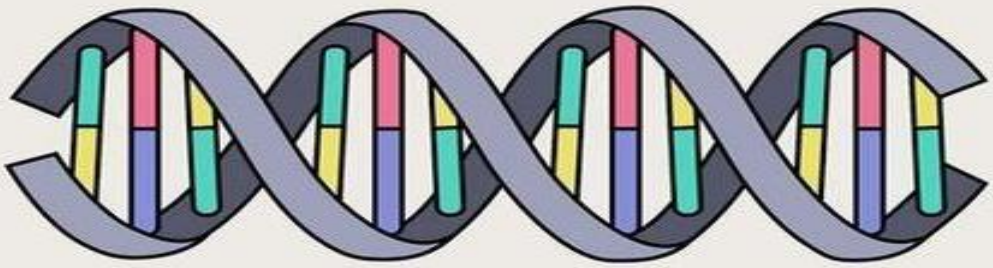
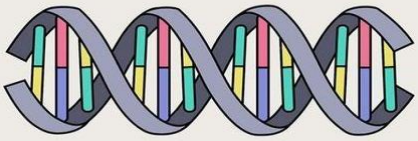
-GENEius-





שם :רעות כהן

ת.ז: 325858181

שם מנחה: המו' תמר זקס

שם בית ספר: בית יעקב רכסים

תאריך הגשה:ט"ו סיון,11/06/25

תוכן עניינים

הצעת פרויקט

סמל מוסד: 340695

שם מכללה: סמינר בית יעקב, רכסים

שם הסטודנט : רעות כהן

ת.ז. הסטודנט:325858181

שם הפרויקט: **Geneius**

מחלות גנטיות הן הפרעות הנגרמות על ידי חריגות בדנ"א של הפרט, או בירושה או שנרכשו באמצעות מוטציות חדשות

כדי להבין כיצד עלולות להיווצר מחלות גנטיות עלינו להבין כמה מושגי יסוד בסיסיים:

DNA:-הוא המולקולה המהווה את הבסיס לחיים על פני כדור הארץ. ניתן לחשוב על ה-DNA כעל ספר מתכונים ענק, שמכיל את כל ההוראות הדרושות לבניית ולתפעול של כל יצור חי.

DNA הוא חומר תורשתי שאגור בגרעין התא והוא חומר תורשתי שעובר בתורשה לילדינו.

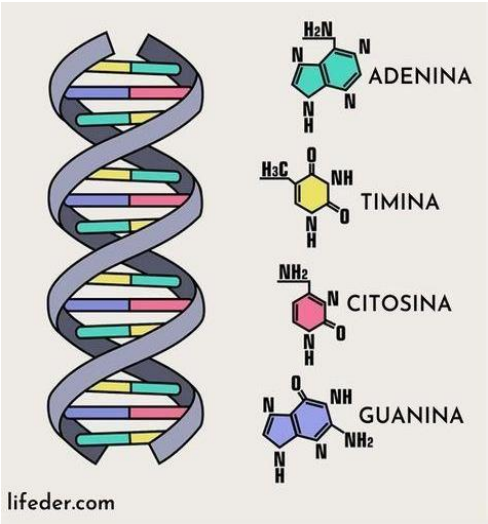
מה תפקידו של הDNA:

- **אחסון מידע תורשתי-** ה-DNA מכיל את כל המידע התורשתי שלנו, החל מצבע העיניים שלנו ועד למחלות שאנחנו עלולים לפתח.
- **העברת מידע לדורות הבאים**: כאשר תאים מתחלקים, עותקים של ה-DNA מועברים לתאי הבת, וכך המידע התורשתי מועבר מדור לדור.
- **יצירת חלבונים:** המידע הגנטי ב-DNA משמש כמתכון לייצור חלבונים, שהם המולקולות המבצעות את רוב העבודה בתאים שלנו.

מבנה ה-DNA:

- **סליל כפול**: ה-DNA בנוי כמו סולם כפול וסלילי.
- **נוקלאוטידים**: כל צד של הסולם מורכב מרצף של יחידות הקרויות נוקלאוטידים.
- **ארבעה סוגי בסיסים**: ישנם ארבעה סוגים שונים של בסיסים בנוקלאוטידים: אדנין (A), תימין (T), גואנין (G) וציטוזין (C).
- **זיווג בסיסים**: הבסיסים משני צידי הסולם תמיד מזווגים באופן ספציפי:

A תמיד מזווג עם T ו-G תמיד מזווג עם C .



החומר הגנטי ארוז במבנים הנקראים כרומוזומים לכל אדם 46 כרומוזומים ולכל כרומוזום יש 2 עותקים אחד שמקבל מהאם ואחד מקבל מהאב. כל כרומוזום מכיל אלפי גנים כל גן מכיל הוראה ספציפית לבניית חלבון מסוים צירוף של כל הגנים קובע את סך כל המאפיינים שלנו ואם המידע שבגן לקוי תהיה פגיעה במבנה ובתפקודו של החלבון שאותו הגן אחראי לייצרו.

הדנא הוא חומר תורשתי שעובר בתורשה לילדינו

הDNA שמור ב23 כרומוזומים לכל כרומוזום יש 2 עותקים-לכל גן בגוף יש שני עותקים ובזמן שיכפול התא יש חלוקה של הכרומוזומים ככל שגיל האמא גדול יותר כך יש יותר סיכוי לחלוקה לא שווה של הכרומזומים-ומזה נהיה תסמונות.

שני עותקים תקינים=בריא

שני עותקים פגומים=חולה

עותק תקין+ עותק פגום=נשא

-**נשא**: אדם שנשא בתוכו מוטציה בגן אחד, אך הגן השני תקין, הוא נשא. מכיוון שהגן התקין "משתלט" על התפקיד, הנשא בדרך כלל בריא לחלוטין, עם זאת, לנשאים יש פוטנציאל להעביר את הגן שעבר מוטציה לצאצאיהם.

-**מחלה גנטית:** היא מחלה שנגרמת כתוצאה משינוי בחומר התורשתי שלנו, ה-DNA. השינוי הזה, שנקרא מוטציה, יכול להשפיע על גן אחד או על מספר גנים, וכתוצאה מכך לגרום לתפקוד לקוי של חלבונים או של מערכות שלמות בגוף.

איך נוצרות מחלות גנטיות?

- תורשה: רוב המחלות הגנטיות מועברות בתורשה מההורים לצאצאים.
- מוטציות חדשות: לפעמים מוטציות יכולות להיווצר באופן ספונטני בתא או בתאים הראשונים של העובר.

מהם הסוגים השונים של מחלות גנטיות?

- מחלות מונוגניות: נגרמות על ידי מוטציה בגן יחיד. דוגמאות: סיסטיק פיברוזיס, אנמיה חרמשית.
- מחלות כרומוזומליות: נגרמות על ידי שינויים במבנה או במספר הכרומוזומים. דוגמה: תסמונת דאון.
- מחלות רב-גניות: נגרמות על ידי שילוב של גורמים גנטיים וסביבתיים. דוגמאות: סוכרת, מחלות לב.

-מוטציה היא שינוי ברצף הDNA. מוטציה היא למעשה "טעות דפוס" ברצף ה DNA

איך נוצרת מוטציה?

מוטציות יכולות להיווצר מסיבות שונות, למשל:

- שגיאות בשכפול ה-**DNA:** כאשר תא מתחלק, הוא משכפל את ה-DNA שלו. לפעמים, במהלך השכפול הזה, מתרחשות טעויות, ונוצרות מוטציות.
- חשיפה לחומרים מסרטנים: חומרים מסוימים, כמו קרינה או כימיקלים מסוימים, יכולים לגרום נזק ל-DNA וליצור מוטציות.
- תהליכים טבעיים: מוטציות יכולות להתרחש באופן ספונטני, ללא סיבה ברורה.

מה ההשפעה של מוטציות?

ההשפעה של מוטציות יכולה להיות מגוונת מאוד, החל משום השפעה כלל ועד לשינויים משמעותיים בתכונות של היצור החי.

- מוטציות ניטרליות: מרבית המוטציות אינן משפיעות כלל על התכונות של האדם.
- מוטציות מועילות: מוטציות מסוימות יכולות להקנות לאדם עמידות למחלות או יכולת להסתגל לסביבה משתנה.
- מוטציות מזיקות: מוטציות אחרות יכולות לגרום למחלות תורשתיות או להפרעות התפתחותיות.

ישנם סוגים שונים של מוטציות, ביניהם:

- מוטציות נקודתיות: שינוי בבסיס בודד בDNA- במבנה או במיקום של גן אחד.
- הוספה או מחיקה של בסיסים: הוספה או מחיקה של אחד או יותר בסיסים בDNA
- שינויים מבניים בכרומוזומים: שינויים גדולים יותר במבנה הכרומוזום, כמו שבירה, היפוך או החלפה של חלקים בכרומוזום.

חשיבות המוטציות:

מוטציות יוצרות את השונות הגנטית שמאפשרת ליצורים חיים להסתגל לסביבות משתנות ולהתפתח למגוון מינים.

עץ גנטי:

הוא כלי חשוב בביואינפורמטיקה שמשמש לניתוח ולהבנה של היחסיים הגנטי וזה יסייע לנו לקבוע את מקור המחלות הגנטיות הוא מאפשר לחקור את התפתחות האורגניזם ולחזות תוצאות שונות של שילובים גנטיים. הוא משתמש במונחים מתמטיים כדי לתאר את ההתפתחות של תכונות במערכת גנטית בעץ גנטי כל צומת מיוצג על ידי צומת שהוא כולל את השם או המזהה של האדם.העץ הגנטי מאורגן לדורות שונים כאשר כל דור מייצג רמה שונה של מוצא.בעצי משפחה גנטיים, אנשים החולקים כמות משמעותית של חומר גנטי מקושרים זה לזה, מה שמצביע על התאמה גנטית או קשר קרוב. זה יכול להיות מיוצג על ידי קווים מקשרים או על ידי קיבוץ אנשים קשורים יחד.

העץ עוזר להעריך את הסכנות הבריאותיות של כל אחד מבני המשפחה, שכן יש אפשרות שירשו כל מיני מחלות תורשתיות. ידיעה אודות מחלות ומצבים רפואיים של בני המשפחה

יכולים לאפשר לאדם להתעניין מראש בסכנות שעומדות בפניו ולקחת את הטיפול המתאים, האזהרות המתאימות לסכנות שייגרמו להתפתחות המחלה. גנוגרמים רפואיים יכולים לספק הערכה של מחלות תורשתיות ומחלות קלות שיכולות להתפתח במשפחה. בשביל להעריך מחלות תורשתיות ותבניות רפואיות במשפחה כדאי לציין דורות רבים אחורה, אמנם לפעמים בשביל להעריך סיכויי מחלה- 4 דורות אחורה מספק מידע נחוץ למדי.

-יעוץ גנטי: הוא תהליך שבו רופא גנטיקאי או יועץ גנטי מספק מידע ומענה על שאלות הקשורות למחלות תורשתיות. המטרה העיקרית של הייעוץ הגנטי היא להעריך את הסיכון של אדם או משפחה לפתח מחלה גנטית, ולספק להם את המידע הדרוש כדי לקבל החלטות מושכלות לגבי בריאותם ובריאות משפחתם.

כאשר שני ההורים הם נשאים של אותה הפרעה, יש סיכוי לילד שלהם לרשת שני עותקים של הגן שעבר מוטציה במקרים מסוימים, נשאים של מוטציות או מחלות גנטיות מסוימות יכולים לפתח תסמינים ולהיות מושפעים מהמצב בשלב מאוחר יותר בחיים. בעוד שנשאים בדרך כלל אינם מציגים תסמינים בעצמם, ישנם מצבים שבהם גורמים חיצוניים או אינטראקציות עם גנים אחרים יכולים לעורר את הביטוי של המחלה.

הסבירות של נשא לחלות בשלב מאוחר יותר בחיים תלויה בגורמים שונים, כולל המוטציה הגנטית הספציפית, אופי המחלה ונסיבות אינדיבידואליות. לחלק מהפרעות גנטיות יש ביטוי משתנה, כלומר התסמינים והחומרה יכולים להיות שונים בין נשאים. בנוסף, גורמים סביבתיים מסוימים, בחירות אורח חיים או שינויים גנטיים אחרים יכולים להשפיע על הסיכון לפתח תסמינים.

למה צריך ייעוץ גנטי?

- **אבחון מחלות גנטיות**: אם יש חשד שהאדם או מישהו מבני משפחתו סובל ממחלה גנטית, ייעוץ גנטי יכול לעזור באבחון מדויק.
- **הערכת סיכון**: אם יש היסטוריה משפחתית של מחלה גנטית, ייעוץ גנטי יכול לעזור להעריך את הסיכון של אותו אדם או של בני משפחתו לפתח את המחלה.
- **תכנון משפחה**: זוגות שמתכננים להתחתן ולהקים משפחה יכולים לפנות לייעוץ גנטי כדי להעריך את הסיכון להעברת מחלה גנטית לצאצאיהם.
- **הבנת תוצאות בדיקות גנטיות**: אם האדם ביצע בדיקה גנטית, ייעוץ גנטי יכול לעזור להבין את משמעות התוצאות ולהתמודד עם המידע.
- חשוב לציין שלא כל הנשאים יפתחו תסמינים של המחלה, ונשאים רבים חיים כל חייהם מבלי להיפגע. עם זאת, מומלץ לנשאים להישאר מעודכנים, לעבור בדיקות רפואיות סדירות ולדון בכל דאגה עם אנשי מקצוע בתחום הבריאות שיכולים לספק ייעוץ מותאם אישית על סמך הפרופיל הגנטי הספציפי שלהם.

בדיקות גנטיות-

בדיקות גנטיות הן בדיקות הנעשות על מנת לזהות ולאבחן זוגות הנמצאים בסיכון גבוה ללידת ילדים בעלי מחלות תורשתיות ומומים קשים. מחקרים רבים שנערכו בכל רחבי העולם במשך עשרות השנים האחרונות קובעים באופן חד משמעי כי לבדיקות הגנטיות המוכרות כיום, רמת יעילות ואמינות גבוהים במיוחד.הבדיקות נעשות על ידי נטילת דגימת דם.

