קליני המבוצעים ע"י ריצוף חומצות גרעין.

בין שתי הפלטפורמות הנפוצות קיום עבור NGS, נתין לציין את Illumina שנמצא בשימוש בביה"ח הדסה, ועבורו מתבצע פרויקט זה.

תיאור הבעיה

<u>תיאור פורמאלי</u>:
בהינתן:

X – רצף של תווים באורך $|X|$.

$S = \{s_1, s_1, \dots, s_m\}$ – אוסף של תווים בעלי אורך w, $w \ll |X|$.

X_i – המחרוזת על גבי X באורך m, המתחילה באינדקס i.

e – מרחק העריכה המותר בהשוואת 2 מחרוזות, כך שהן יחשבו עדיין "דומות".

בפרויקט שלנו:

X – רצף הגנום האנושי, באורך של $\sim 6\ X10^9 bp$.

S – אוסף של מקטעי ה DNA שנדגמו ע"י הIlluminaה, כל דגימה היא באורך של $35bp$,

מנסיון העבר, על מנת לקבל (סטטיסטית) כיסוי טוב של כל הגנום – יש לדגום דגימות אקראיות באורך כולל של פי 30 מאורך הגנום:

$m = |S| = 30 * \frac{|X|}{w} = 30 * \frac{6\ X10^9}{35}$.

הגדרת הבעיה:
עלינו למצוא עבור כל s_j את אוסף כל הX_i הנמצאים במרחק עריכה e ממנו.

לשם המחשה, בהינתן מחרוזת:

אינדקס	0	1	2	3	4	5	6	7	8	9	10	11	12	13
ערך	A	T	G	C	C	T	A	G	G	C	A	A	A	T

ודגימה

C	A	T

ומרחק עריכה מותר $e = 1$
נצפה לקבל את האינדקסים הבאים בתור התאמות: 9, 11 מפני שהן CAA והן AAT שוות ל- CAT עם אות אחת שהתחלפה.

הצעות לפתרונות (לא מעשיים)

חיפוש ללא טעות

נתאר מעט את תהליך ההשוואה של מחרוזות שלא נפלה בהן טעות כדי לעמוד על הקושי הכרוך ביישום מעשי של אלגוריתם שכזה:

1. **<u>האלגוריתם הנאיבי</u>**:
   עבור כל s_j נבדוק עבור כל X_i האם $X_i == s_j$
   יעילות האלגוריתם עבור דגימה בודדת: $\theta(n \cdot |w|)$.
   יעילות האלגוריתם עבור m דגימות: $\theta(m \cdot n \cdot |w|)$.
   מפני ש $\theta(m) = n$: $\theta(n^2 \cdot |w|)$

2. **<u>KMP</u>**
   אלגוריתם המנצל את מבנה התבנית על מנת ליעל את החיפוש.
   יעילות האלגוריתם עבור דגימה בודדת: $\theta(n)$
   יעילות האלגוריתם עבור m דגימות: $\theta(m \cdot n \cdot |w|)$.
   מפני ש $\theta(m) = n$: $\theta(n^2)$

מכיוון שn גדול מאוד, אלגוריתמים אלו אינם מעשיים במקרה שלנו.

חיפוש עם טעות

כפי שציינו, באלגוריתמים שסקרנו הנחנו שבמחרוזת שמחפשים לא נפלו טעויות. אם נניח שייתכן שנפלו טעויות במחרוזת שמחפשים, זמן הריצה בפועל גדל משמעותית.

נעמוד על הקושי שבהשוואה שכזו על ידי ניתוח של אלגוריתם השוואה נאיבי של קריאה (מחרוזת) שיתכן שנפלו בה 0-2 שגיאות במקום לא ידוע:

נניח שאורך של קריאה הוא 100.

Table 1

המקרה	מספר המחרוזות להשוואה		מיקום השגיאות	שגיאות אפשריות
לא נפלה אף שגיאה	ישנה מחרוזת 1 להשוואה	$1 * 1 = 1$	$\binom{100}{0}$	$\binom{4}{1}^0$
נפלה שגיאה אחת	ישנן 400 מחרוזות להשוואה	$4 * 100 = 400$	$\binom{100}{1}$	$\binom{4}{1}^1$
נפלו 2 שגיאות	ישנם 79,200 מחרוזות להשוואה	$4 * 4 * 4950 = 79{,}200$	$\binom{100}{2}$	$\binom{4}{1}^2$

סה"כ, בהנחה של 2 שגיאות, עברנו מקריאה אחת באורך 100 ל- 79,200 מחרוזות באורך 100 שנצטרך להשוות. מכיוון שמלכתחילה יש לנו מספר גדול של מחרוזות כאלו, ברור שחיפוש שכזה אינו ישים עבור מידע מסדר גודל של הגנום האנושי.

<u>הערה:</u> בתיאור זה הנחנו, לשם הפשטות, שטעויות באות לידי ביטוי בהחלפת אות אחת באות אחרת בעוד שלמעשה יתכנו טעויות של החלפת מיקומים של אותיות / קטעים, וכן טעויות של השמטת / הוספת אות.
על מנת להתגבר על בעיות כגון אלו, בשלב הדגימה לוקחים המון דגימות – בכמות כזו שסטטיסיטית כל מקטע של ה-DNA נדגם מספר פעמים. דבר זה עוזר כדי לוודא שאכן כל ה-DNA נדגם (בסבירות גבוהה) וגם מחפה על טעויות בקריאה (לא סביר שתהיה טעות קריאה של המכונה באותו המקום בכל הדגימות).

