

המחלקה להנדסת תוכנה

### פרוייקט גמר בהנדסת תוכנה – תשע"ה

**הצעת פרויקט**

**Optimizing BWA-Aligner**

**יעול חיפוש מיקומם של מקטעי DNA על פני הגנום האנושי.**

|  |
| --- |
| **נ ת ו נ י ם א י ש י י ם** |
| **אבי טרנר**  **ת.ז: 039991708**  **דוא"ל:** [**avi.turner111@gmail.com**](mailto:avi.turner111@gmail.com) |
| **ארגון**  **עזריאלי – המכללה האקדמית בירושלים** |
| **מנחה אקדמי**  **ד"ר יהודה חסין** |

# תיאור מסגרת הפרויקט

הפרויקט הוא פרויקט מחקרי, על מנת לאפשר הבנה טובה יותר של הבעיה והפתרון המוצע, נפתח בתיאור קצר של הרקע ושל מונחים הקשורים לפרויקט.

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

המבנה של הDNA 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 שכ99.9% מהDNA של כל בני האדם משותף למרות אבני הבניין המועטות והפשוטות– וזוהי תכונה קרדינלית לפרויקט זה.

אם כן, את המידע המגולם בDNA ניתן לייצג כמחרוזת של הבסיסים המרכיבים אותו, וכך נתייחס אליו בפרויקט זה (לדוג': TGACCGTCAG....)

הDNA מורכב מכ שלבים. במונחים דיגיטלים, המידע הגלום בDNA שווה ערך למעט יותר מ 1.6 GB (או כ2 דיסקים...)

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

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

1. לקיחת דגימת DNA.
2. השוואת הDNA לDNA של אדם בריא.
3. איתור מוטציות בDNA.
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 ע"מ ליעל את החישוב הארוך של השוואה.

עם זאת, גם כיום שלב זה לוקח כיום שלם – פרויקט זה יתמקד בייעול שלב זה.

## הגדרות – שפה משותפת לפרויקט

* – קריאה – קטע מהDNA של חולה שנדגם לצורך בדיקה.
* – רפרנס - הגנום שנשתמש בו להשוואה לכל הקריאות.
* – base pair - זוג בסיסים המהווים שלב ב"סולם הDNA". משמש כמידת אורך של קריאה.
* - אורך של קריאה. באופן טיפוסי הקריאות הן באורך של bp 100- bp 200 אך ה"לקוח" הספציפי של תוצר הפרויקט, משתמש בקריאות באורך bp35.
* – המחרוזת על גבי באורך , המתחילה באינדקס .
* – הקריאה שהחלה במקום ה ב X
* – אורכו של X ביחידות bp -

# תיאור הבעיה

***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 בSequencing גדל בצורה אקספוננציאלית בעיקר בשירות מעבדות קליניות ומחקר בכל רחבי העולם, כשהמתשמש הבולט מביניהם היה פרויקט מיפוי הגנום האנושי שהסתיים ב2003.*

*בפרויקט מיפוי הגנום האנושי, נעשה שימוש בSanger Sequencing שפותח ב1977 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 הDNA נחתכים ל"אורך" בגלל המבנה של הDNA , ניתן לשחזר כל צד מהDNA ע"י בן זוגו שנחתף. 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 הNGS 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.*

**הקדמה:**

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

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

הIllumina עובדת במתודולוגיה הנקראת NGS (ראה מסגרת להרחבה).

**תיאור פורמאלי**:

בהינתן:

1. – רצף של תווים באורך .
2. – אוסף של תווים בעלי אורך , .
3. – המחרוזת על גבי באורך , המתחילה באינדקס .
4. – מרחק העריכה המותר בהשוואת 2 מחרוזות, כך שהן יחשבו עדיין "דומות".

אזי עלינו למצוא עבור כל את אוסף כל ה הנמצאים במרחק עריכה ממנו.