היום יש לרפואה כלים טובים שמאפשרים להימנע מבעיות גנטיות. הבדיקות פשוטות לביצוע, אינן דורשות זמן רב מצד הנבדקים ונעשות באווירה מכילה ובתהליך ברור, כשכל שאלה מקבלת תשובה מראש.

מה מגלות הבדיקות הגנטיות?

לאחר נטילת דגימת הדם מהנבדק או הנבדקת וביצוע בדיקות גנטיות, אנו בודקים האם הנבדק הוא נשא של אחת מהמחלות התורשתיות הנבדקות. עובדת היותו נשא היא רק חצי מהסיפור. כאשר תתבקש התאמה מול המיועדת לנישואין, תינתן התשובה אם יש התאמה גנטית בין בני הזוג למחלות שנבדקו, אם לאו. כלומר, האם שני בני הזוג הללו נשאים לאותה מחלה ומצויים בסיכון גבוה ללידת ילדים עם אחת או יותר מהמחלות והמומים הנבדקים בבדיקות הגנטיות.

ידע הוא מרכיב חשוב בקבלת החלטות עתידיות והבדיקות הגנטיות הן משמעותיות. ככל שהנושא מתפתח, המודעות גדלה וההחלטה לבצע את הבדיקה המוקדמת, טבעית יותר.

מה רמת האפקטיביות של בדיקות גנטיות?

רמת האפקטיביות של בדיקות גנטיות הינה ברורה ומובהקת ללא כל ספק. במגזרים בהם עלתה רמת המודעות וכמות הנבדקים בבדיקה מוקדמת עלתה, ניתן לראות במקביל ירידה מובהקת וברורה בכמות הלידות של תינוקות עם אותם מחלות קשות. מנהלי בתי חולים ומחלקות המטפלות בתחום, מעידים על מחלקות שלימות שנסגרו בעקבות עליית המודעות וריבוי הבדיקות הגנטיות המוקדמות שמבוצעות באותה סביבה.

אפשר לראות את האפקטיביות של הבדיקות לא רק בקרב זוגות אשר יודעים שהם בקבוצות סיכון. היעילות של הבדיקות באה לידי ביטוי בשלבים שונים של מעגל החיים. כשמבצעים את הבדיקות טרם הנישואין, הן מאפשרות לקבל מידע גנטי רחב ואפקטיבי יותר ובהתאם לכך, כל נבדק ונבדקת יכולים לקבל החלטות מושכלות בשלב מוקדם בחיים.

אגודה למניעת מחלות גנטיות

'**דור ישרים**'- אגודה למניעת מחלות גנטיות הינה עמותה שמטרתה להפחית את שכיחותן של מחלות תורשתיות קשות בקרב האוכלוסייה היהודית.

אגודת דור ישרים מבצעת בדיקות סקר גנטיות ומיידעת בני זוג השוקלים להינשא האם הם בסיכון ללדת ילדים חולים במחלות גנטיות הנבדקות.

חשוב! 'דור ישרים' לא בודקת האם הנבדק חולה במחלה מסוימת או לא, 'דור ישרים' בודקת רק האם הנבדק נשא של גן פגום או לא, לצורך מתן תשובות התאמה גנטיות בין בני זוג לפני נישואין, כדי למנוע הולדת ילדים חולים.

מי צריך להיבדק בבדיקה גנטית?

בשונה מהגישה הרווחת בעולם, לקיים בדיקות גנטיות רק לזוגות המתכננים הריון או לאחר תחילת הריון, כאשר במקרים רבים מדובר בפעולה שמגיעה מאוחר מידי וכבר אין דרך למנוע את הבעיה, ב'דור ישרים' ממליצים לכל אדם צעיר עוד בטרם יצירת קשר זוגי לבצע את הבדיקה ולבדוק את ההתאמה לפני שהופכים לזוג נשוי. זאת במטרה למנוע את הבעיה לפני שהיא נוצרת.

המודעות לחשיבות המניעה יחד עם ההתפתחות הטכנולוגית בתחום הרפואי, מאפשרות להציע בדיקות מקדימות שיובילו לקבלת החלטות עתידיות טובות. מדובר בקבלת מידע מקדים שיסייע לבני הזוג לקבל את ההחלטות הנכונות מראש. זוהי הדרך הטובה ביותר לקבל את כל המידע הנדרש על הפוטנציאל הגנטי.

אלגוריתמים קיימים:

חיפוש מוטציות נקודתיות :(SNPs) אלגוריתמים אלו מחפשים שינויים בודדים בבסיס אחד ב.DNA-

BWA, Bowtie2-אלגוריתמים למיפוי רצפי DNA לקובץ התייחסות, הם מאוד מהירים ומשתמשים בטכניקות של חיפוש ביולוגי.

GATK -חבילה של כלים לניתוח וריאציות גנומיות.

VEP -כלי לחיזוי ההשפעה הפונקציונלית של וריאציות גנומיות.

DeepVariant -אלגוריתם מבוסס למידה עמוקה לזיהוי וריאציות גנומיות

PLINK -כלי לניתוח נתונים גנטיים המיועד לניהול וריאציות גנטיות

תיאור הפרויקט:

הפרויקט שלי עוסק בבדיקת רצף גנים של בני זוג (בתחילה מבת הזוג) אם יש

בעיה ברצף כמו שינוי בנוקלאוטיד או הכפלה לא במקום של רצף.

הרעיון של הפרויקט שלי מזכיר את הרעיון של ארגון דור ישרים שעליו פירטתי ברקע

בשלב ראשון: הפרויקט יקבל תחילה כקלט רצף של גנים של בת הזוג.

הפרויקט יבדוק:

1. האם הנבדקת בריאה-אם אכן כך הנבדקת תקבל הודעה מתאימה על מציאתה

בריאה מהמחלות שנבחנו-ולא נצטרך לבצע בדיקת נוספת על בן הזוג.

.2האם הנבדקת נשאית- (היא לא חולה אך קיימת בעיה ברצף ולכן היא נחשבת לנשא)

נערוך בדיקה נוספת על בן הזוג.

אם תמצא התאמה- כלומר אם הבן זוג הוא לא נשא של אותה המחלה-הנבדקים יקבלו הודעה מתאימה שהסיכון שיצא להם ילד שיחלה במחלה זו הוא נמוך מאוד

3.אם תמצא אי התאמה-כלומר גם בן הזוג הוא נשא של אותה המחלה הם יקבלו פלט-המלצה לגשת ליעוץ גנטי.

מאחר והילד עלול לרשת שני עותקים פגומים - הפרויקט יבדוק את מצב הנשאות

(זאת אומרת את הסיכויים שלו לחלות)על פי בחינת פרמטרים שונים.

מאחר ושני הוריו נשאים של מחלה גנטית רציסיבית (נשלטת(אבל הם לא חולים במחלה יש לצאצא סיכוי גבוה של אחוזים מסוימים לחלות (תלוי בסוג המחלה.(אם משהו מקרבת משפחתם של אותם נבדקים חולה- הסיכון למחלה עומד על אחוזים יותר גבוהים.

קלט: במקרה מספר 3 הפרויקט יקבל מבני הזוג:

נתונים על הרקע המשפחתי שלהם- האם קרוב משפחה או כמה קרובים היו חולים במחלה זו, משום שכך אחוזי הסיכון שלו לחלות במחלה זו גבוהים יותר.

ככל שהקשר המשפחתי יותר רחוק כך אחוזי הסיכון יורדים, ועל כן לפי הנתונים המתקבלים

הפרויקט יבנה עץ גנטי מתאים.

בגלל שלכל מחלה יש אחוזים שונים שהיא תורש לצאצאים הבאים אז לפי המחלה שגיליתי

ששניהם נשאים אני אבדוק מה רמת האחוזים שהצאצא יחלה

בעיה אלגוריתמית:

אגביל את מספר המחלות שיכול לזהות הפרויקט למספר מחלות מסוימות בהן אתמקד

ועליהן אחליט בהמשך.

צורת המימוש:

אני יוצרת dataset שמלא ברצפי דנ"א כדי שיוכל לזהות סוגי מחלות גנטיות על פי הרצף המתקבל עבורו. בניית ה dataset תעשה על ידי הגדרת רצפים שונים ואגדיר מהי נשאות ומה מספר על מחלה וכן איך אדע עבור כל מחלה- אגדיר מהי בעיה ברצף- ומה מראה שהרצף תקין. אעשה זאת על פי חישובים שונים. לדוגמא- על פי מספר הפעמים שברצף יש את הכפילות A אגדיר לו מהי נשאות- אחרי שה dataset יהיה מוכן אאמן אותו כך שידע לזהות את המחלה והאם הנבדק בריא או נשא באחת המחלות שהגדרתי. הזיהוי יעשה על פי מספר העותקים הפגומים- כלומר לבריא קיימים שני עותקים תקינים, לנשא יש עותק אחד פגום ועותק שני תקין. מכאן ואילך אממש את האלגוריתם ע"י בנית אוטומט דטרמיניסטי (סופי) המקבל את תוצאת הdataset ופועל בהתאם.

במידה והdataset מחזיר:

1. שהנבדקת בריאה- יוחזר פלט המעדכן אותה בתוצאת הבדיקה המבשרת על בריאותו.

2. שהנבדקת הינה נשאית- תוחזר לנבדקת בקשה לשלוח גם את בן הזוג לבדיקה.

3. ש2 הנבדקים נשאים לאותה מחלה והנבדקים כן הולכים להנשא- האוטומט יתחיל לפעול והוא יחשב את הסיכון לצאצא לחלות במחלה המסוימת בה 2 ההורים נשאים על ידי הכנסת המצב הבריאותו שלו ושל בני משפחתו.

האוטומט ימומש על ידי מערך מצבים שכל תא בו יצביע למבנה המכיל פרטים על המצב (מה צריך לבצע במצב זה) וכן טבלת גיבוב שעל פיה ידע האוטומט לאיזה מצב עליו להמשיך מהמצב הנתון.

לדוגמא: אם האוטומט נמצא במצב 2Q ואימו של הנבדק חולה במחלה זו ואביו לא, האוטומט ייגש לטבלת הגיבוב במקום המרמז לכך (אמא כן, אבא לא) ע"י פונקציית הגיבוב שאבנה ויתקדם למצב הרשום בו.

הערה: בתורת החישוביות אוטומט סופי דטרמיניסטי הוא מודל מתמטי ,המגדיר שפה פורמלית. המודל מורכב מאוסף סופי של מצבים וכללי מעבר ביניהם. בהינתן קלט, הבנוי מסדרה של סמלים (סימנים) מתוך א"ב (אוסף כל הסימנים האפשריים) ידוע, מתבצע מעבר סדרתי על הסמלים, ובהתאם, מתבצעים מעברים בין מצבי האוטומט – אחד עבור כל סמל. המצב ההתחלתי ידוע מראש וכל מעבר מוגדר באופן חד-ערכי ויחיד "(דטרמיניסטי(" על פי הסמל הבא שנקרא. כאשר נקראים כל הסמלים שבקלט, מתבצעת בדיקה של סוג המצב בו נמצא האוטומט)המצב האחרון שאליו הגיע(אם מדובר ב"מצב מקבל," הקלט נחשב כ"מילה בשפה" של האוטומט, אחרת, הוא נחשב כמילה שאיננה שייכת לשפת האוטומט. אך בפרויקט שלי החלטתי להשתמש במודל זה כדי לחשב את הסיכון שהצאצא של הנבדקים עלול לחלות בצורה אוטומטית ונקיה.

תיאור הטכנולוגיה:

שפת תכנות בצד שרתc++: ,פייתון

שפת תכנות בצד לקוח react:

לוחות זמנים:

הצעה ואפיון: נובמבר

למידת חקר על הגנטיקה: ספטמבר-דצמבר

בדיקת סביבות עבודה מתאימות והתקנת ספריות: ינואר

כתיבת צד שרת ולקוח: פברואר-מרץ

חיבורים בין השפות השונות: אפריל

כתיבת ספר פרויקט+ בדיקות: מאי

הגנה והכנה להגשה: מאי

הגשה: יוני

מנחות פרויקט: המורה תמר זקס

חתימת הסטודנט: רעות כהן

חתימת רכז המגמה:אלה גליק

-מבוא-

רקע תאורטי על הפרויקט

הפרויקט משמש כתוכנה לבדיקת נשאות למחלה גנטית והתאמה בין בני זוג על ידי רצף גנים וחישוב הצאצאים לחלות -במידה וקיימת התאמה בין בני הזוג. המשתמש הראשון מכניס את רצף הגנים שלו על ידי קובץ fasta .

הקובץ נשלח לשרת ומופעל האלגוריתם שסורק את הרצף שהתקבל בקובץ ובדק אותו האם קיימת נשאות לאחת או יותר מהמחלות שעליהן אני עושה את הפרוייקט, כל התוצאות נכנסות למילון המחלות שמתאר האם קיימת נשאות למחלה או לא, המתואר על ידי הערכים TRUE או FALSE לאחר מכן במידה והיה קיים במילון את הערך TRUE לאחת מהמחלות נבקש מהמשתמש השני להעלות קובץ fasta לבדיקת התאמה בין הבני זוג נבצע את אותו התהליך עבור המשתמש השני ומידה וקיימת התאמה בינהם כלומר יש לשניהם את הערך TRUE עבור אותו המפתח-המחלה, נחשב את הסיכויים של הצאצאים לחלות באותה המחלה.

מכאן ואילך מימשתי את האלגוריתם ע"י בנית אוטומט דטרמיניסטי (סופי) המקבל את תוצאות הבדיקה בין 2 המשתמשים ופועל בהתאם.

תהליך המחקר

מצב קיים

כל אדם יש לו תורשה של DNA ובתוכו יש מוטציות,מחלות או נשאות למחלה מאחר ואי אפשר לדעת מראש איזה מחלה יכולה להגרם בגלל שינוי בגן או הכפלה של נוקלאוטיד וכו, יש אפשרות למנוע מראש את המחלה על ידי ניתוח גנטי שהוא שילוב התכנות עם הביולוגיה כדי לקבל תובנות לגבי המבנה הגנטי שלו וכיצד הוא משפיע על האדם עצמו וצאצאיו.