תיאור הפתרון

אלגוריתם BWA

על מנת להתגבר על בעית זמן הריצה, פותח האלגוריתם BWA.
בבסיסו, האלגוריתם מחולק לשלושה שלבים:

1. *Index* - אינדוקס של הגנום – שמירת הגנום בצורה שניתן לחפש עליו מחרוזות בצורה יעילה. זו פעולה שיש לעשות אותה פעם אחת בלבד (ולא בכל בדיקה של חולה...). האינדקס שמור כעץ סיפות (ראה הרחבה בהמשך).
2. *Alignment* – מציאת המיקום של הקריאות על פני הגנום.
3. *Pairing* – כחלק מקריאת ה-DNA של החולה, הדגימות נחתכות ל- 2 ויש צורך למצוא התאמה בין 2 חלקי הדגימה.

שלב ה-*Alignment* (2), שבו מתמקד פרויקט זה, מיועל ע"י ישום של "עץ סיפות" השומר את כל הסיומות האפשריות של הגנום (זהו האינדקס משלב 1), ומימוש של אלגוריתם המחפש ב"צורה חכמה" עבור כל קריאה לאיזה סיומת היא מתאימה בהינתן מספר טעויות כלשהו (זהו פרמטר הניתן לשינוי).

יעילות האלגוריתם : $\theta(|w|)$ (לא תלוי באורך הגנום!).

החיפוש החכם והיעילות הגבוהה של האלגוריתם מושג בזכות שתי תכונות עיקריות:

1. החיפוש לא מתבצע על כל האינדקס אלא מתחשב בפרמטר של מספר הטעויות המותר אל מול מספר הטעויות שנמצאו עד כה ומפסיק את החיפוש בענף הנכחי של האינדקס מרגע שאפסו הסיכויים למצוא התאמות נוספות.
2. מחרוזות בעלות סיומת זהה נמצאות באותו המיקום באינדקס וחיפוש בודד ימצא את כולן.

המחשה גרפית לחיפוש נמצאת בעמוד הבא.

בפועל, יש לאלגוריתם זה עלויות נוספות:

- Pre Processing : $O(n)$
- את הפעולה מבצעים על מספר גדול מאוד של קריאות, m.

ולכן, סה"כ יעילות הלגוריתם היא $\theta(n) + m\theta(|w|)$
כיום, למרות השימוש באלגוריתם זה, התהליך לוקח זמן רב (כיום שלם) עקב ריבוי הקריאות (m).

בניית האינדקס בעזרת מערך סייפות:
יהי X = googol$
נבצע הזזה מחזורית של X, ונשמור תוצאה של כל הזזה כרשומה.
לאחר מכן נמיין את הרשומות מיון לקסוגרפי.
לאחר המיון, אוסף התווים הראשונים מכל רשומה מהווים את מערך הסיומת S(i) (6,3,0,5,2,4,1)

השרשור של התווים האחרונים של הרשומות נותן לנו את "מחרוזות BWT" B[i]- lo$oogg .
וזהו למעשה האינדקס שנשתמש בו.
(ניתן לעיין בנוגע ל טרנספורם BWT – זהו נושא שלם בפני עצמו).

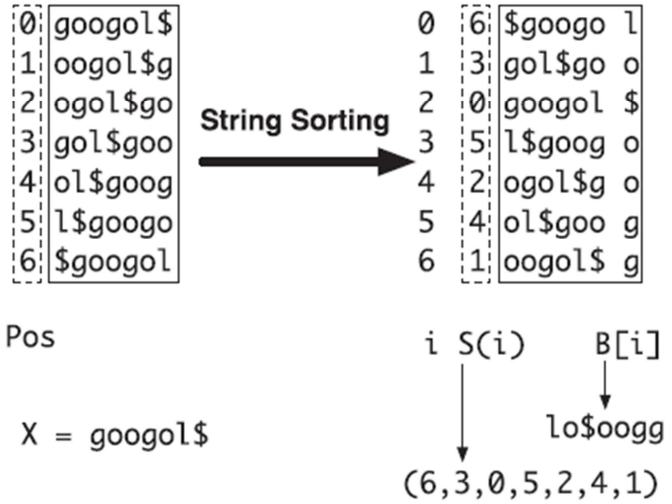


Figure 1 - בניית האינדקס - עץ סייפות

נמחיש את צורת העבודה של האלגוריתם באמצעות עץ רישות.

<u>הערה</u>: מבלי להוכיח זאת, עץ רישות שקול למערך סיפות (לצורך אינטואיציה – עץ הרישות של X זהה לעץ הסיפות של $Reverse(X)$, ולכן כל הדגמה על עץ רישות נכונה גם עבור עץ סיפות).