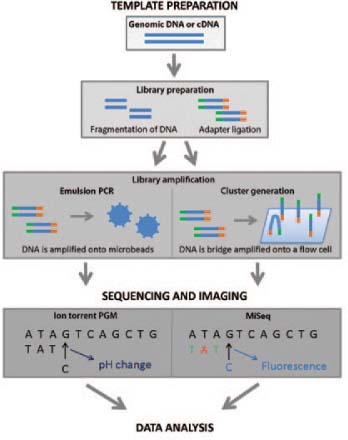
**בפרויקט שלנו**:

1. – רצף הגנום האנושי, באורך של ~.
2. – אוסף של מקטעי ה DNA שנדגמו ע"י הIllumina, כל דגימה היא באורך של ,

.

כלומר סה"כ הדגימות מכסים את כל הגנום כ30 פעמים (בלי לקחת בחשבון חפיפות).

1. – פרמטר למרחק עריכה מותר – באופן טיפוסי - 2.



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

1. **האלגוריתם הנאיבי**:

עבור כל נבדוק עבור כל האם

יעילות האלגוריתם: .

מכיוון ש גדול מאוד, אלגוריתם זה אינו מעשי במקרה שלנו.

1. **KMP**

אלגוריתם המנצל את מבנה התבנית על מנת ליעל את החיפוש.

יעילות האלגוריתם:

למרות שאלגוריתם זה הוא יעיל למדי, עקב העובדה ש שלנו גדול מאוד, ומפני שיש צורך לבצע אלגוריתם זה עבור כ"א מהקריאות (סדר גודל של כמה מליונים), גם הוא לא טוב מספיק.

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.

אם לא נפלה אף שגיאה – ישנה מחרוזת 1 להשוואה - .

אם נפלה שגיאה אחת – ישנם 100 מחרוזות להשוואה .

אם נפלו 2 שגיאות – ישנם 4,950מחרוזות להשוואה .

סה"כ, בהנחה של עד 2 שגיאות, עברנו מקריאה אחת באורך 100 ל5,051 מחרוזות באורך 100 שנצטרך להשוות. מכיוון שמלכתחילה יש לנו כ3,000,000,000 מחרוזות כאלו, ברור שחיפוש שכזה אינו ישים עבור מידע מסדר גודל של הגנום האנושי.

**הערה:** 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 בקריאה של המכונה הדוגמת.

על מנת להתגבר על בעיות כגון אלו, בשלב הדגימה לוקחים המון דגימות – בכמות כזו שסטטיסיטית כל מקטע של הDNA נדגם מספר פעמים. דבר זה עוזר כדי לוודא שאכן כל הDNA נדגם (בסבירות גבוהה) וגם מחפה על טעויות בקריאה (לא סביר שתהיה טעות קריאה של המכונה באותו המקום בכל הדגימות).

# תיאור הפתרון

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 הDNA של החולה, הדגימות נחתכות ל- 2 ויש צורך למצוא התאמה בין 2 חלקי הדגימה.

שלב ה *Alignment* (2), שבו יתמקד 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: (לא תלוי באורך באורך הגנום!).

בפועל, יש לאלגוריתם זה עלויות נוספות:

* Pre Processing :
* את הפעולה מבצעים על מספר גדול מאוד של קריאות, .

ולכן, סה"כ יעילות הלגוריתם היא

כיום, למרות השימוש באלגוריתם זה, התהליך לוקח זמן רב (כיום שלם) עקב ריבוי הקריאות ().

כאמור, אלגוריתם זה כבר נמצא בשימוש ע"י בתי החולים ששוכרים חוות שרתים לביצוע חישוב זה, ועדיין התהליך לוקח כיום שלם.

בפרויקט זה ניעל, את האלגוריתם של BWA-Alignment באמצעות מיקבולו על גבי מאיץ גרפי כך ש הפעולות לא יתבצעו באופן סדרתי.

לשם הבנת יתרון המקבול בצורה פשוטה, אם היה ברשותנו כלי למקבול בקנה מידה אינסופי (או לפחות גדול מ).