ישנם גופים רבים העוסקים בבדיקת נשאות למחלות גנטיות חלקם עבור מניעת התפרצות המחלה אצל אדם הנשא אליה ועלול לפתח את המחלה אצלו בגיל מאוחר וחלקם עבור מניעת העברת המחלה אצל אדם מסויים או בני זוג(תלוי אם המחלה היא רציסיבית או דומיננטית) לצאצאיהם-אחד מהארגונים שיש הוא אגודת "דור ישרים" שהיא אגודה למניעת מחלות גנטיות רצסיביות. מטרת האגודה היא למנוע לידת תינוקות הלוקים במחלות גנטיות חשוכות מרפא ע"י בדיקה מוקדמת ובחינת ההתאמה הגנטית לפני האירוסין .- על הרעיון של האגודה הזו מתבסס הרעיון של הפרוייקט שלי בתוספת חישוב הסיכוי להעברת המחלה לצאצאים.

הפרוייקט שלי-מוצא נשאות למחלה\למחלות על ידי רצף גנטי של בני הזוג,בודק אם יש בינהם התאמה-כלומר אם הם נשאים לאותה המחלה או לאותן המחלות +מחשב את הסיכון של הצאצאים לחלות על ידי שאלון תורשתי.

סקירה ספרותית

אתרים בהם השתמשתי במחקר

רקע –

ביואנפורמטיקה-

https://he.wikipedia.org/wiki/%D7%91%D7%99%D7%95%D7%90%D7%99%D7%A0%D7%A4%D7%95%D7%A8%D7%9E%D7%98%D7%99%D7%A7%D7%94

https://www.doryes.com/

https://he.wikipedia.org/wiki/DNA

https://davidson.weizmann.ac.il/online/askexpert/life_sci/%D7%9E%D7%94%D7%95-dna-%D7%95%D7%9B%D7%99%D7%A6%D7%93-%D7%94%D7%95%D7%90-%D7%A0%D7%95%D7%A6%D7%A8-%D7%A2%D7%95%D7%9E%D7%A8

https://he.wikipedia.org/wiki/%D7%90%D7%A0%D7%9E%D7%99%D7%94_%D7%97%D7%A8%D7%9E%D7%A9%D7%99%D7%AA

https://he.wikipedia.org/wiki/%D7%A1%D7%99%D7%A1%D7%98%D7%99%D7%A7_%D7%A4%D7%99%D7%91%D7%A8%D7%95%D7%96%D7%99%D7%A1

https://he.wikipedia.org/wiki/%D7%A4%D7%A0%D7%99%D7%9C%D7%A7%D7%98%D7%95%D7%A0%D7%95%D7%A8%D7%99%D7%94

https://genes.co.il/%D7%94%D7%95%D7%A8%D7%A9%D7%94-%D7%90%D7%95%D7%98%D7%95%D7%96%D7%95%D7%9E%D7%9C%D7%99%D7/%AA-%D7%A8%D7%A6%D7%A1%D7%99%D7%91%D7%99%D7%AA

https://genopedia.co.il/index.php/%D7%98%D7%99%D7%99_%D7%96%D7%A7%D7%A1

אוטומט

https://www.youtube.com/watch?v=VOQGOvqmeRg

https://dorazaria.github.io/assets/summaries/%D7%90%D7%95%D7%98%D7%95%D7%9E%D7%98%D7%99%D7%9D%201%20-%20%D7%A1%D7%99%D7%9B%D7%95%D7%9D%20%D7%94%D7%A8%D7%A6%D7%90%D7%95%D7%AA.pdf

ספריית Biopython

https://biopython.org/

לימוד שפת c++

https://www.w3schools.com/cpp/

מטרת הפרויקט

מטרת הפרויקט שלי הוא מציאת נשאות למחלה גנטית, התאמה בין בני זוג על פי הרקע הרפואי שלהם וחישוב הסיכויים לילד חולה במידה וקיימת התאמה ידיעתה מראש "ידיעת הבעיה היא חצי הפתרון"

ברגע שבני זוג הבאים להנשא יודע שהוא נשא למחלה מסוימת זה יעורר בו לנהוג במשנה זהירות לפני הקמת משפחה

מטרתי היא גם לרכוש ידע בסוגי טכנולוגיות שונות בעולם התכנות ולפתח את כישורי כתיבת הקוד שלי בשפות תכנות שונות.

היעדים לפרויקט:

1. בניית מערכת שתזהה נשאות למחלה על ידי רצף DNA.

2. קבלת רצף גנטי בקובץ fasta והחזרת רשימה של המחלות ללקוח לאחר סקירה מעמיקה של הרצף הנבדק עם מידע עבור כל אחת מהמחלות האם הוא נשא או לא

3.במידה וקיימת נשאות עבור לפחות אחת מהמחלות הלקוח יתבקש להעלות את רצף הDNA של בן\בת הזוג השני\ה וגם עליו תתבצע הבדיקה

4.במידה וגם המשתמש השני נשא לפחות לאחת מהמחלות- תתבצע בדיקה האם קיימת ביניהם התאמה-כלומר האם הם נשאים לאותה המחלה\ות

5.במידה וכן יופעל האוטומט והוא יבצע חישוב לסיכויי הילדים שלהם העתידיים שלהם לחלות, על פי מידע רפואי של עץ המשפחה של כל אחד מהצדדים.

אתגרים

- למידת כל החקר זה נושא רחב שגם חוקרים ששנים בתחום לא הגיעו לכל הידע

הרב שכולל עולם הגנטיקה.

- מחלות גנטיות שאפשר למצא נשאות להן על ידי רצף גנטי בלבד.
- מציאת המחלות גנטיות הנמצאות ברצף DNA על ידי קוד יעיל ומדוייק.
- למידת הספריות והשפה של python ברמה גבוהה.
- למידת השפה c++ מהבסיס.
- למידת האוטומט
- בניית האוטומט בצורה האופטימלית והמדוייקת ביותר.
- לדאוג לכל מקרי הקצה האפשריים.
- חיבור בין 2 השרתים

אתגרים מרכזיים:

- **חקר על הגנום האנושי ומחלות גנטיות**-כשהתחלתי לעבוד על הפרויקט שלי ידעתי שיש לפניי כמות משמעותית של חקר ולמידת ביואינפורמטיקה הפרויקט עסק בתחום שלא הכרתי לעומק איך כל הגנטיקה ואיך הדנ"א משפיע על גופנו והשפעת המחלות עליו ולכן הייתי צריכה להקדיש לזה זמן רב. החקר לקח לי הרבה יותר זמן משאר בנות הקבוצה משום שלמידת הנושא הזה היא קשה. ביואינפורמטיקה הוא מקצוע שלומדים כמה שנים ולא למידה של כמה חודשים-הייתי צריכה להבין מה ההבדל בין מחלה רציסיבית למחלה דומיננטית וכיצד הן משפיעות על דור העתיד ומידה ו2 ההורים נשאים לאותה המחלה.
- **חקר על מחלות גנטיות שאפשר למצוא נשאות להן על ידי רצף גנטי בלבד** -לאחר שלמדתי הייתי צריכה להתמקד בחקר על מחלות רציסיביות שנשאות אליהן מאופיינת על ידי מוטציות שכיחות הניתנות לזיהוי כאשר נבצע סקירה על רצף הDNA
- **בניית האוטומט בצורה האופטימלית והמדוייקת ביותר**-בניית האוטומט הייתה מורכבת מאוד משום גודלו והעובדה שכל שלב בבניה שלו תלויה בשלב הקודם. בתחילה הגודל של האוטומט שלי הוא של כ- 1092 מצבים סופיים- גודל שמצריך עבודה עקבית עם שימת לב ותפיסת כל הנתונים במקביל. לבסוף מצאתי דרכים לצמצם את האוטומט לצורה יעילה ומדוייקת יותר.
- **מציאת המחלות גנטיות הנמצאות ברצף DNA על ידי קוד יעיל ומדוייק**-בנוסף חשבתי בתחילה לממש את הקוד למציאת הנשאות למחלה על ידי למידת מכונה ולכן עסקתי זמן רב במציאת datasets שמתארים לי נשאות של המחלה בתוך רצף DNA לאותן המחלות שבחרתי .ומכיוון שהייתי צריכה dataset ספיציפי מאוד לא מצאתי dataset מתאים וזמין שיתאים לי לפרוייקט לכן בחרתי שלא להשתמש בלמידת מכונה בשביל הפרוייקט שלי וחיפשתי אחר אלטרנטיבה אחרת.

אחד הקשיים שנתקלתי בהם היה חיפוש אחר מחלות שיתאימו לי לפרוייקט החלטתי לקחת 4 מחלות גנטיות שעליהן יתבסס הפרוייקט שלי ולאחר שמצאתי 4 מחלות גנטיות רציסיביות וכתבתי את הקוד למציאת נשאות למחלה עבור כל אחת מהן גיליתי שאחת המחלות (הX השביר) לא תתאים לי לפרוייקט -אומנם היא מחלה רציסיבית אך היא שונה מעט משאר המחלות גנטיות הרצסיביות בהפעתה על דור העתיד (היא פוגעת בבנים יותר מאשר הבנות ובמחלה הזו לא קיים בן שהוא נשא של המחלה אלא הוא יכול להיות רק חולה או בריא) ולכן הייתי צריכה להפסיק את המשך הפרוייקט ולחזור לנקודת ההתחלה בחיפוש אחר מחלה גנטית רציסיבית נוספת,הכנת קבצי fasta נוספים שמתארים לי נשאות למחלה וכתיבת קוד מתאים למציאת הנשאות למחלה.

בהתחלה חשבתי שאני אצטרך להתעסק בלמידת מכונה אז התחלתי לחקור מעט אודותיה ללמוד מה זה אומר למידת מכונה, איך מלמדים את מחשב את מה שאנחנו רוצים ולפני שנכנסתי לחלק התאורטי בצורה עמוקה רציתי קודם לראות אם יש לי חומר לעבוד איתו אז התחלתי לחפש dataset של DNA שמתארים לי נשאות למחלה, אבל לאחר שלא מצאתי dataset מתאים החלטתי לחפש אלטרנטיבה אחרת שעל כך אפרט בהמשך, למרות כל האתגרים למדתי מיומנויות בלמידה עצמית, תיקון באגים, ועמידה ביעדים.

מדדי הצלחה:

- קוד ברור ויעיל
- שימוש נכון במבני נתונים
- זיהוי נכון של נשאות המחלה
- כתיבת אוטומט שיעל בצורה המדוייקת ביותר ובצורה אוטומטית ככל האפשר

בחרתי בנושא הזה כי:

תמיד התעניינתי בעולם הרפואה (לפני שהלכתי על תכנות התלבטתי באם ללכת לכיוון של אחיות) ולכן כשנדרשתי לבחור נושא לפרויקט, חיפשתי משהו שישלב בין תכנות לרפואה וראיתי שעולם הגנטיקה הוא ענק ומעניין. שילבתי בפרויקט סיפריות ביולוגיות בפייתון ואוטומט חישובי שאלו כלים טכנולוגיים מתקדמים.

ניתוח חלופות מערכתי

כשנגשתי לראשונה לבצע את הפרויקט לא ידעתי דבר וחצי דבר שיוכל לעזור בכדי להתמודד עם הבעיה האלגוריתמית.

אחרי העבודה הקשה שהייתה לי, ללמוד ולחקור את עולם הגנטיקה הרחב, הייתי צריכה לבחור במה להתמקד ואיך לממש את מציאת הנשאות ברצף. הייתי מוכרחה להגביל את הפרויקט שלי לממדים כאלו שאוכל להתמודד איתם תוך הזמן הקצוב שניתן לנו.

בהתחלה חשבתי לממש את מציאת הנשאות ברצף על ידי בלמידת מכונה (עדיין לא ידעתי באיזה סוג) ולכן חיפשתי dataset של רצפי גנים. גם לאחר חיפושים רבים לא מצאתי וידעתי שלבנות datasets זוהי עבודת נמלים שתיקח לי הרבה מאוד זמן ולכן המשכתי לחפש תוך כדי לימוד של חומרים נוספים שנצרכתי להם.

התחלתי ללמוד את כל החומר התאורטי שאני צריכה בשביל הפרוייקט וזה היה ארוך במיוחד (בעיקר בגלל כל השמות המסובכים והתתי נושאים שקשורים אחד לשני שעוזרים להבנה המעמיקה יותר של החומר שאותו הייתי צריכה) אחרי שגמרתי ללמוד את החומר התאורטי של ביואינפורמטיקה, אז התחלתי במשך ימים ולילות ללמוד אילו מחלות יתאימו לי לפרוייקט אחרי מחלות רבות שחקרתי אודותיהן החלטתי על 4 מחלות שעליהן אני אבצע את מציאת המחלה לאחר מכן התחלתי לחפש כיוון כיצד אני מממשת את התהליך של מציאת הנשאות בתוך רצף הDNA , בהתחלה חשבתי שהדרך הטובה ביותר היא על ידי אימון מודל שיידע לזהות מהו רצף תקין ומה הוא רצף בעל נשאות למחלה,

אחרי שחשבתי שאני אתחיל לממש את תהליך מציאת המחלה\המחלות על ידי למידת מכונה, אז התחלתי לחפש datasets של אותן המחלות שבחרתי ואחרי חיפוש מעמיק במלא מאגרים של dataset ואתרים על גנטיקה ורצפי DNA כגון-GENEBANK,EBI, NCBI, UCSC

-לא הצלחתי למצוא dataset עבור המחלות שעליהן אני עושה את הפרוייקט שלי –מכיון שה datasets שחיפשתי הם מאוד ספציפים ובדרך כלל רצפי DNA של אנשים אלו דברים סודיים שלא מפרסמים אותם כך שהרבה אתרים לא מביאים מידע עבור רצפי DNA של אנשים כי

זהו מידע אישי ונוגד את חוקי האתיקה וידעתי שלבנות datasets לבד זוהי עבודת נמלים שתיקח לי הרבה מאוד זמן ולכן המשכתי לחפש תוך כדי לימוד של חומרים נוספים שנצרכתי להם.

לבסוף לאחר חיפושים ממושכים הבנתי שבניית מכונה לא תתאים לי מחוסר משאבים שימושי ללמידת המכונה

אז המשכתי לחפש אילו עוד דרכים אני יכולה להשתמש בפרוייקט כדי למצוא נשאות ברצף,

לאחר חיפושים ממושכים מצאתי את הסיפרייה Biopython

תיאור החלופה הנבחרת

כשנגשתי לראשונה לבצע את הפרויקט לא ידעתי דבר וחצי דבר שיוכל לעזור בכדי להתמודד עם הבעיה האלגוריתמית.