להלן **עץ רישות** של המחרוזת "Googol". הסמל ∧ מסמן את תחילתה של המחרוזת.
שני המספרים בכל צומת הם האינטרוול ב- SA(מערך הסיפות) של המחרוזת המיוצגת על ידי הצומת.

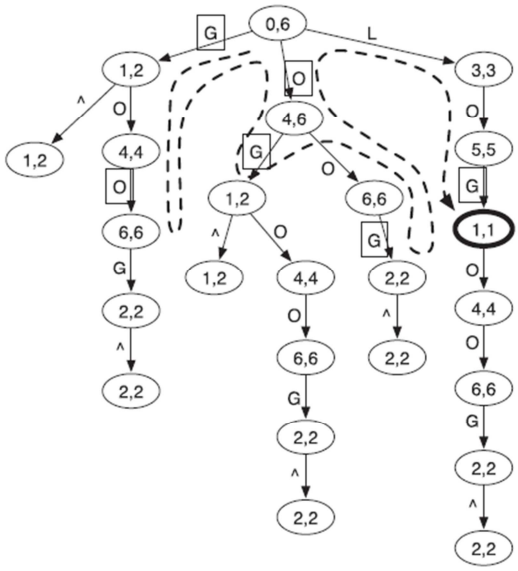


Figure 2 - חיפוש על גבי עץ רישות

החלק המעניין בתרשים זה הוא הקו המקווקו המראה את המסלול של האלגוריתם בחיפוש אחר המחרוזת 'LOL', תוך אפשור חוסר התאמה אחד. נשים לב **לחידוש הגדול שבאלגוריתם** זה כפי שהוא מצוין בנקודות הבאות:

1. הקו לא יורד לכל עומק העץ – האלגוריתם יודע להתמודד עם שגיאות, **וממשיך לרדת** במורד העץ גם לאחר שגיאה אחת. ומאידך, ברגע שישנן יותר מדי שגיאות (2 במקרה הזה) **החיפוש נעצר** והקו עולה בחזרה במעלה העץ.
2. רישות שלהן יש רישה משותפת, **נמצאים על אותו מסלול של האלגוריתם!** (חסכון בזמן ריצה).

האותיות על הצלעות המוקפות ריבוע,מסמנות חוסר התאמה ("שגיאה") לשאילתא בחיפוש. ההתאמה היחידה לחיפוש היא

הצומת המודגשת [1,1] המייצגת את המחרוזת 'GOL'.

הסבר: מדוע הצומת המודגשת [1,1] המייצגת את המחרוזת 'GOL'?
[1,1] מייצג טווח של אינדקסים, ובמקרה שלנו – אינקדס יחיד: 1.
נלך לטבלת ה-BWT, ואכן באינדקס 1, מופיעה המחרוזת 'GOL'.

אלגוריתם BWA ממוקבל

על מנת ליישם אלגוריתם זה בצורה טובה יותר – ניתן למקבל אותו.
לשם הבנת יתרון המקבול בצורה פשוטה, אם היה ברשותנו כלי למקבול בקנה מידה אינסופי (או לפחות גדול מ-m).
אזי, עבור כל בדיקת DNA הזמן שידרש לריצת האלגוריתם היה $\theta(|w|)$.
מכיוון שאין כלי שכזה בנמצא, אזי עבור "פקטור מקבול" t, זמן הריצה בפועל יהיה: $\frac{\mathrm{m}}{t}\theta(|w|)$

במעבד מרובע ליבות i7, בו השתמשנו, פקטור המקבול הוא מעט יותר מ-4 בזכות טכנולוגיית hyper-threading.
נדגיש כי ניתן ליישם את המחקר שהמתבצע בפרויקט זה גם במקבול על גבי כרטיסי מסך, שבהם פקטור המקבול יכול להגיע לכמה אלפים בודדים.

כאמור, אלגוריתם זה ודומיו הם המובילים בעולם ביו אינפורמטיקה בתחום התאמת קריאות על פני הגנום, וכבר נמצא בשימוש ע"י בתי החולים ששוכרים חוות שרתים לביצוע חישוב זה, ועדיין התהליך לוקח כיום שלם.
בפרויקט זה נחקור את יעול האלגוריתם של BWA-Alignment באמצעות מיקבולו על גבי מעבד מרובה ליבות, כך ש-10^9 הפעולות לא יתבצעו באופן סדרתי.

תיאור המערכת – מימוש BWA

ביישום הפרויקט השתמשנו בשפות הבאות:
1. #C - לבניית אב טיפוס, הוכחת התכנות, בדיקות יחידה וניסויי שיפור באלגוריתם, השוואת זמני ריצה.

ובמערכות ההפעלה הבאות:
1. Windows – לבניית מוצר עם ממשק גרפי להדגמת זמני הריצה של האלגוריתם.

פלטפורמות אלו נבחרו בעיקר על מנת לאפשר פיתוח מהיר יותר תוך שימוש בכלים קיימים המוכרים למפתח.

שימוש ברכיבים קיימים

- הפרויקט מבוסס על יעול של אלגוריתם קיים. לכן, אנו השתמשנו במודל המבוסס על המסמך המדעי של BWA[i].
- עבור כלי בדיקה, נשתמש ב-nunit.