אזי, עבור כל בדיקת DNA הזמן שידרש לריצת האלגוריתם היה .

מכייון שאין כלי שכזה בנמצא, אזי עבור "פקטור מקבול" , זמן הריצה בפועל יהיה:

## המחשה ראשונית של שלבים האלגוריתם

## 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 התוים האחרונים של הרשומות נותן לנו את "מחרוזות BWT" b[i]- lo$oogg.

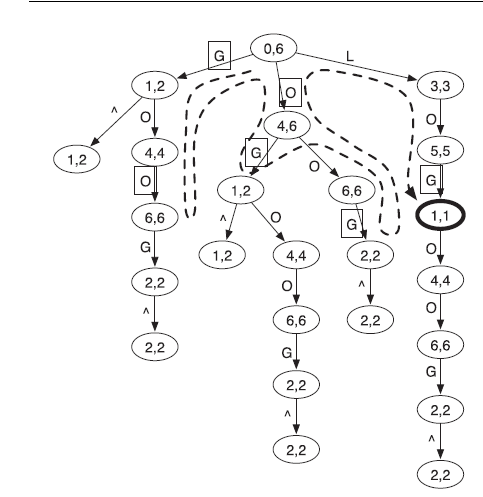
וזהו למעשה האינדקס שנשתמש בו.

(ניתן לעיין בנוגע ל טרנספורם BWT – זהו נושא שלם בפני עצמו).

**נמחיש את צורת העבודה של האלגוריתם באמצעות עץ רישות.**

**הערה**: מבלי להוכיח זאת, עץ רישות שקול למערך סיפות (לצורך אינטואיציה – עץ הרישות של זהה לעץ הסיפות של , 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.

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 הBWT, ואכן באינדקס 1, מופיעה המחרוזת 'GOL'.

## תיכון ראשוני:

# לשם יישום הפרויקט נשתמש בשפות הבאות:

# C# - לבניית אב טיפוס, הוכחת התכנות, בדיקות יחידה וניסויי שיפור באלגוריתם.

# C++ - שימוש במעטפת קיימת של אלגורית BWA, קריאות לקוד Cuda.

# Cuda – מימוש החלק המקבילי בפרויקט ע"ג מאיץ גרפי.

# במהלך הפיתוח נשתמש במערכות ההפעלה הבאות:

# Linux – מערכת ההפעלה שעליה פרויקט הBWA הנכחי יכול לרוץ

# Windows – לבניית אב טיפוס וייתכן שהמוצר הסופי יעבוד על גבי מערכת הפעלה זו (תלוי בהיקף העבודה שידרש להמרת קוד קיים כך שיוכל לרוץ במערכת הפעלה זו).

## שימוש ברכיבים קיימים:

* הפרויקט מבוסס על יעול של אלגוריתם קיים. לכן, אנו נשתמש ברכיבי הפרויקט של BWA[[1]](#endnote-1).
* עבור כלי בדיקה, נשתמש בnunit.

## מהי המערכת

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

## תהליכים ונתוני המערכת

במערכת ישנם שלושה תהליכים:

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.
3. Pairing – כחלק מקריאת הDNA של החולה, הדגימות נחתכות ל- 2 ויש צורך למצוא התאמה בין 2 חלקי הדגימה.

בפרויקט זה נממש את שלב ה Alignment.

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 נשתמש במאיץ גרפי ובשפת C for Cuda בסביבת עובדה של Visual Studio / Eclipse.

התכנות בסביבת CUDA, ובעיקר בשפת התכנות C for CUDA, מורכב משני אלמנטי תוכנה עיקריים[[2]](#endnote-2):

1. הקוד המתמטי-מדעי, הנכתב בשפת C for CUDA ומכיל את הגרעין (Kernels) החישובי.
2. קוד המעטפת (בדרך כלל בשפת C) אשר אחראי להקצאות ושחרור זיכרון, העתקות יחידות מידע אל ההתקן ומן ההתקן חזרה אל המחשב המארח, והפעלת גרעיני החישוב.