אחרי העבודה הקשה שהייתה לי, ללמוד ולחקור את עולם הגנטיקה הרחב, הייתי צריכה לבחור במה להתמקד ואיך לממש את החיזוי הגנטי. הייתי מוכרחה להגביל את הפרויקט שלי לממדים כאלו שאוכל להתמודד איתם תוך הזמן הקצוב שניתן לנו.

אחרי חיפוש מעמיק מצאתי סיפרייה בפייתון שנקראת Biopython .

Biopython- היא אוסף של כלים ומודולים המיועדים לניתוח נתונים ביולוגיים, במיוחד בתחום הביואינפורמטיקה .היא מספקת מודולים שונים המאפשרים לבצע מגוון רחב של משימות, כגון:

- **ניתוח רצפים ביולוגיים :**קריאה, כתיבה, ועיבוד של רצפי DNA, RNA וחלבונים בפורמטים שונים (FASTA, GenBank, FASTQ) ועוד.
- **ביצוע יישור רצפים :**שימוש באלגוריתמים ליישור רצפים גלובליים ולוקאליים.
- **גישה למאגרי מידע ביולוגיים :**אינטראקציה עם מאגרי מידע מקוונים כמו NCBI Entrez ו ExPASy- ועוד...

אני בחרתי להשתמש בספרייה זו כי היא תוכל לעזור לי בלקרא קבצי fasta- שאלו קבצים המכילים רצף DNA של אדם מסויים,

בנוסף חשבתי להשתמש הסיפריית re בפייתון- בעזרת ספרייה זו אני אוכל לזהות ביעילות מוטציות ברצפי DNA על ידי חיפוש תבניות מסוימות של נוקלאוטידים (A, T, C, G) שמתארות שינוי גנטי צפוי או מוכר בהתאם למחלה שעליה אני עובדת

אחרי שהבנתי שהספריות הללו יוכלו לעזור לי במציאת נשאות למחלה חקרתי על אילו מחלות אני אעשה את הפרוייקט שלי והינה התמצות עליהן:

1. פנילקטונוריה
2. סיסטיק פיברוזיס
3. אנמיה חרמשית- (Sickle Cell Anemia)
4. מחלת טיי-זקס (Tay-Sachs Disease)

1.פנילקטונוריה- פנילקטונוריה היא מחלה גנטית רציסיבית, מחלה זו היא מחלה גנטית תורשתית נדירה שבה הגוף אינו יכול לפרק כראוי חומצת אמינו בשם פנילאלנין. חומצות אמינו הן אבני הבניין של חלבונים. פנילאלנין מצויה בעיקר במזונות עשירים בחלבון. ניתן לזהות נשאות למחלה גנטית על ידי חיפוש אחר מוטציות ספציפיות בגנים שאחראים למחלה,לדוגמא כאשר כמקומות מסויימים בגן מוחלפת האות G בA זה עלול לבטא מוטציה שקשורה למחלה

2 . **סיסטיק פיברוזיס-**תסמונת הכישור החסר היא מחלה גנטית תורשתית הגורמת להצטברות של ריר סמיך וחסר באיברים שונים בגוף, בעיקר בריאות ובמעיים. המחלה נגרמת ממוטציה בגן CFTR, האחראי על ייצור חלבון המסדיר את מעבר המים והמלחים דרך קרומי התאים.
אחת מוטציות הנפוצות של מחלה זו היא החסרה של 3 נוקלאוטידים הגורמים לאיבוד חומצת האמינו פנילאנין

3. **אנמיה חרמשית-** אנמיה חרמשית היא מחלה כרונית ולכל החיים הנגרמת על ידי מוטציה בגן האחראי לייצור חלק מחלבון ההמוגלובין במצב תקין, תאי הדם האדומים הם עגולים וגמישים, מה שמאפשר להם לנוע בקלות בכלי הדם. באנמיה חרמשית, חלק מתאי הדם האדומים מקבלים צורה חרמשית (כמו סהר או האות C) והופכים קשים ודביקים ,מוטציה שכיחה בגן-כאשר יש החלפה של הנוקלאוטיד G בA לדוגמא: הרצף CGGGCT ישתנה כאשר G->A לרצף - CGAGCT

4. **מחלת טיי-זקס-** מחלת טיי-זקס היא הפרעה גנטית תורשתית נדירה וקטלנית, הפוגעת בעיקר בתינוקות וילדים צעירים. המחלה נגרמת כתוצאה מחוסר באנזים חיוני האחראי על פירוק חומר שומני במוח ובתאי העצב.כאשר האנזים חסר או לא מתפקד כראוי השומנים מצטברים בתאי העצב ופוגעים בהם, הצטברות זו גורמת לנזק מוחי מתקדם המוביל להדרדרות נוירולוגית קשה. אחת המוטציות הנפוצות למחלה זו היא כאשר יש החלפה של התימין-T בציטוזין C מופיע בגן הרצף GCCC במקום GCCT

אלה 3 מחלות שעליהם אני בונה את האוטומט

מהו אלגוריתם: DFA

-DFA אוטומט סופי דטרמיניסטי הוא מודל לזיהוי רצפי קלט .

הוא מורכב מקבוצה גדולה של מצבים, שעליהם תהיה פונקציית מעבר שהיא עוברת כך:

מצב התחלתי וקבוצת מצבי קבלה- המציינת אילו מצבים מתקבלים או מסיימים בהם.

כל מצב הגדרתי על ידי הקלט שלו מהמצב הקודם והסימן שהוא מקבל. פונקציית המעבר תגדיר בבירור את הדרך שבה האוטומט מתקדם בין המצבים, בהתאם למעבר שאותו צריכה לעבור. בסיום קריאת הסימנים, האוטומט יהיה במצב סופי, שאז הוא ישכלל את האחוזי הסיכוי שהנבדק יחלה.

האוטומט מדמה תהליך בדיקה גנטית לזוגות נשאים, ומחשב את סיכויי ההעברה של מחלות גנטיות לילדים, על בסיס קלטים גנטיים. הוא בנוי בצורת עץ טרינארי ,כלומר: לכל צומת יכולים להיות עד שלושה מעברים, בהתאמה לשלושה קלטים אפשריים שונים. כאשר מגיעים למצב מסוים, נוסיף את משתנה האחוזים שבצומת הנוכחית. עד שנגיע למצב מקבל ולבסוף נקבל את תוצאות הסיכויים בהתאם לקלטים שהוכנסו לאוטומט.

אפיון המערכת

חומרה: מעבד i7 16GB RAM,

עמדת פיתוח: מחשב hp

מערכת הפעלה:Windows 10 pro

תוכנה שפות:python ,C++

כלי תוכנה לפיתוח המערכת:PyCharm, visual studio

מסד נתונים: אין

מודול המערכת:

- הפרויקט עוסק רק במחלות שמתפרצות כאשר יש 2 עותקים פגומים- מחלות רציסיביות, ואלה המחלות: 1. פנילקטונוריה 2. סיסטיק פיבוריזיס 3. אנמיה החרמשית 4.טאי זקס.
- המערכת נותנת חיזוי באחוזים של סיכויי התחלואה במחלה שהתקבלה מהבדיקה.

הגבלות הפרויקט:

1.הפרויקט לא מתעסק במחלות דומיננטיות.

2. הפרויקט לא מקבל רצף DNA אלה מקטע של הרצף-RNA.

3. הפרויקט לא מתעסק במחלות על פי מוצא.

אפיון פונקציונלי

פונקציות עיקריות: צד שרת:
get_genetic_sequence()

- פונקציה המקבלת קובץ fasta ומחלצת מתוכו את רצף הDNA וממירה אותו לstring על מנת למצוא בו מוטציות אפשריות.

- findCarrier(sequence)

הפונקציה מקבלת את הרצף כמחרוזת ומפעילה עליו חיפוש מוטציות אפשריות עבור כל אחת מהמחלות,לבסוף הפונציה מחזירה עבור כל מפתח ממילון המתאר נשאות למחלות ערך true\false בהתאם.

- compare_results(results_girl, results_boy)

במידה והייתה קיימת נשאות עבור 2 הנבדקים הפונקציה הזו תופעל ותשווה בין הערכים במילון על פי המפתח, במידה והייתה קיימת התאמה בינהם-כלומר,היה את הערך true לשניהם באותו המפתח-הפונקציה תחזיר שאכן קיימת התאמה בין 2 הנדבקים.

- calculateSiblingsChance()

פונקציה המקבלת מידע עבור האחים ומחשבת בהתאם את הסיכויים שלבסוף התווספו לחישוב הסופי של כל עץ המשפחה מהאוטומט.

ביצועים עיקריים

שלב ראשון: קבלת קובץ fasta מהמשתמש והמרת הרצף למחרוזת

שלב שני: מציאת נשאות לאחת או יותר מהמחלות והכנסת הסטטוס-נשא או לא- למילון המחלות

שלב שלישי: במידה וקיימת נשאות-חזרה על שלב ראשון ושני עבור המשתמש השני וקביעה האם קיימת התאמה ביניהם.

שלב רביעי: אוטומט שיביא תוצאה סופית של אחוזי סיכון לחלות.

אילוצים

1.קבלת קובץ fasta בלבד על ידי פונקציות בסיפריית biopython והמרתו למחרוזת על מנת לחפש מוטציות אפשריות על ידי סיפריית re.

2. יכול להיות מצב בו הרצף יראה שתי מחלות שיש ללקוח אבל המערכת תביא את המחלה היותר חזקה אצלו.

3. בניית האוטומט בגודל של 15 מצבים סופיים שיכיל את כל הפרמטרים הנדרשים למציאת האחוז המדויק לסיכון לחלות עד כמה שניתן

4.המערכת נבנתה בזמן מוגבל

5.המערכת לא אחראית על הכנסת נתונים שגויים בעץ המשפחה.

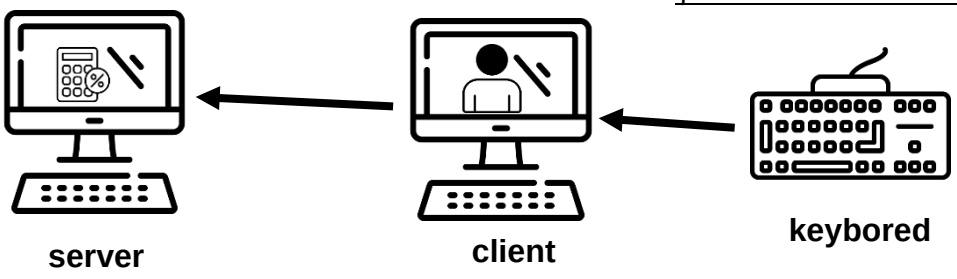
תיאור הארכיטקטורה

הארכיטקטורה של הפתרון המוצע

1 **שרת 1:** שרת python יהיה אחראי להביא תוצאות הנשאות עבור כל אחת מהמחלות על ידי ניתוח הקצף שהתקבל ומציאת מוטציות אפשריות ברצף.

2. שרת: 2 שרת C++ -יטפל בנתוני שאלון העץ המשפחתי שהלקוח נתן, על ידי אוטומט גדול והוא ייתן חלק בחישוב האחוזים.

תיאור הרכיבים בפתרון



תאור פרוטוקלי תקשורת

אין

תיאור הCase Use העיקריים במערכת

רשימת הCase Use

- העלאת קובץ fasta לשרת
- המרת תוכן הקובץ למחרוזת וסריקתו למציאת נשאות למחלה
- בדיקת התאמה
- חישוב סיכויי הצאצאים לחלות במחלה

UC1

- UC1:Identifier
- Name : העלאת קובץ.
- Description: המשתמש יכניס קובץ fasta המכיל את רצף הDNA שלו על מנת למצוא נשאות למחלה.
- Actors : משתמש
- Post-Condition: מקלדת

UC2

- UC2:Identifier
- Name : בריא.
- Description: לאחר סריקת הרצף תוחזר הודעה מתאימה ללקוח.
- Actors : משתמש

UC3

- UC3:Identifier
- Name : נשא.
- Description: תוחזר הודעה ללקוח על אילו מחלות הוא נשא אליהן ויבקש מהנבדק השני לעלות את רצף הגנים שלו.
- Actors : משתמש
- Post-Condition: מקלדת

UC4

- UC4:Identifier
- Name : לא קיימת התאמה בינהם.
- Description: תוחזר הודעה מתאימה.
- Actors : משתמש

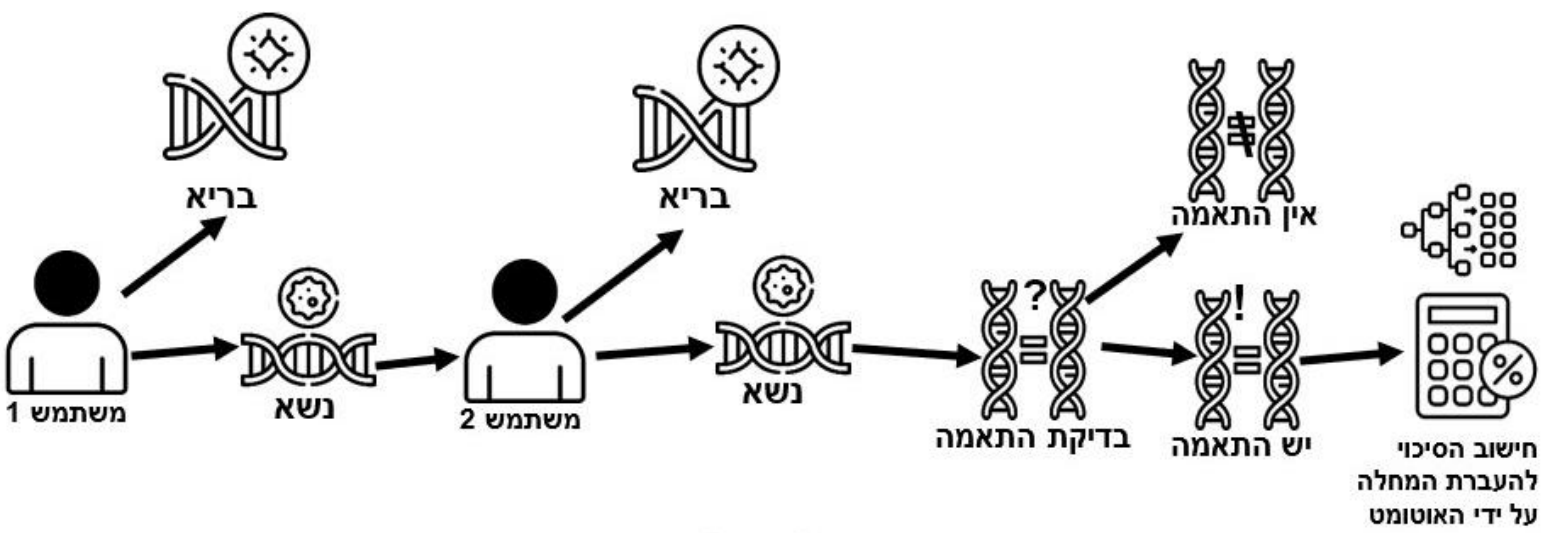
UC5

- UC5:Identifier
- Name : קיימת התאמה בינהם.
- Description: המשתמשים ימלאו את הפרטים על עץ משפחתם.
- Actors : משתמש
- Post-Condition: מקלדת

UC6

- UC6:Identifier
- Name : הפעלת האוטומט והחזרת התוצאה.
- Description: תוחזר התוצאה של האחוזים.