מהי המערכת

המערכת היא תכנה שמבצעת חיפוש של מחרוזת קצרה בתוך מחרוזת ארוכה (מאוד), כאשר החיפוש יניב תוצאות גם אם נפלו שגיאות במחרוזת הקצרה, כך שהיא לא זהה במדוייק למקטע במחרוזת האורכה.

תהליכים ונתוני המערכת

במערכת ישנם שני תהליכים עיקריים:

1. Index - אינדוקס של הגנום – שמירת הגנום בצורה שניתן לגשת אליו בצורה יעילה. זו פעולה שיש לעשות אותה פעם אחת בלבד (ולא בכל בדיקה של חולה...).
2. Alignment – מציאת המיקום של הקריאות על פני הגנום.

המידע שקיים במערכת הוא:
1. הגנום האנושי המאונדקס (מידע סטטי).
2. קטעי דגימות DNA (הקלט של המערכת).

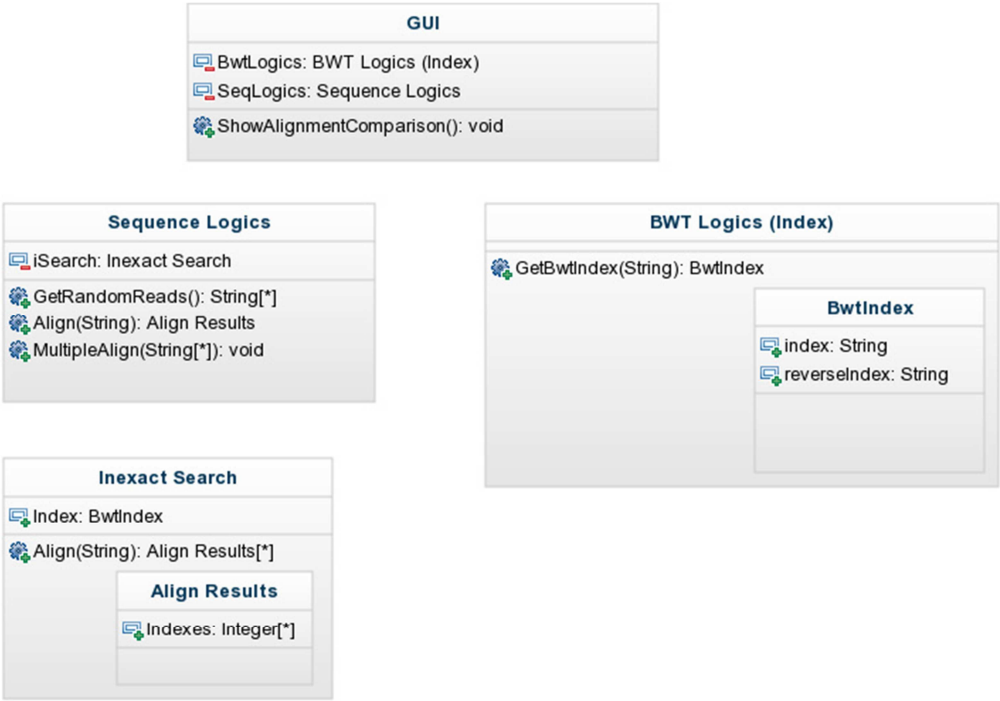
תיאור הכלים המשמשים לפתרון

ע"מ לייעל את מימוש האלגוריתם BWA השתמשנו במעבד מרובה ליבות (כולל טכנולוגית Hyper threading) בסביבת עבודה של Visual Studio, וספציפית השתמשנו בספריית Parallel של מייקרוסופט לשם מקבול מיטבי.

תיאור המימוש

(במאגר הקוד זמינים: **קבצים בינארים**[ii] **וקוד**[iii]):

לשם השוואת נתוני המקבול, מומשה מערכת גרפית היכולה ליצור אינדקס, ליצור מאגר דגימות הכוללות טעות לפי פרמטר הסתברותי, להשוות את הדגימות לאינדקס בצורה סדרתית / ממוקבלת, להשוות זמני ריצה ולשרטט את גרף זמני הריצה כתלות בפרמטרים שונים.



המערכת מכילה את המודולים הבאים:

a. ממשק משתמש כולל יכולות של:
   i. בחירת פרמטרים להשוואה וביניהם:
      - ביצוע חיפושים ממוקבלים / סדרתיים עם אורך דגימה משתנה.
      - ביצוע חיפושים ממוקבלים / סדרתיים עם מספר דגימות משתנה.
      - ביצוע חיפושים ממוקבלים עם נתונים ממויינים / לא ממויינים ועם מספר דגימות משתנה.
   ii. הצגת השוואת זמני ריצה ופלט טקסטואלי של השוואות BWA.
b. Sequence Logics – מודול לטיפול בענייני דגימות:
   i. מציאת התאמה של קריאה / מספר קריאות על פני הגנום (כולל חיפוש על דגימות שמחושבות בזמן ריצה).
   ii. יצירת מדגם דגימות המדמות דגימות מחולה, כולל פרמטרים כגון:
      - הסתברות לטעות בכל תו.
      - אורך הדגימה.
      - מספר הדגימות.
c. BWT Logics – הכנת האינדקס כולל:
   i. ביצוע טרנספורם BWT.
   ii. הכנת אינדקס של טקסט המדמה גנום אנושי (מבוסס על עץ סיפות).
d. Inexact Search - חיפוש של מחרוזת בודדת באינדקס.