סביבת הפיתוח כוללת את המרכיבים הבאים:

1. מהדר (Compiler)
2. ממשק תוכנה (סט קובצי C header files) וספריות תוכנה.
3. מנהל התקן (driver)
4. סביבת הרצה: CUDA תומכת במערכות ההפעלה: Microsoft Windows, לינוקס ו-Mac OS.

# תיאור אב הטיפוס שמומש

# בשלב זה מימשנו 2 אבי טיפוס:

אבטיפוס לאלגוריתם

# (במאגר הקוד זמינים: [קבצים בינארים](https://github.com/turner11/BWA-Final_Project/blob/master/Files/Prototype.zip?raw=true)[[3]](#endnote-3) [וקוד](https://github.com/turner11/BWA-Final_Project/tree/master/Code)[[4]](#endnote-4)):

# האבטיפוס מכיל את הרכיבים הבאים:

# ביצוע טרנספורם BWT (חלק מתהליך האינדוקס)

# הכנת אינדקס של טקסט המדמה גנום אנושי (מבוסס על עץ סיפות).

# חיפוש של מחרוזת בודדת באינדקס.

# יצירת מדגם דגימות המדמות דגימות מחולה, כולל פרמטרים כגון:

# הסתברות לטעות בכל תו.

# אורך הדגימה.

# מספר הדגימות.

# חיפוש של כל אחת מהדימות מסעיף (d) באינדקס.

# החלק העיקרי של אב הטיפוס מבצע את החיפוש של הדגימות מסעיף (d) פעמיים, פעם בתהליכון אחד, ופעם עם ריבוי תהליכונים (מספר התהליכונים נקבע באופן אוטומטי ע"י ספריית .Net).

# לשם השוואה, על מחשב עם הנתונים הבאים:

* זכרון:
  + 8GB DDR3 SDRAM
* מעבד:
  + 2nd Gen Intel(r) Core™ i7-2670QM2
  + 2.2GHz processor speed
  + Turbo Boost up to 3.1GHz
  + 8-way processing

עבור קלט של:

* 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:

* בטבלה היחסית יש לקחת בחשבון שהחישובים נעשו על גבי CPU, וסביר ששכל ששהעומס עולה מערכת ההפעלה לא יכולה להפנות את כל המשאבים הנצרכים לחישוב.

# מסקנות

לאחר ניתוח של הקוד, נראה שהוא מסובך להבנה בצורה יוצאת דופן. הוא עבר כנראה הרבה ייעולים, הן של השפה (C), והן של ה אלגוריתם (BWA), וכל זאת ללא שום תיעוד.

בין הנקודות שעלו מהקוד הנסקר:

1. הקוד מבצע רקורסיה ע"י ניהול מחסנית משל עצמו.

כתוצאה, התקבלו ההחלטות הבאות:

1. בשלב ראשון אנו נתחיל במקבול של הפונקציה שעושה את העבודה המשמעותית (פונקצית Gap), מבלי להבין אותה.
2. לאחר שתהיה לנו גרסה ממוקבלת ועובדת, נבצע Benchmarking על מנת לבדוק עד כמה התייעלה הלולאה שקוראת לפונקציה זו.
3. עם נתוני הBenchmarking מהסעיף הקודם, נשנה את הפרמטרים כדי לנסות ולקבל תוצאות טובות יותר.
4. במידה והתוצאות לא יהיו מספיק טובות, נעמיק יותר בחקר הפונקציה, ואם יהיה צורך, נממש אותה בעצמנו עם קוד פחות יעיל אך שניתן למיק.