Use Case Diagram



מבנה נתונים בהם השתמשתי

מציאת נשאות:

- **מערכים** : מערך המכיל בתוכו רצפים של נוקלאוטידים המתארים מוטציות אפשריות למחלה גנטית ספציפית.
- **מילון:** מילון המכיל פרטים על מצב הבריאותי של המשתמש -המפתח הוא שם המחלה והערך הוא true\false המתאר אם המשתמש נשא או לא.

חישוב האחוזים

- **מפה:** המבנה העיקרי של האוטומט הסופי הדטרמיניסטי (DFA) מיוצג באמצעות מפה וזאת כדי:

1)לייצג טבלת מעברים: המפה החיצונית ממפה כל State המצב הנוכחי למפה פנימית. המפה הפנימית ממפה כל קלט ל-מידע על המעבר. זה מאפשר גישה יעילה למידע על המעבר בהינתן מצב נוכחי וקלט.

2)חיפוש מהיר: מפות מספקות גישה אוטומטית ומהירה(ללא צורך בif) למצב הבא- על פי הקלט שהתקבל- מה שהופך את קביעת המצב הבא והמידע הנלווה על סמך המצב הנוכחי והקלט ליעילה ברורה ומהירה.

3)ארגון ברור: המבנה המקונן של המפות משקף באופן ישיר את טבלת המעברים של האוטומט, מה שהופך את הקוד לקריא וקל להבנה.

- **וקטור**: חלק ממבנה האוטומט, השתמשתי בוקטור כדי לאחסן את רשימת הקלטים התקינים עבור מעבר מסוים ממצב מסוים. בחרתי להשתמש בוקטור כי וקטור הוא אוסף (מערך) דינמי שיכול לגדול או לקטון לפי הצורך. במקרה זה, מספר הקלטים התקינים עבור כל מעבר יכול להיות שונה. וכן על מנת לאפשר בדיקת קלט פשוטה, כך ניתן לעבור בקלות על כל הקלטים התקינים בוקטור כדי לבדוק אם הקלט שהמשתמש הזין הוא חוקי עבור המצב הנוכחי.

- **מחסנית**:השתמשתי בשני מחסניות:

1)כדי לשמור את היסטוריית המצבים שבהם האוטומט עבר.

2) כדי לשמור את היסטוריית האחוזים שהצטברו בכל מעבר.

השתמשתי במחסנית כדי לאפשר למשתמש לחזור אחורה במידה והוא הקיש מידע שגוי על אחת מבני משפחתו ומכיון שהמחסנית עובדת בשיטת LIFO -האחרון שנכנס הוא הראשון שיוצא זה יקל על המשתמש כאשר הוא מזין 'U' ישלוף את המצב הקודם ואת האחוז הקודם מהראש המחסנית כדי לחזור למצב הקודם ולשחזר את החישוב.

חישוב יעילות האלגוריתם

אלגוריתם בדיקת הנשאות- פועל בסיבוכיות של o(n)-מאחר והוא עובר על רצף DNA, כל שאר הבדיקות באלגוריתם מתבצעות בזמן קבועה o(1)

בס"ד

רעות כהן

אוטומט-כל הפעולות עובדות בזמן קבוע ,חוץ מהwhile-שזה תלוי במשתמש -אם הוא מקיש הכל כמו צריך הסיבוכיות צריכה להיות קבוע ואם לא אז הלולאה יכולה לרוץ קצת יותר זמן אבל עדיין בזמן קבוע.

תיאור התוכנה-

סביבות עבודה: Visual studio code, pycharm

שפת תיכנות:שרת 1-בפייתון

שרת2 :בC++

תיאור אלגוריתמים מרכזיים

אלגוריתם למציאת נשאות למחלה

לצורך מציאת נשאות למחלה-יש להעלות קובץ FASTA-קובץ FASTA הוא פורמט טקסט פשוט המשמש לייצוג רצפי נוקלאוטידים(DNA, RNA) והוא משמש לביצוע השוואות בין רצפים, חיפוש תבניות ועל ידי כך יהיה ניתן למצוא נשאות למחלה.

את תוכן הקובץ (במידה והוא לא ריק,כמובן)נמיר למחזורת על מנת לבצע פעולות עם ספריות רגולריות בפייתון.

הפונקציה get_genetic_sequence()

```
def get_genetic_sequence():  2 usages
    """
    מבקשת מהמשתמש נתיב לקובץ FASTA ומחזירה את רצף ה-DNA אם נמצא.
    """
    fasta_file = input("Enter the path to the FASTA file: ").strip()
    fasta_file = os.path.abspath(fasta_file)

    if not os.path.exists(fasta_file):
        print(f"Error: The file does not exist: {fasta_file}")
        return None

    try:
        return read_fasta(fasta_file)
    except Exception as e:
        print(f"Error reading the FASTA file: {e}")
        return None
```

הפונקציה קוראת את נתיב הקובץ ובמידה והוא לא ריק הוא מזמנת את הפונקציה read_fasta()

```
def read_fasta(file_path):  1 usage
    """
    קוראת קובץ FASTA ובודקת שהוא מכיל בדיוק רשומה אחת.
    אם יש יותר מרשומה אחת - מחזירה שגיאה.
    """
    with open(file_path, "r") as file:
        records = list(SeqIO.parse(file, format: "fasta"))  # המרת הגנרטור לרשימה כדי לספור את הרשומות

        if len(records) == 0:
            print("Error: The FASTA file is empty. Please provide a valid file with one sequence.")
            return None
        elif len(records) > 1:
            print("Error: The FASTA file contains multiple sequences. Please upload a file with only one s
            return None

        return str(records[0].seq)  # מחזיר את רצף ה-DNA של הרשומה היחידה
```

פונקציה זו מקבלת את נתיב הקובץ וקוראת ממנו את התוכן-בתחילה נבדוק אם התוכן של הקובץ תקין-קובץ תקין הוא קובץ שהתוכן שלו מכיל רק רשומה אחת כלומר-רק רצף אחד של RNA\DNA במידה וקיימת יותר מרשומה אחת -תוחזר הודעה מתאימה ללקוח,במידה והקובץ תקין הפונקציה תחזיר את הרשומה היחידה.למשתנה sequence_girl בmain

```
def main():  1 usage
    sequence_girl = get_genetic_sequence()
    if sequence_girl is None:
        print("Failed to read genetic sequence. Exiting.")
        return

    results_girl = findCarrier(sequence_girl)
    result(results_girl)

    if True in results_girl.values():
        sequence_boy = get_genetic_sequence()
        if sequence_boy is None:
            print("Failed to read boy's genetic sequence. Exiting.")
            return

        results_boy = findCarrier(sequence_boy)
        result(results_boy)
        compare_results(results_girl, results_boy)  # ומש במשתנים גלובליים
```

במידה והמשתנה לא ריק נשלח אותו לפוקציה findCarrier(sequence)

- הבודקת את הרצף האם קיימת בו נשאות לאחת או יותר מהמחלות הנבדקות.

```
def findCarrier(sequence):  2 usages
    """
    בודקת נשאות למחלות שונות על פי רצף DNA ומחזירה מילון עם התוצאות.
    """
    return {
        "Sickle Cell Anemia": Sickle_Cell_Anemia.check_sickle_cell(sequence),
        "Tay Sachs": Tay_Sachs.check_tay_sachs(sequence),
        "Pku": Pku.check_pku(sequence),
        "Cystic Fibrosis": Cystic_Fibrosis.find_cftr_mutations(sequence)
    }
```

הפונקציה מחזירה מילון המכיל בתוכו את המידע הסריאותי עבור המשתמש הפונקציה שולחת את הרצף לכל אחד מהדפים המכילים פונקציות שונות למציאת נשאות למחלה גנטית-כל דף בודק מחלה אחרת ,לדוגמא אחד מהדפים הוא-

```
def check_pku(sequence):  1 usage
    # רשימה של מוטציות נפוצות שמתארות נשאות למחלה
    characteristic_fragments = [
        "GACTGCTGAGG",
        "CTGGAGATGAC",
        "AAGCTGGTGTC",
        "TGGAAGCTCTG"
    ]

    for fragment in characteristic_fragments:
        if fragment in sequence:
            return True

    return False
```

בדף הזה מוצאים נשאות למחלה הפנילקטונוריה על ידי מציאת רצף נקלאוטידים המתארים מוטציה שמתארת נשאות למחלה , הפונקציה מחזירה trueבמידה ונמצא אחד מהרצפים והערך ביוחזר יכנס לתוך הערך במילון במקום שהמפתח מתאר את שם המחלה.

עוד דוגמא לדף המוצא מחלה גנטית ונעזר בספריית re בפייתון-

```
import re

# רצף ההתייחסות של הגן HBB (קטע קריטי)
REFERENCE_HBB_REGION = "CTGTGGGGCAAGGTGAACGTGGATGAAGTTGGTGGTGAGGCCCTGGGCAGG"

def check_sickle_cell(sequence):  1 usage
    """
    מזהה אם המוטציה באנמיה חרמשית קיימת בגן HBB.
    """
    mutated_codon = "GTG"  # ואלין (מוטציה חרמשית)

    # מוודא שהרצף מכיל את האזור הקריטי של HBB
    if REFERENCE_HBB_REGION not in sequence:
        return False  # לא נמצא HBB, לא יכול להיות אנמיה חרמשית

    # חיפוש מוטציה (יכול להיות מסביב רצפים נוספים)
    match = re.search(r"(.{0,50})" + mutated_codon + r"(.{0,50})", sequence)
    if match:
        return True  # נמצאה מוטציה
    return False  # לא נמצאה מוטציה
```

הפונקציה check_sickle_cell(sequence) נועדה לבדוק האם ברצף מופיעה מוטציה הקשורה למחלת אנמיה חרמשית ,(Sickle Cell Anemia) והיא עושה זאת ע"י חיפוש של קודון מוטנטי (GTG) באזור מסוים של גן ההמוגלובין.HBB , בתחילה יצרתי משתנה המכיל רצף ייחוס תקין מתוך גן ההמוגלובין .HBB -הרצף הזה משמש כנקודת ייחוס כדי לוודא שאנחנו אכן בודקים את הגן הנכון-מכיון שהמוטציה למחלה מופיעה באזור הזה של הגן, בתחילה נרצה לבדוק שאנחנו נמצאים במקום הנכון אם הרצף הנתון לא מכיל את האזור התקין של גן ,HBB הפונקציה מחזירה .False זה אומר שאנחנו לא במקום הנכון של הגן ולכן אין טעם לבדוק בו מוטציות.

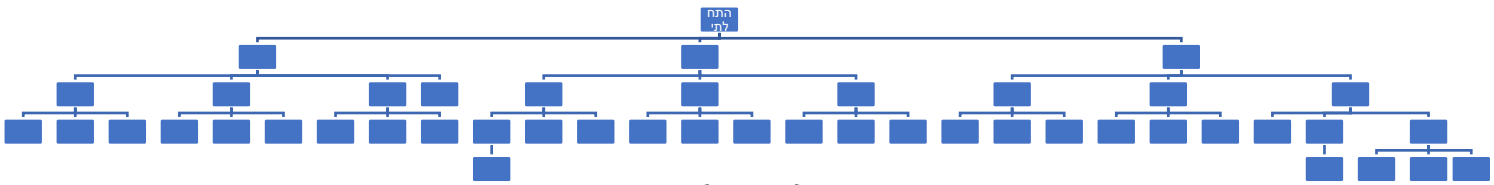
בנוסף הגדרתי משתנה נוסף המכיל את הקודון המוטנטי- GTG (ולא הקודון התקין GAG) שאותו נחפש בגן- על מנת לדעת אם האדם נשא או בריא. כדי לסרוק את הרצף ולחפש את הקודון השתמשתי בביטוי רגולרי כדי למצוא את הקודון GTG(המוטציה) שמופיע עד 50 תווים לפני ואחרי.האזור המתאר את המוגלובין, כלומר מחפשים את המוטציה באיזור שקרוב לאזור התקין של.HBB אם נמצאה התאמה — כלומר המוטציה GTG קיימת בקרבת מקום — מחזירה True, אחרת False