החלק העיקרי בתוכנה מבצע את החיפוש של הדגימות מסעיף (a i) פעמיים, פעם בתהליכון אחד, ופעם עם ריבוי תהליכונים (מספר התהליכונים נקבע באופן אוטומטי ע"י ספריית Net. / ידנית ע"י המשתמש).

בדיקות

תכנון וכתיבת בדיקות הן חלק בלתי נפרד מכתיבת תוכנה. הן עוזרות לנו לגלות תקלות בשלב מוקדם ומאפשרות עדכון בסיס קוד קיים תוך סיכון מינימלי בפגיעה במערכת.
בתכנון הבדיקות ניסיתי להקיף את הנקודות שעלולות לגרום לבאגים בקוד, ולוודא שאחרי המקבול:

1. הפלט של הפונקציה אינו משתנה בעקבות המקבול.
2. זמן הריצה משתפר משמעותית (מטרת הפרויקט).

	סוג הבדיקה	שם הבדיקה	אופן הבדיקה	פרמטרים	תוצאה מצופה
1	יחידה	זיהוי מחרוזת זהה	נקח קטע באורך 35bp מתוך הגנום עצמו וניתן לאלגוריתם לחפש אותו.	קטע באורך 35bp, 0 טעויות מותרות.	Match באינדקס שממנו נלקח הקטע
2	יחידה	אי זיהוי מחרוזת עם n טעויות של תווים שהתחלפו.	נקח קטע באורך 35bp מתוך הגנום עצמו, נשנה n תווים בקטע, וניתן לאגוריתם לחפש אותו.	קטע באורך 35bp n-1 טעויות מותרות.	אין Match באינדקס שממנו נלקח הקטע
3	יחידה	אי זיהוי מחרוזת עם n טעויות של תווים שנמחקו.	נקח קטע באורך 35bp מתוך הגנום עצמו, נמחק n תווים בקטע, וניתן לאגוריתם לחפש אותו.	קטע באורך 35bp n-1 טעויות מותרות.	אין Match באינדקס שממנו נלקח הקטע
4	יחידה	זיהוי של מחרוזת עם n טעויות של תווים שהתחלפו.	נקח קטע באורך 35bp מתוך הגנום עצמו, נשנה n תווים בקטע, וניתן לאגוריתם לחפש אותו.	קטע באורך 35bp n טעויות מותרות.	Match באינדקס שממנו נלקח הקטע
5	יחידה	זיהוי של מחרוזת עם n טעויות של תווים שנמחקו.	נקח קטע באורך 35bp מתוך הגנום עצמו, נמחק n תווים בקטע, וניתן לאגוריתם לחפש אותו.	קטע באורך 35bp n טעויות מותרות.	Match באינדקס שממנו נלקח הקטע
6	אינטגרציה	השוואת פלט פונקציה ממוקבלת לפונקציה לא ממוקבלת	בהינתן גנום וסט של קריאות נבדוק שעבור אותם הפרמטרים הפונקציה הממוקבלת והלא ממוקבלת מוציאות את אותה התוצאה.		אותן תוצאות ב2 ההרצות.
7	קבלה	השוואת זמני ריצה של פונקציה ממוקבלת לפונקציה לא ממוקבלת	בהינתן גנום, פרמטרים, וסט של קריאות נבדוק שעבור שזמני הריצה של הפונקציה הממוקבלת טובה באופן משמעותי מהפונקציה הלא ממוקבלת.		אותן תוצאות ב2 ההרצות. עם זמן ריצה טוב יותר באופן משמעותי עם המקבול (משמעותי = פי 3)

- הבדיקות מתוארות במונחים כללים של "n" טעויות בגלל הקלות של שכפול בדיקות עם פרמטרים שונים באמצעות הספרייה nunit ושימוש ב- TestCase.
- כל אחת מהבדיקות הנ"ל מייצגת למעשה מספר בדיקות שונות:
  - עבור קטע מתחילת הגנום.
  - עבור קטע מאמצע הגנום.
  - עבור קטע מסוף הגנום.

באותו אופן כל מקטע טומן בחובו מספר בדיקות :
- עבור טעויות שנפלו בתחילת הקטע
- עבור טעויות שנפלו באמצע הקטע
- עבור טעויות שנפלו בסוף הקטע

ושוב, כל אחד מאלו מחזיק בתוכו בדיקות עבור:
- אות שהוחלפה.
- אות שהוחסרה

בסופו של דבר, עם כל ה- Test Cases **<u>קיימות 66 בדיקות למערכת</u>**.

ניסויים ומסקנות

התוצאות הבאות התקבלו של מחשב עם הנתונים הבאים:

- זכרון:
  - 16 GB DDR3 SDRAM

2. מעבד:
   - 2nd Gen Intel(r) Core™ i7-2670QM2
   - 2.2 GHz processor speed
   - Turbo Boost up to 3.1GHz
   - 8-way processing

עבור קלט:
- גנום באורך 2000 (זה לא אמור לשנות לזמן ריצה שתלוי אורך הקריאה...)
- אפשור ל 2 טעויות.
- סיכוי של 3% לטעות בכל תו בקריאה.