# תכנון בדיקות

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | סוג הבדיקה | שם הבדיקה | אופן הבדיקה | פרמטרים | תוצאה מצופה |
| 1 | יחידה | זיהוי מחרוזת זהה | נקח קטע באורך bp35 מתוך הגנום עצמו וניתן לאלגוריתם לחפש אותו. | קטע באורך bp35, 0 טעויות מותרות. | Match באינדקס שממנו נלקח הקטע |
| 2 | יחידה | אי זיהוי מחרוזת עם n טעויות של תווים שהתחלפו. | נקח קטע באורך bp35 מתוך הגנום עצמו, נשנה n תווים בקטע, וניתן לאגוריתם לחפש אותו. | קטע באורך bp35 n-1 טעויות מותרות. | אין Match באינדקס שממנו נלקח הקטע |
| 3 | יחידה | אי זיהוי מחרוזת עם n טעויות של תווים שנמחקו. | נקח קטע באורך bp35 מתוך הגנום עצמו, נמחק n תווים בקטע, וניתן לאגוריתם לחפש אותו. | קטע באורך bp35 n-1 טעויות מותרות. | אין Match באינדקס שממנו נלקח הקטע |
| 4 | יחידה | זיהוי של מחרוזת עם n טעויות של תווים שהתחלפו. | נקח קטע באורך bp35 מתוך הגנום עצמו, נשנה n תווים בקטע, וניתן לאגוריתם לחפש אותו. | קטע באורך bp35 n טעויות מותרות. | Match באינדקס שממנו נלקח הקטע |
| 5 | יחידה | זיהוי של מחרוזת עם n טעויות של תווים שנמחקו. | נקח קטע באורך bp35 מתוך הגנום עצמו, נמחק n תווים בקטע, וניתן לאגוריתם לחפש אותו. | קטע באורך bp35 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 פלט מתכנת BWA ממוקבלת ולא ממוקבלת | בהינתן גנום, פרמטרים, וסט של קריאות נבדוק שעבור שהתוצאה של הרצת התכנה זהה עבור התנה המקורית והממוקבלת. |  | 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 3 בדיקות שונות:
  + עבור קטע מתחילת הגנום.
  + עבור קטע מאמצע הגנום.
  + עבור קטע מסוף הגנום.

כך שלמעשה ישנן פה 15 בדיקות יחידה עוד לפני השימוש בTestCase

בתכנון הבדיקות ניסיתי להקיף את הנקודות שעלולות לגרום לבאגים בקוד, ולוודא שאחרי המקבול הפלט והממשק עם הפונקציה עם התכנה לא משתנה, אך זמן הריצה משתפר משמעותית (מטרת הפרויקט).

# השוואה לפתרונות בספרות

בנוסף לאלגוריתמים שתוארו בסעיף "תיאור הבעיה" פרויקט הקוד הפתוח BarraCUDA[[5]](#endnote-5) יישם גם כן את אלגוריתם BWA בעזרת תכנות מקבילי. בשלב זה אנו נמנעים מלהסתכל בצורת המימוש שלו ע"מ לשמור על ראש פתוח לרעיונות ייעול משלנו.

# תכנון הפרויקט

## לוח זמנים מתוכנן

|  |  |  |
| --- | --- | --- |
| 26.11.14 | בוצע? | הגשת הצעת הפרויקט |
| 15.12.14 | ✓ | אב טיפוס מוכן. |
| 22.12.14 | ✓ | הבנת פרמטרים מפרויקט BWA :   1. ערך ברירת מחדל של מספר השגיאות המותר. 2. בדיקה על אב טיפוס - תלותת של זמן ריצה ב:    1. אורך הקלט.    2. מספר שגיאות מותר. |
| 16.1.15 | ✓ | הרצת קוד BWA לא ממוקבל מהפרויקט הקיים. |
| 23.1.15 | ✓ | איתור הקוד שנמקבל. |
| 20.3.15 |  | מיקבול ראשוני של הקוד. |
| 20.4.15 |  | ייעול הגדרות: מספר תהליכונים, גודל בלוק וכד' |
| 15.4.15 |  | סיום פיתוח. |
| 28.6.15 |  | הגשת הפרויקט |
| 28.7.15 |  | הצגת הפרויקט |

**הערה:** בגלל אופיו המחקרי של הפרויקט, נפח העבודה במונחים של כמות קוד היא קטנה ולכן הן Waterfall והן agile לא נמצאו מתאימים כמתודלוגיית עבודה. צווארי הבקבוק שאובחנו הם:

1. הכנת אבטיפוס.
2. מיקבול הקוד.