האוטומט

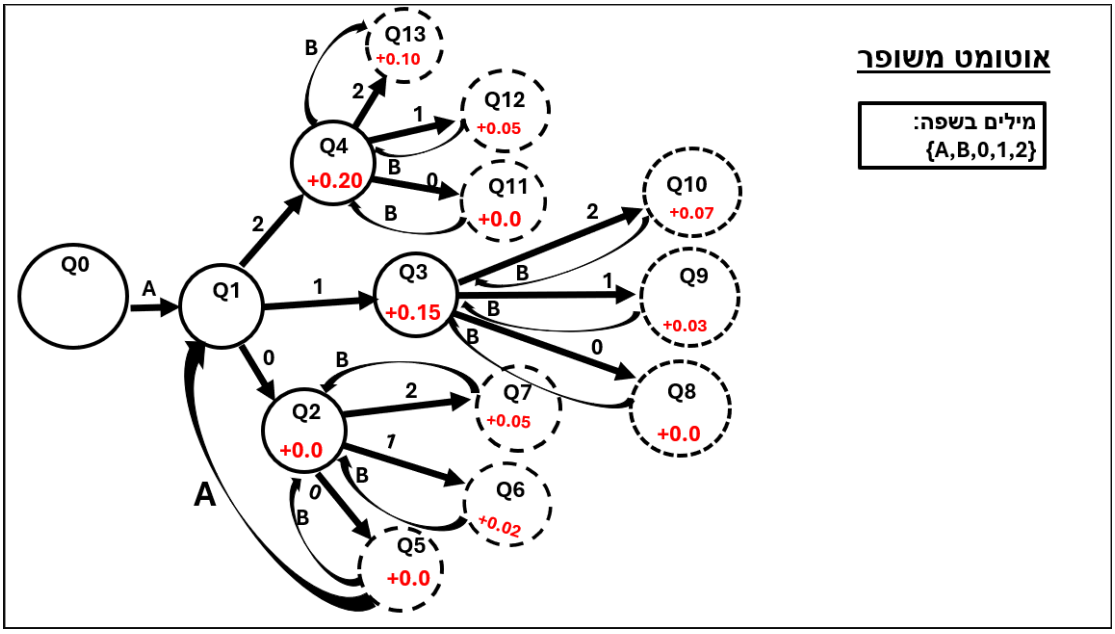
האוטומט נועד לחשב את אחוז הסיכויים שיש לילד העתידי לרשת 2 עותקים מהמחלה - כלומר לחלות במחלה, וזאת על סמך בדיקת הנשאות של ההורים שפירטתי לעיל וכעת על ידי מידע נוסף על המצב הבריאותי של המשפחה,כרגע המצב התחלתי הוא 25% כי אם 2 ההורים נשאים זה אומר שלכל אחד מהם יש עותק פגום והסיכויים שהילד העתידי יירש 2 עותקים פגומים (כלומר אחד מכל הורה) הוא 25%. המידע על המצב הבריאותי של המשפחה משקף את העברת או אי העברת המחלה לצאצאים ולכן המידע הזה מדייק את חישוב הסיכויים להעבת המחלה.

האטומט שיצרתי בהתחלה אינו דומה בכלל לאוטומט הסופי-הרעיון הבסיסי היה שמאחר והתוצאות של בני המשפחה תלויות זה בזה אז נעבור אחד אחד מבני המשפחה-אבא ואמא של הנשא ואז הסבא והסבתא מצד אבא ולאחר מכן הסבא והסבתא מצד אמא ונגיע למצב המקבל-אבל מאז האוטמט עבר תהליך ארוך עד שהוא הגיע להיות האוטומט המהיר המדוייק והיעיל שהוא עכשיו-

בהתחלה שתכננתי את האוטומט חשבתי שאצטרך בשבילו מצבים רבים -תכנון האוטומט היה בצורה נאיבית-כלומר, מתחילים את הבדיקה מהאבא ובהתאם לקלט נעבור על האמא וכן הלאה,אז חשבתי לממש זאת על ידי עץ טרינארי -מאחר ועבור כל צומת (המתארת את אחד מבני המשפחה) יש 3 קשתות-עבור בריא\נשא\חולה ומאחר והם תלויים אז האחוזים שמתווספים עבור כל שלב הם שונים בהתאם לתוצאות הקודמות-לכן בהתחלה כל רמה בעץ הייתה אחד משני המשפחה -באבא היה ברמה 0, האמא ברמה 1 וכן הלאה...אז החישוב של סך הצמתים הוא-$7^0 + 7^1 + 7^2 + 7^3 + 7^4 + 7^5$ -כך שיוצא שבאוטומט יהיו 1092 מצבים(!!) התחלתי לסרטט את האוטומט אבל זו הייתה עבודת נמלים ובקושי הצלחתי להכניס את זה לתוך הדף-זו ההתחלה של הסרטוט- 4 הרמות הראשונות בעץ:



לאחר מחשבה ארוכה ובדיקה מעמיקה הבנתי שישנה דרך לצמצם את האוטומט-הרי התלות היא מהורה לילד -לכן האחוזים בין האבא- של הנשא לאמא של נשא ישארו אותו דבר- בלי קשר למצב הקודם אבל האחוזים של הורים של האבא (לדוגמא) ישתנו בהתאם למצב הגנטי של האבא כי בינו לבין הוריו יש תלות (וכן לגבי הסבא והסבתא -ביניהם אין תלות) ולכן סירטטתי את האוטומט הבא:



בעצם כדי להתחיל את האוטומט יצטרכו להקיש A ואז נגיע לq1 שזה המצב בו מכניסים את המידע על הורה אחד (נתחיל מהאבא) 0-בריא, 1-נשא, 2-חולה בהתאם לקלט נגיע לאחד המצבים הבאים Q2/Q3/Q4 -האחוזים התווספו בהתאם ואז נכניס את המידע עבור הוריו (נתחיל מהסבא) נקיש את מצבו ונגיע לאחד המצבים המקבלים -התווספו האחוזים בהתאם ואז המשתמש יצטרך להקיש B כדי לחזור למצב הקודם שבו הוא היה- כדי להקיש מידע על הסבתא-לדוגמא אם הוא היה במצב 2Q והוא הקיש שסבא שלו בריא אז הוא הגיע לq5 ואז הוא יצטרך להקיש B כדי לחזור אחורה לq2 כדי להקיש את המידע של הסבתא-כי הסבא והסבתא לא תלויים אחד בשני ולכן האחוזים שהתווספו הם אותו דבר אבל האבא כן תלוי בהם ולכן האחוזים של הסבא והסבתא ישתנו בהתאם לקלט שהוכנס לאבא של הנשא, לאחר שהמשתמש הקיש פרטים על הסבתא הוא שוב מגיע למצב מקבל,מתווספים האחוזים ואז הוא יצטרך להקיש שוב A כדי לחזור חזרה לq1 כדי להקיש פרטים על אמא שלו וההורים שלה-ובעצם האוטומט יפעל שוב כמו שתארתי וכשהוא יסיים למלא את הפרטים האוטומט יופעל שוב על המשתמש השני (הנשאית) כדי למלא את כל עץ המשפחה שלה לאחר מכן יוחזר הסיכויי הסופי למשתמש מהאוטומט

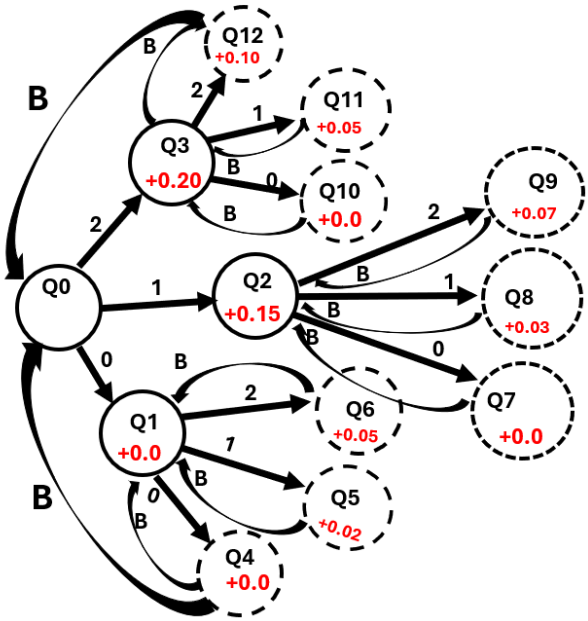
לסיכום הסדר של הכנסת הנתונים באוטומט הוא:

- נשא 1:
  1. אבא
  2. סבא מצד אבא
  3. סבתא מצד אבא
  4. אמא
  5. סבא מצד אמא
  6. סבתא מצד אמא

- נשא 2:
  1. אבא
  2. סבא מצד אבא
  3. סבתא מצד אבא
  4. אמא
  5. סבא מצד אמא
  6. סבתא מצד אמא

ולכן עבור האוטומט הזה יש בסך הכל 13 מצבים (במקום 1092!!)

לאחר מחשבה נוספת רציתי לייעל את האוטומט אפילו יותר את האוטומט אז חשבתי על הסירטוט הזה-



השינוי הוא שהמשתמש לא יצטרך להקיש את האות A כדי להתחיל אלא מיד הוא יכניס את הפרטים עבור אביו, וכן הוא יקיש רק B כדי לחזור-אז בעצם צימצמתי את המילה בשפה (הורדתי את A) וכן הורדתי את מספר המצבים באוטומט (בסך הכל 12 מצבים)

אז התחלתי לכתוב את האוטומט בשפת C++

בהתחלה יצרתי enum המכיל לי שמות מצבים-

```
enum class State { Q0, Q1, Q2, Q3, Q4, Q5, Q6, Q7, Q8, Q9, Q10, Q11, Q12 };
```

ואז מימשתי אותו באמצעות 2 טבלאות -טבלה אחת עבור המצבים וטבלה שניה עבור האחוזים-

```
//הגדרת טבלת מעברים
std::map<State, std::map<char, State>> transition;
transition[Q1]['0'] = Q2;
transition[Q1]['1'] = Q3;
transition[Q1]['2'] = Q4;
transition[Q2]['B'] = Q5;
transition[Q3]['B'] = Q9;
transition[Q4]['B'] = Q11;
transition[Q5]['0'] = Q6;
transition[Q5]['1'] = Q7;
transition[Q5]['2'] = Q12;
transition[Q9]['0'] = Q8;
transition[Q9]['1'] = Q7;
transition[Q9]['2'] = Q10;
transition[Q11]['0'] = Q11;
transition[Q11]['1'] = Q12;
transition[Q11]['2'] = Q12;
transition[Q6]['B'] = Q1;
transition[Q7]['B'] = Q1;
transition[Q8]['B'] = Q1;
transition[Q10]['B'] = Q1;
```

```
//הגדרת אחוזים
std::map<State, std::map<char, double>> percentages;
percentages[Q1]['0'] = 0.0;
percentages[Q1]['1'] = 0.15;
percentages[Q1]['2'] = 0.20;
percentages[Q5]['0'] = 0.0;
percentages[Q5]['1'] = 0.02;
percentages[Q5]['2'] = 0.05;
percentages[Q9]['0'] = 0.0;
percentages[Q9]['1'] = 0.03;
percentages[Q9]['2'] = 0.07;
percentages[Q11]['0'] = 0.0;
percentages[Q11]['1'] = 0.05;
percentages[Q11]['2'] = 0.10;
```

כאשר התחלתי לכתוב את הקוד הייתי צריכה לוודא שהמשתמש לא יקיש יותר מידי פעמים את האות B לכן יצרתי כמה משתנים שיעזרו לי בכך – מהקוד החלקי שמפורט כאן

בס"ד

רעות כהן

עשיתי לולאת for שתעבור פעמיים-כל פעם עבור נשא אחד שממלא את עץ משפחתו

המשתנה parentCount שבתוך הwhile נועד כדי להפסיק את האוטומט ברגע שעברנו על 2 ההורים של הנשא הנבדק כעת

אם אנחנו במצב 0Q אז כאשר נקיש B ההצבעה תיהיה למצב הקודם (לדוגמא-מ4q נחזור למצב הקודם שלו 1Q)

נכנס לתנאי הראשון כאשר עברנו כבר על הצד של האבא של הנשא וכעת אנחנו באמא והתנאי הנוסף כאשר הגענו לאחד המצבים הנל-נשנה את ההצבעה של ה nextבחזרה לq0 כדי שכעת נחזור לבדיקה על האמא

```
for(int i=0;i<2;i++){
while (parentCount < 2) {
  if (currentState == Q0) {
    transition[Q4]['B'] = Q1;
    transition[Q5]['B'] = Q1;
    transition[Q6]['B'] = Q1;
    transition[Q7]['B'] = Q2;
    transition[Q8]['B'] = Q2;
    transition[Q9]['B'] = Q2;
    transition[Q10]['B'] = Q3;
    transition[Q11]['B'] = Q3;
    transition[Q12]['B'] = Q3;
if (parentCount == 1){
      parent = "Mother";
    std::cout << "Input infornation for your "<< parent<< "- press 0 (healty) / 1 (carrier) / 2 (sick) or U to change your last choice  " << std::endl;
    std::cin >> input;
    next = transition[currentState][input];  }
  else if (currentState == Q1 || currentState == Q2 || currentState == Q3) {
    std::cout << "Input infornation for your grandfather- press 0 (healty) / 1 (carrier) / 2 (sick) or U to change your last choice  " << std::endl;
    std::cin >> input;
    next = transition[currentState][input];
    transition[next]['B'] = Q0;        }
  else if (currentState == Q4 || currentState == Q5 || currentState == Q6 || currentState == Q7 || currentState == Q8 || currentState == Q9 ||
currentState == Q10 || currentState == Q12)  {
    std::cout << "Input infornation for your grandmather- press 0 (healty) / 1 (carrier) / 2 (sick) or U to change your last choice  " << std::endl;
    std::cin >> input;
    totalPercentage += percentages[currentState][input];
    next = transition[currentState][input];  }
else  {
  std::cout << "Input infornation for your grandmather- press 0 (healty) / 1 (carrier) / 2 (sick) or U to change your last choice  ";
    std::cin >> input;
 next = transition[currentState][input];}
 if (currentState != Q0) {
   totalPercentage += percentages[currentState][input];}
  if (currentState == Q0) {
   parentCount++;  }}
```

אבל לאחר שכתבתי את הקוד עדיין הרגשתי שחסרים עוד כמה דברים כדי להפוך את האוטומט לטוב יותר,אחרי מחשבה מעמיקה החלטתי-

1. כמה שפחות להשתמש בif
2. שהמשתמש לא יצטרך להקיש כלל את האות B
3. לעשות בדיקות תקינות קלט כדי למנוע מהמשתמש להכניס קלט שגוי לחישוב- הקלטים המותאמים שיכולים להיות הם 0,1,2