קיבלנו את התוצאות הבאות:

בדיקה: האם אורך הדגימות משפיע על פקטור המיקבול?

מסקנה:
המיקבול נותר יעיל פי 4-5 ללא תלות באורך הדגימה.

ציר X: אורך הדגימה
ציר Y: סה"כ שניות שנדרשו להשוואת 100 דגימות (ממוקבל / לא ממוקבל)

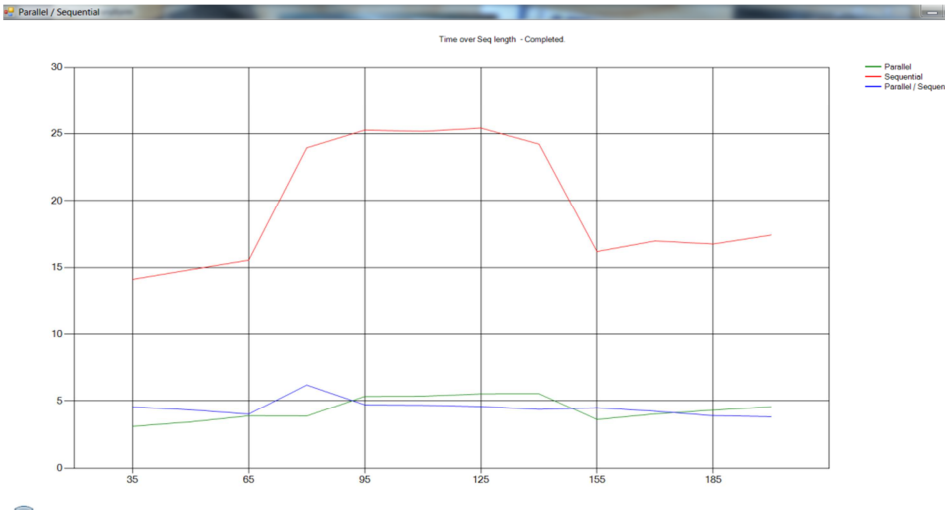


Chart 1 – ניתן לראות שהגרף הכחול נותר יציב סביב 4-5. כלומר המקבול נותן באופן יציב תפוקה טובה יותר פי 4-5 ללא תלות באורך הדגימות

בדיקה: האם מספר הדגימות משפיע על פקטור המיקבול?

מסקנה: המיקבול נותר יעיל פי 4-5 ללא תלות במספר הדגימות.

ציר X: מספר הדגימות
ציר Y: סה"כ שניות שנדרשו להשוואת X דגימות באורך 35bp (ממוקבל / לא ממוקבל)

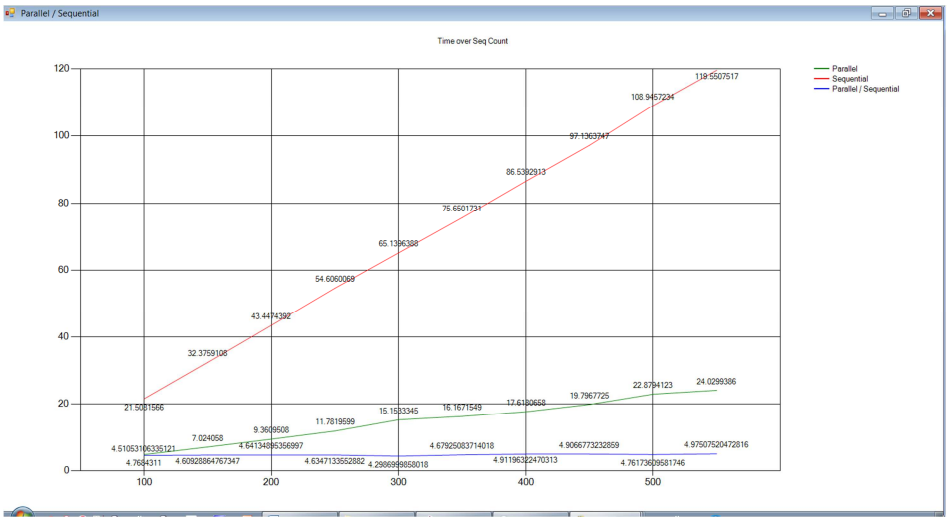


Chart 2 - ניתן לראות שהגרף הכחול נותר יציב סביב 4-5. כלומר המקבול נותן באופן יציב תפוקה טובה יותר פי 4-5 ללא תלות במספר הדגימות

בדיקה: האם מיון הדגימות לפני ההתאמה משפיע על ביצוע ההתאמה באופן ממוקבל?