## טבלת סיכונים

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | הסיכון | חומרה | סבירות | מדד סיכון (סבירות X חומרה) | מענה אפשרי |
| 1 | המיקבול יתברר כלא אפשרי. | 5 | 1 | 5 | ננסה ליעל את האלגוריתם עצמו |
| 2 | המיקבול יהיה פחות יעיל ממקבול קיים. | 3 | 3 | 9 | נמקבל מה שאפשרי, וננסה ליעל את האלגוריתם. |
| 3 | חוסר הבנה של האלגוריתם. | 3 | 3 | 9 | נבצע בדיקות של זמן ריצה ותוצאות בסיוע של בדיקות יחידה לוודא נכונות גם בלי הבנה מעמיקה. אם יהיה צורך, נממש את הפונקציה בצורה פחות יעילה – אך הפיצוי לכך יהיה במיקבול. |
| 4 | מבני הנתונים שבשימוש הפונקציה הממוקבלת גדולים מדי בשביל המקבול על גבי כרטיס מסך | 5 | 2 | 10 | נכנס לעובי הקורה ונראה היכן אפשר להשמיט / לצמצם אוביקיטים (לדוגמא ע"י השמטת שדות שלא בשימוש |
|  |  |  |  |  |  |

# רשימת ספרות

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

# נספחים

**יומן הפרויקט:** <http://projects.jce.ac.il/moodle/mod/wiki/view.php?wid=1&title=%D7%99%D7%95%D7%9E%D7%9F+%D7%A4%D7%A8%D7%95%D7%99%D7%A7%D7%98&groupanduser=9-26>

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

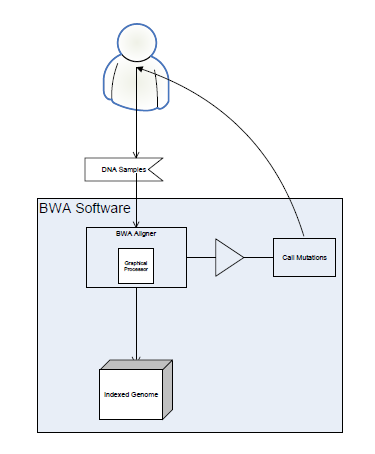
**רשימת משימות פתוחות:** [**https://github.com/turner11/BWA-Final\_Project/issues?q=is%3Aopen+is%3Aissue**](https://github.com/turner11/BWA-Final_Project/issues?q=is%3Aopen+is%3Aissue)

**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>

**דיאגרמת רכיבים \ הפצה (UML)**

באופן כללי, מכייון שהפרויקט הוא מחקרי, לא מצאתי דיאגרמות שיכולות לתאר אותו נאמנה.

הדיאגרמה הבאבה מסבירה באופן סכמטי כיצד ממקבלים את הקוד באמצעות GPU:



**טבלת דרישות (User Requirement Document)**

|  |  |
| --- | --- |
| מס' דרישה | תיאור |
| 1 | GPU – מאיץ גרפי מתוצרת NVidia |
| 2 | דגימות DNA מסוג NGS – עבור קלט למערכת. |
| 3 | מחשב עם התכנה מותכנת. |

1. <http://sourceforge.net/projects/bio-bwa/?source=navbar> [↑](#endnote-ref-1)
2. <http://he.wikipedia.org/wiki/CUDA> [↑](#endnote-ref-2)
3. <https://github.com/turner11/BWA-Final_Project/blob/master/Files/Prototype.zip?raw=true> [↑](#endnote-ref-3)
4. <https://github.com/turner11/BWA-Final_Project/tree/master/Code> [↑](#endnote-ref-4)
5. <http://seqbarracuda.sourceforge.net/> [↑](#endnote-ref-5)