כשהסתכלתי שוב על האוטומט הגעתי למסקנה שאני יכולה לעשות אותו אפילו יותר מדוייק-

ריציתי לכסות כל מקרה קצה שיכול להיות וככה למנוע טעויות בחישוב

מאחר ולכל אדם יש 2 עותקים אז החוק עובד כך:

לאדם בריא-יש 2 עותקים תקינים

אדם נשא -עותק פגום ועותק תקין

אדם חולה-2 עותקים פגומים

כל הורה מוריש לצאצאיו עותק אחד

ולכן אם המשתמש הקיש לדוגמא שאבא שלו בריא-כלומר שהוא ירש 2 עותקים תקינים

אז הוא לא יכול להקיש שהסבא\הסבתא חולים

וכן אם הוא הקיש שהאבא חולה-כלומר שהוא ירש 2 עותקים פגומים

אז הוא לא יכול להקיש שהסבא\הסבתא בריאים

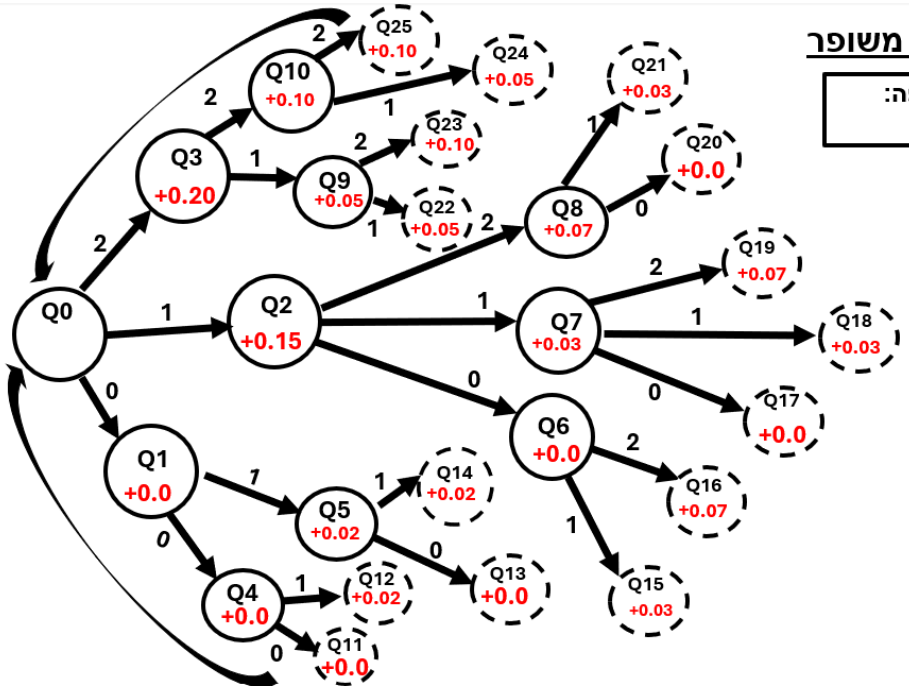
וכן אם הוא הקיש שאבא שלו נשא-אז הוא לא יכול להקיש שגם הסבא וגם הסבתא בריאים\חולים אלא הוא יכול להקיש ששתיהם נשאים או שאחד מהם חולה ואחד מהם בריא.

וכן אם המשתמש הקיש שאבא שלו חולה -הוא לא יכול להקיש שאמא שלו גם חולה-כי המשתמש בעצמו הוא נשא שזה אומר שהוא חייב לרשת עותק תקין אחד מהוריו

וכן אם המשתמש הקיש שאבא שלו בריא -הוא לא יכול להקיש שאמא שלו גם בריאה-כי המשתמש בעצמו הוא נשא שזה אומר שהוא חייב לרשת עותק פגום אחד מהוריו.

בעקרון הייתי יכולה להתעלם מזה ולסמוך על המשתמש שיקיש את המצב הנכון עבור כל אחד מבני משפחתו, אבל רציתי לקחת אחריות גם על זה ולוודא שלא יהיו טעויות בחישוב.

לכן עשיתי סירטוט חדש של האוטומט-



לאחר שסירטטתי את האוטומט שיניתי גם את הקוד ולאחר עבודה רבה פתרתי כל אחת מהבעיות האלו

1. הגדרתי כל צומת שתיהיה struct ועל ידי כך צימצמתי את השימוש בif, ואפילו את השימוש בטבלה נוספת עבור האחוזים
2. הכנסתי למבנה משתנה בוליאני שיתאר האם המצב הנוכחי הוא מצב מקבל או לא וכך לא הייתי צריכה את השימוש בB
3. הכנסתי למבנה וקטור שיכיל לי רק הקלטים המותאמים לאותו מצב(לכל מצב יש קלטים תקינים שונים)
4. נתתי למשתמש אפשרות לחזור אחורה ולשנות את מה שהוא הקיש לפני, במידה והוא הקיש מידע שגוי(אבל אחד מהקלטים התקינים) עבור אחד מבני משפחתו
5. הוספתי פעולה לחישוב נוסף על ידי מצב הבריאותי של האחים של כל משתמש

תיאור האוטומט הסופי

אחרי חודשים של עבודה סופסוף יצרתי את האוטומט הכי מדוייק ומהיר שיכולתי לעשות בזמן שהיה נתון לי –

יצירת המבנה כולל הenum-

```
enum class State { Q0, Q1, Q2, Q3, Q4, Q5, Q6, Q7, Q8, Q9, Q10, Q11, Q12, Q13, Q14, Q15, Q16, Q17, Q18, Q19, Q20, Q21, Q22, Q23, Q24, Q25 };

// הגדרת מבנה נתונים עבור מעברים
struct TransitionData {
    State nextState; // מצב הבא
    double percentage; // אחוזים עבור כל מצב
    std::string message;// הודעה עבור כל מצב
    std::vector<char> validInputs; //  וקטור עבור קלטים תקינים
    bool isAcceptingState; //  שדה בוליאני למצב מקבל

};
```

פעולה לחישוב האחים-

```
double calculateSiblingsChance()
{
    std::cout << "Let's start with your siblings -\n";
    while (true)
    {
        unsigned int sickSiblings = 0;
        unsigned int carrierSiblings = 0;
        unsigned int numOfSiblings = 0;
        double siblingsPercent = 0;
        double siblingsChance = 0;

        // פונקציה (לחדר) מגדירה לאחות אלה מספרי תקין //
        auto getUnsignedIntInput = [](const std::string& prompt) -> unsigned int {//auto נותן לקומפיילר לחבין לבד את הטיפוס של הפונקציה.

            //[] כלום "ליבד" (לא "מיוניים)מסמן אאין טיפוס במשתנים שיוניים
            unsigned int value;
            while (true) {
                std::cout << prompt << std::endl;
                std::cin >> value;

                if (std::cin.fail()) {
                    std::cin.clear(); // מנקה את מצב השגיאה
                    std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n'); // מנקה את שורת הקלט
                    std::cout << "Invalid input. Please enter a whole number only." << std::endl;
                }
                else {
                    return value;
                }
            }
            };

        numOfSiblings = getUnsignedIntInput("Please enter the number of your siblings (if any):");

        if (numOfSiblings == 0)
            return 0.0;

        carrierSiblings = getUnsignedIntInput("Enter the number of your *carrier* siblings:");
        while (carrierSiblings > numOfSiblings) {
            std::cout << "Carrier siblings can't be more than the total number of siblings." << std::endl;
            carrierSiblings = getUnsignedIntInput("Enter the number of your *carrier* siblings:");
        }

        sickSiblings = getUnsignedIntInput("Enter the number of your *sick* siblings:");
        while (sickSiblings > numOfSiblings) {
            std::cout << "Sick siblings can't be more than the total number of siblings." << std::endl;
            sickSiblings = getUnsignedIntInput("Enter the number of your *sick* siblings:");
        }

        if (sickSiblings + carrierSiblings > numOfSiblings) {
            std::cout << "One or more of the inputs entered are incorrect. Please try again.\n" << std::endl;
        }
        else {
            siblingsPercent = 10.0 / numOfSiblings;
            siblingsChance = ((siblingsPercent * sickSiblings) + ((siblingsPercent / 2) * carrierSiblings)) / 100.0;
            std::cout << "The total chance contribution from siblings is: " << siblingsChance << std::endl;
            return siblingsChance;
        }
    }
}
```

חישוב האחים החולים מתבצע באופן יחסי כמובן והתוצאה תוסיף 10% (עבור כל צד, סך הכל 20%) לחישוב הכולל של האוטומט,

לכן כדי לחשב אני אקבל מהמשתמש את מספר האחים יחלק ב10 וכך אני אקבל את מספר האחוזים להוספה עבור כל אח חולה,

עבור אח נשא זה מספר האחים חלקי 20, אח בריא לא מוסיף כלום כמובן,

וזו הנוסחא שניסחתי:

$$\frac{10.0}{numOfSiblings} = siblingsPercent$$

$$\frac{(carrierSiblings \times \frac{siblingsPercent}{2}) + (siblingsPercent \times sickSiblings)}{100.0} = siblingsChance$$

בנוסף, בפעולה עצמה ווידאתי שהמשתמש יזין רק קלטים תקינים-שיהיו רק מסוג int ושמספר האחים הנשאים\החולים\שניהם ביחד לא יעלה על מספר האחים הכולל שהמשתמש הכניס

הגדרת האוטומט:

```
int main()

    std::map<State, std::map<char, TransitionData>> transitions;
    // הגדרת טבלת המעברים עם קלטים תקינים
    transitions[State::Q0]['0'] = { State::Q1, 0.8, "", {'0', '1', '2','U'},false };
    transitions[State::Q0]['1'] = { State::Q2, 0.15, "", {'0', '1', '2','U'},false };
    transitions[State::Q0]['2'] = { State::Q3, 0.20, "", {'0', '1', '2','U'},false };

    transitions[State::Q1]['0'] = { State::Q4, 0.8, "Input information for your grandfather- press 0 (healty) / 1 (carrier) or U to change your last choice", {'0', '1','U'},false };
    transitions[State::Q1]['1'] = { State::Q5, 0.02, "Input information for your grandfather- press 0 (healty) / 1 (carrier) or U to change your last choice", {'0', '1','U'},false };

    transitions[State::Q2]['0'] = { State::Q6, 0.8, "Input information for your grandfather- press 0 (healty) / 1 (carrier) / 2 (sick) or U to change your last choice", {'0', '1', '2','U'} ,false };
    transitions[State::Q2]['1'] = { State::Q7, 0.03, "Input information for your grandfather- press 0 (healty) / 1 (carrier) / 2 (sick) or U to change your last choice", {'0', '1', '2','U'},false };
    transitions[State::Q2]['2'] = { State::Q8, 0.07, "Input information for your grandfather- press 0 (healty) / 1 (carrier) / 2 (sick) or U to change your last choice", {'0', '1', '2','U'},false };

    transitions[State::Q3]['1'] = { State::Q9, 0.05, "Input information for your grandfather- press 1 (carrier) / 2 (sick) or U to change your last choice", { '1', '2'},false };
    transitions[State::Q3]['2'] = { State::Q10, 0.1, "Input information for your grandfather- press 1 (carrier) / 2 (sick) or U to change your last choice", { '1', '2'},false };

    transitions[State::Q4]['0'] = { State::Q11, 0.0, "Input information for your grandmather- press 0 (healty) / 1 (carrier) or U to change your last choice", {'0', '1','U'},false };
    transitions[State::Q4]['1'] = { State::Q12, 0.02, "Input information for your grandmather- press 0 (healty) / 1 (carrier) or U to change your last choice", {'0', '1','U'},false };

    transitions[State::Q5]['0'] = { State::Q13, 0.0, "Input information for your grandmather- press 0 (healty) / 1 (carrier) or U to change your last choice", {'0', '1','U'},false };
    transitions[State::Q5]['1'] = { State::Q14, 0.02, "Input information for your grandmather- press 0 (healty) / 1 (carrier) or U to change your last choice", {'0', '1','U'},false };

    transitions[State::Q6]['1'] = { State::Q15, 0.03, "Input information for your grandmather- press 1 (carrier) / 2 (sick) or U to change your last choice", {'1', '2','U'},false };
    transitions[State::Q6]['2'] = { State::Q16, 0.07, "Input information for your grandmather- press 1 (carrier) / 2 (sick) or U to change your last choice", {'1', '2','U'},false };

    transitions[State::Q7]['0'] = { State::Q17, 0.0, "Input information for your grandmather- press 0 (healty) / 1 (carrier) / 2 (sick) or U to change your last choice", {'0', '1', '2','U'},false };
    transitions[State::Q7]['1'] = { State::Q18, 0.03, "Input information for your grandmather- press 0 (healty) / 1 (carrier) / 2 (sick) or U to change your last choice", {'0', '1', '2','U'},false };
    transitions[State::Q7]['2'] = { State::Q19, 0.07, "Input information for your grandmather- press 0 (healty) / 1 (carrier) / 2 (sick) or U to change your last choice", {'0', '1', '2','U'},false };


    transitions[State::Q8]['0'] = { State::Q20, 0.0, "Input information for your grandmather- press 0 (healty) / 1 (carrier) or U to change your last choice", {'0', '1','U'},false };
    transitions[State::Q8]['1'] = { State::Q21, 0.03, "Input information for your grandmather- press 0 (healty) / 1 (carrier) or U to change your last choice", {'0', '1','U'},false };

    transitions[State::Q9]['1'] = { State::Q22, 0.05, "Input information for your grandmather- press 1 (carrier) / 2 (sick) or U to change your last choice", {'1', '2','U'},false };
    transitions[State::Q9]['2'] = { State::Q23, 0.1, "Input information for your grandmather- press 1 (carrier) / 2 (sick) or U to change your last choice", {'1', '2','U'},false };

    transitions[State::Q10]['1'] = { State::Q24, 0.05, "Input information for your grandmather- press 1 (carrier) / 2 (sick) or U to change your last choice", {'1', '2','U'},false };
    transitions[State::Q10]['2'] = { State::Q25, 0.1, "Input information for your grandmather- press 1 (carrier) / 2 (sick) or U to change your last choice", {'1', '2','U'},false };

    transitions[State::Q11]['B'] = { State::Q0, 0.0, "Input information for your mother- press 0 (healty) / 1 (carrier) / 2 (sick) or U to change your last choice", {'B','U'},true };
    transitions[State::Q12]['B'] = { State::Q0, 0.0, "Input information for your mother- press 0 (healty) / 1 (carrier) / 2 (sick) or U to change your last choice", {'B','U'},true };
    transitions[State::Q13]['B'] = { State::Q0, 0.0, "Input information for your mother- press 0 (healty) / 1 (carrier) / 2 (sick) or U to change your last choice", {'B','U'},true };
    transitions[State::Q14]['B'] = { State::Q0, 0.0, "Input information for your mother- press 0 (healty) / 1 (carrier) / 2 (sick) or U to change your last choice", {'B','U'},true };
    transitions[State::Q15]['B'] = { State::Q0, 0.0, "Input information for your mother- press 0 (healty) / 1 (carrier) / 2 (sick) or U to change your last choice", {'B','U'},true };
    transitions[State::Q16]['B'] = { State::Q0, 0.0, "Input information for your mother- press 0 (healty) / 1 (carrier) / 2 (sick) or U to change your last choice", {'B','U'},true };
    transitions[State::Q17]['B'] = { State::Q0, 0.0, "Input information for your mother- press 0 (healty) / 1 (carrier) / 2 (sick) or U to change your last choice", {'B','U'},true };
    transitions[State::Q18]['B'] = { State::Q0, 0.0, "Input information for your mother- press 0 (healty) / 1 (carrier) / 2 (sick) or U to change your last choice", {'B','U'},true };
    transitions[State::Q19]['B'] = { State::Q0, 0.0, "Input information for your mother- press 0 (healty) / 1 (carrier) / 2 (sick) or U to change your last choice", {'B','U'},true };
    transitions[State::Q20]['B'] = { State::Q0, 0.0, "Input information for your mother- press 0 (healty) / 1 (carrier) / 2 (sick) or U to change your last choice", {'B','U'},true };
    transitions[State::Q21]['B'] = { State::Q0, 0.0, "Input information for your mother- press 0 (healty) / 1 (carrier) / 2 (sick) or U to change your last choice", {'B','U'},true };
    transitions[State::Q22]['B'] = { State::Q0, 0.0, "Input information for your mother- press 0 (healty) / 1 (carrier) / 2 (sick) or U to change your last choice", {'B','U'},true };
    transitions[State::Q23]['B'] = { State::Q0, 0.0, "Input information for your mother- press 0 (healty) / 1 (carrier) / 2 (sick) or U to change your last choice", {'B','U'},true };
    transitions[State::Q24]['B'] = { State::Q0, 0.0, "Input information for your mother- press 0 (healty) / 1 (carrier) / 2 (sick) or U to change your last choice", {'B','U'},true };
    transitions[State::Q25]['B'] = { State::Q0, 0.0, "Input information for your mother- press 0 (healty) / 1 (carrier) / 2 (sick) or U to change your last choice", {'B','U'},true };
```