מסקנה: המיון מסייע באופן עקבי לזמן הריצה, אך רק בפקטור של כ 1%-1.5%

ציר X: מספר הדגימות
ציר Y: סה"כ שניות שנדרשו להשוואת X דגימות באורך 35bp בתהליך ממוקבל (ממויין / לא ממויין)

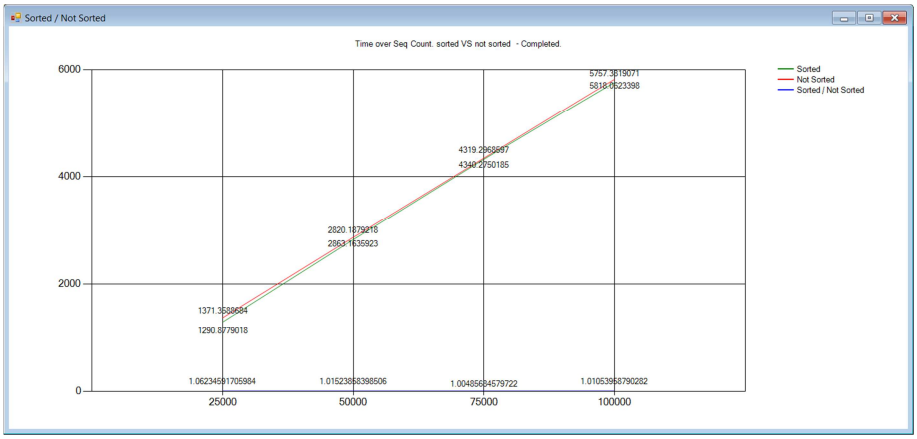


Chart 3 – ניתן לראות שהמקבול נות תוצאות טובות יותר, אך רק בפקטור של כ- 1.5%

בדיקות נוספות:

1. נבחנו שתי אפשרויות למקבול:
   - מקבול התהליך - כל קריאה תרוץ על פני תהליכון אחד משלה.
   - מקבול האלגוריתם – כל קריאה תמופה באמצעות מספר תהליכונים, שכל אחד מהם יהיה אחראי לחלק מהרקורסיות שבאלגוריתם.

מסקנות

1. אורך הדגימות אינו משפיע על זמן הריצה של האלגוריתם הממוקבל ביחס לאלגוריתם הסדרתי.
2. מיון / אי מיון הדגימות אינו משפיע באופן ניכר על זמן הריצה של האלגוריתם הממוקבל ביחס לאלגוריתם הסדרתי.
3. הרקורסיה של האלגוריתם מעמיסה על הזכרון ובעייתית בהעברת המימוש לכרטיס מסך.
4. לאחר בדיקה והשוואות נראה שהתוצאות הטובות ביותר מתקבלות ע"י מקבול התהליך, באופן שבו כל דגימת DNA מקבלת תהליכון. מספר סיבות לתוצאה הנ"ל:

a. עדיף לתת לכל תהליכון מספיק עבודה כך שמערכת ההפעלה תשקיע את המשאבים בעבודה עצמה ולא בניהול תהליכונים.
b. מקבול שכזה מונע בעייתיות של תהליכונים שסיימו את עבודתם וממתינים לתהליכונים אחרים. במקבול התהליך – אין שום תלות בין התהליכונים.
c. הרצת כל קריאה על פני תהליכון יחיד עוזרת מבחינת העומס על הזכרון. סטטיסטית – ישנן פחות רקורסיות "עמוקות" באותו הזמן.

לסיכום נראה שהאלגוריתם הממוקבל נשאר יציב, מבחינת סדרי הגודל של היעול, גם כאשר מעמיסים עליו דגימות ארוכות.
מאידך, נראה שמיון הדגימות, לא עוזר לשפר את זמן הריצה באופן ניכר.

במחשב מרובע ליבות i7 קיבלנו זמן ריצה טוב פי 4-5 בעזרת האלגוריתם הממוקבל (הערה: זמן ריצה טוב ביותר מפי 4 על מחשב עם 4 ליבות התאפשר בזכות טכנולוגיית Hyper Threading הממומשת בחומרה של מעבדי i7)

נציין, שאם האלגוריתם ימוקבל על גבי כרטיס מסך, המיון צפוי לעזור באופן משמעותי בעקבות הורדה דרסטית של הBranching(סנכרון תהליכונים) המאפיין תכנות מקבילי שכזה.
קיווינו למקבול זה במסגרת הפרויקט, אך החומרה הנדרשת סופקה רק ביום מועד ההגשה.

השוואה לפתרונות בספרות

בנוסף לאלגוריתמים שתוארו בסעיף "תיאור הבעיה" קיימים הפרויקטים הבאים:

1. פרויקט הקוד הפתוח BarraCUDA[iv], המיישם את אלגוריתם BWA בעזרת תכנות מקבילי על גבי כרטיס מסך. הפרויקט בתהליכי פיתוח כ- 6 שנים, כחלק מהמעבר לתכנות על גבי כרטיס מסך, הוסרה האפשרות לחפש מחרוזות עם תווים שהוסרו.
2. פרויקט BWA – הפרויקט המוביל בתחום הביואינפורמטיקה שמימש את שלגוריתם הBWA – עם תאימות ללינוקס בלבד. זמני הריצה של BWA טובים משלנו, אך הקוד נכתב בצורה של תיקון על גבי תיקון ויעול על גבי יעול במהלך שנים, וללא תיעוד – כך שקשה מאוד להכנס אליו.
3. BowTie[1] – אלגוריתם מקביל המתבסס על אותם עקרונות כמו BWA.

סיכונים שהתממשו

#	הסיכון	מענה
1	האלגוריתם הקיים של BWA היה כתוב בצורה לא מתועדת ועם ייעול על גבי ייעול כך שהיה קשה מאוד לקשר אותו למסמך המדעי של BWA	כתבנו את הקוד בעצמנו ומקבלנו אותו.
2	לא קיבלנו מחשב עם כטיס מסך ע"מ למקבל עליו את הקוד.	מיקבלנו את הקוד על CPU

רשימת ספרות

1. להכנת מסמך זה נעזרתי ב:
   a. ספרות עם חומר שרלוונטי לפרויקט:
      i. המסמך המדעי המתאר את צורת העבודה של האלגוריתם: http://www.math.pku.edu.cn/teachers/xirb/Courses/biostatistics2013/Bioinformatics-2009-Li-1754-60.pdf
      ii. הרצאות בביואינפורמטיקה פרקים 17-21. https://www.cs.cmu.edu/~ckingsf/bioinfo-lectures/.
      iii. שאלות נפוצות בקשר ל- BWA: http://bio-bwa.sourceforge.net/
      iv. השוואה בין שיטות NGS: http://en.wikipedia.org/wiki/DNA_sequencing#Next-generation_methods
   b. בויקיפדיה:
      i. Cuda: http://he.wikipedia.org/wiki/CUDA
      ii. Short-Read Sequence Alignment: http://en.wikipedia.org/wiki/List_of_sequence_alignment_software#Short-Read_Sequence_Alignment
   c. האיציקלופדיה הקטנה של המדעים מאת רן לוי (מהדורה ראשונה 2011).

[1] http://bowtie-bio.sourceforge.net/index.shtml

נספחים

יומן הפרויקט:

https://github.com/turner11/BWA-Final_Project/blob/master/Documents/%D7%94%D7%92%D7%A9%D7%AA%20%D7%A4%D7%A8%D7%95%D7%99%D7%A7%D7%98/%D7%99%D7%95%D7%9E%D7%9F.docx?raw=true

מאגר קוד: https://github.com/turner11/BWA-Final_Project

רשימת משימות פתוחות: **https://github.com/turner11/BWA-Final_Project/issues?q=is%3Aopen+is%3Aissue**

רשימת משימות סגורות: https://github.com/turner11/BWA-Final_Project/issues?q=is%3Aissue+is%3Aclosedhttps://github.com/turner11/BWA-Final_Project/issues?q=is%3Aissue+is%3Aclosed

Abstract

This is a research project which aims for exploring the BWA algorithm and examining the possibilities for making it parallel.

The BWA algorithm is an algorithm for finding the location of short strings within a large constant string. Its most common use is in the bioinformatics filed, particularly aligning DNA samples to the human genome.

The DNA, taken from a healthy subject, is represented by a long sequence of characters (e.g. GCATGCTAGCTAGCTCGATGCTACTGACCGTCAG…) and is used as a reference for aligning future samples (this is possible as about 99.9% of human DNA is common to all human beings.)

When a subject is being genetically diagnosed, his DNA is sampled and fragmented into many (a few billions) short (35-200bp) samples.

This is a challenging task due to the fact that the samples are not identical to the reference DNA (The samples might contain mutations / machine read errors).

Due to the large amount of samples, the genome length, and the complexity that depends on genome length, "traditional" substring searches are not capable of handling this task in a reasonable timeline. For example:

1. The naïve search: $\theta(\mathrm{n}^2 \cdot |w|)$
2. KMP: $\theta(n^2)$

Where:

$|w|$ - sample length

n – The genome length.

This calculation is using the (correct) assumption that the number of samples is at the same magnitude of the genome length.

Unlike the above, BWA algorithm aligns samples with a $\theta(|w|)$ complexity (independent of genome length!) and can handle mismatch as well.

In this project we have implemented the algorithm and parallelized it in a manner that every sample is being processed by a single thread.

In addition, a program with user interface was implemented including generating samples with errors, aligning the samples and producing graphical results for the alignment benchmarks.

For this optimization we have tested and benchmarked a few items:

1. The impact of the length of samples on calculation run time.
2. The impact of the number of samples on calculation run time.
3. The impact of sorting the samples on calculation run time.
4. The impact of making the process parallel on multiple cores.

After testing and benchmarking, it seems that the best results are achieved when making the process parallel in a manner that every sampled DNA sequence is aligned using a separate thread (i.e. rather than having the alignment algorithm being paralleled internally).

This type of optimization remains stable independent of the sample length/ number / sort.

Software Engineering Department

Paralleling BWA-Aligner

Aligning DNA sequence along the human genome

By

Avi Turner

July 2015

Software Engineering Department

Paralleling BWA-Aligner

Aligning DNA sequence along the human genome

By

Avi Turner

Supervisor: Dr. Hassin Yehuda

[i] http://sourceforge.net/projects/bio-bwa/?source=navbar

[ii] https://github.com/turner11/BWA-Final_Project/blob/master/Files/BWA-Aligner.zip?raw=true

[iii] https://github.com/turner11/BWA-Final_Project/tree/master/Code

[iv] http://seqbarracuda.sourceforge.net/