הגדרת המשתנים:

לצורך החישוב ->

```
double SumTotalPercentage = 0;//סוכם את האחוזים של 2 הנשאים
std::stack<State> stateHistory; // מחסנית להיסטוריית מצבים
std::stack<double> percentageHistory; // מחסנית להיסטוריית אחוזים
double finalCalculation = 0;//החישוב הסופי
double siblingsChance = 0;//מתאר את אחוז הסיכויים לחלות על ידי האחים
```

התחלת ה—for עובר פעמיים עבור הנשא והנשאית, בכל תחילת איטרציה נכניס למשתנה של אחוז האחים את התוצאה שנקבל מהפונקציה לחישוב שהראיתי לעיל.

אני מאתחלת בכל איטרציה את ההודעה למשתמש כי היא עתידה להשתנות במהלך התוכנית,

ברירת המחדל של הקלט היא 1 (כי 1 נמצא בכל מצב אפשרי),מתחילים במצב 0Q

```
for (int i = 0;i < 2;i++)
{
    double siblingsChance = calculateSiblingsChance();
    int parentCount = 0;//סופר את ההורים
    transitions[State::Q0]['0'] = { State::Q1, 0.0, "Input information for your father- press 0 (healty) / 1 (carrier) / 2 (sick) or U to change your last choice", {'0', '1', '2','U'},false };
    transitions[State::Q0]['1'] = { State::Q2, 0.15, "Input information for your father- press 0 (healty) / 1 (carrier) / 2 (sick) or U to change your last choice", {'0', '1', '2','U'},false };
    transitions[State::Q0]['2'] = { State::Q3, 0.20, "Input information for your father- press 0 (healty) / 1 (carrier) / 2 (sick) or U to change your last choice", {'0', '1', '2','U'},false };
    char input = '1';
    State currentState = State::Q0;
    double totalPercentage = 0;//מחשב את האחוזים עבור כל נשא

    while (parentCount < 2)
```

תיאור הwhile -הוא קורה עד שמונה ההורים שווה ל2, זה אומר בעצם שעברנו על 2 ההורים של המשתמש ומלאנו את כל העץ עבור המשתמש הנוכחי,

קבלת נתונים מהשתמש,

במידה והוא הקיש U והמחסניות לא ריקות נחזיר בחזרה את האחוזים והמצב לצעד הקודם,במידה וריק תוחזר הודעה מתאימה.

בדיקת קלט מהמשתמש,

ושמירת האחוזים והמצב במחסניות המתאימות

```
while (parentCount < 2)
{
    std::cout << transitions[currentState][input].message << std::endl;
    std::cin >> input;

    // חזרה אחורה
    if (input == 'U')
    {
        if (!stateHistory.empty() && !percentageHistory.empty())
        {
            currentState = stateHistory.top();
            stateHistory.pop();
            totalPercentage = percentageHistory.top();
            percentageHistory.pop();
            std::cout << "Returned to previous state. Chance so far: " << totalPercentage << std::endl;
            continue;
        }
        else
        {
            std::cout << "No previous state to return to." << std::endl;
            continue;
        }
    }

    // בדיקת קלט תקין לפי הוקטור
    bool validInput = false;
    for (char validChar : transitions[currentState][input].validInputs)
    {
        if (input == validChar)
        {
            validInput = true;
            break;
        }
    }
    if (!validInput) {
        std::cout << "Invalid input. Please try again." << std::endl;
        continue;
    }

    // שמירת היסטוריה
    stateHistory.push(currentState);
    percentageHistory.push(totalPercentage);
```

המשך- הגדרתי משתנה נוסף מסוג המבנה שיצרתי- transitionData כדי לשמור את כל התוכן של המצב הנוכחי ,

הוספתי בדיקה על מנת למנוע מהמשתמש להקיש מידע שגוי בפעם הבאה-אם המשתמש כרגע נמצא בq0 כלומר על אביו- ואם הוא הקיש 2 אז נשנה את תוכן וקטור הקלטים לכך שהמשתמש לא יוכל להקיש בפעם הבאה על אמו את הערך 2 , וכן לגבי הערך 0.(הסברתי למעלה את הסיבה)

נשנה את המצב הנוכחי למצב הבא,

נסכום את האחוזים של המצב הנוכחי,

נכניס למשתנה בוליאני את תיאור המצב הנוכחי-אם הוא מצב מקבל או לא,

נדפיס הודעה למשתמש (לא חייב אבל זה כדאי כדי לבדוק איפה אנחנו נמצאים) וכן נדפיס את סך האחוזים שנצברו בינתיים ואת המצב הנוכחי בו אני נמצאים (שוב, לא חייב)

במידה והגענו למצב מקבל-נסכום את האחוזים, נחליף את ההודעה בq0 להודעה מתאימה להכנסת מידע עבור האמא ונאפס בחזרה את totalpercent על מנת לסכום כעת את האחוזים עבור האמא,

נרוקן את המחסניות ונאפס את המצב הנוכחי למצב 0Q,

ונעלה את משתנה parent count ב1 על מנת להפסיק את ה whileלאחר שהנשא הראשון סיים למלא את עץ המשפחה שלו.

```
TransitionData transitionData = transitions[currentState][input];


if (currentState == State::Q0)
{
    if (input == '2')
    {
        transitions[State::Q0]['0'].validInputs = { '0', '1','U' };
        transitions[State::Q0]['1'].validInputs = { '0', '1','U' };
        transitions[State::Q0]['2'].validInputs = { '0', '1','U' };
    }
    else
        if (input == '0')
        {
            transitions[State::Q0]['0'].validInputs = { '1', '2','U' };
            transitions[State::Q0]['1'].validInputs = { '1', '2','U' };
            transitions[State::Q0]['2'].validInputs = { '1', '2','U' };
        }
}

currentState = transitionData.nextState;
totalPercentage += transitionData.percentage;
bool isAcceptingState = transitions[currentState]['B'].isAcceptingState;
std::cout << "isAcceptingState: " << isAcceptingState << std::endl;


std::cout << "Updated chance: " << totalPercentage << std::endl;
std::cout << "Current state: " << static_cast<int>(currentState) << std::endl;

if (isAcceptingState)
{
    SumTotalPercentage += totalPercentage;
    transitions[State::Q0]['0'].message = "Input information for your mather- press 0 (healty) / 1 (carrier) / 2 (sick) or U to change your last choice";
    transitions[State::Q0]['1'].message = "Input information for your mather- press 0 (healty) / 1 (carrier) / 2 (sick) or U to change your last choice";
    transitions[State::Q0]['2'].message = "Input information for your mather- press 0 (healty) / 1 (carrier) / 2 (sick) or U to change your last choice";
    totalPercentage = 0;
    //ריקון המחסניות
    while (!stateHistory.empty()) stateHistory.pop();
    while (!percentageHistory.empty()) percentageHistory.pop();
    currentState = State::Q0;
    parentCount++;
}

}
```

החישוב הסופי והדפסת התוצאה:

בתוך ה whileכל האחוזים נסכמים בתוך המשתנה totalpercent לאחר שיצאנו מה while ועוד לא עברנו לאיטרציה חדשה בfor נכניס את התוצאה שבtotalpercent לתוך המשתנה sum totalpercent אבל חלקי 2 כי כל נשא מהווה מחצית מתוך סך כל האחוזים וכן נכנס בכל איטרציה האחוזים של האחים ולבסוף נכניס את האחוזים המשוכללים של שניהם שבתוך sum totalpercent לתוך המשתנה finalcaculation ונחלק ב75 כי הסכום הכולל ל שניהם מהווה 75% מתוך כל החישוב וגם נוסיף למשתנה את ה25%-המצב ההתחלתי וקיבלנו את התוצאה של הבדיקה.

ולבסוף נדפיס הודעה מתאימה למשתמש.

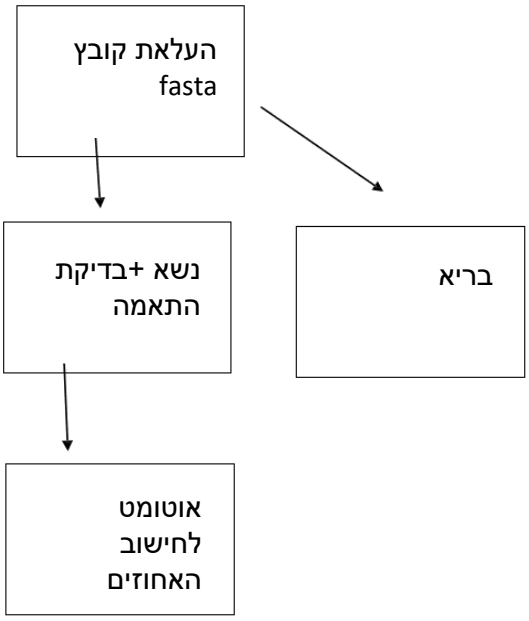
-הסוף-

```
    //totalPercentage = static_cast<int>(totalPercentage) % 2;
    std::cout << "your current state is " << static_cast<int>(currentState) << std::endl;

    //כאן לחשב את הסיכוי הסופי מ2 הנשאים
   // std::cout << "the final chance: " << totalPercentage << std::endl;
    std::cout << "the siblings chance: " << siblingsChance << std::endl;

    //מחשב את האחוז הסופי של הנשא והאחים שלו ומוסיף למשתנה שמשכלל את כל החישוב הסופי
    SumTotalPercentage += (static_cast<int>(totalPercentage) / 2 + static_cast<int>(totalPercentage) % 2) + siblingsChance;
    std::cout << "SumTotalPercentage: " << SumTotalPercentage/2 << "% " << std::endl;
    siblingsChance = 0;//מנקה את האחוזים של האחים
}

//חישוב סופי של 2 הנשאים כולל האחים
finalCalculation = ((SumTotalPercentage/2) * 0.75 + 0.25) * 100.0;
std::cout << "---------------------------------- "<< std::endl;
std::cout << "the final calculation is: " << finalCalculation << "% " << std::endl;
std::cout << "---------------------------------- " << std::endl;

return 0;
};
```

נשמר ב- מחשב זה

תיאור מסכים

תרשים מסכים



צילום מסכים

המשתמש מפעיל את התוכנה ומקבל את הפלט הבא-

```
Enter the path to the FASTA file:
```

לאחר שהמשתמש מכניס את הקובץ תוצג לו הרשימה של המחלות שעליהן הוא נבדק עם המצב הבריאותי שלו עבור כל אחת מהן,במידה והוא נשא לפחות לאחת מהן הוא המשתמש השני יצטרך להכניס את הרצף שלו, במידה והוא לא נשא לאחת מהמחלות תוצג לו הודעה בהתאם

במידה והוא לא נשא:

```
Enter the path to the FASTA file: C:\Users\ADMIN\Documents\gene_project\fasta files\Xbreak\Sample_Fragile_X_Carrier.fasta

Carrier status:
Sickle Cell Anemia: Not Carrier
Tay Sachs: Not Carrier
Pku: Not Carrier
Cystic Fibrosis: Not Carrier

This individual is not a carrier for those diseases.
```

במידה והוא נשא:

```
Carrier status:
Sickle Cell Anemia: Not Carrier
Tay Sachs: Not Carrier
Pku: Not Carrier
Cystic Fibrosis: Carrier

This individual is a carrier for at least one disease.
Enter the path to the FASTA file:
```

אם המשתמש הראשון היה נשא המשתמש השני התבקש להכניס את הקובץ שלו, לאחר שהמשתמש השני הכניס את הקובץ תוצג לו הרשימה של המחלות שעליהן הוא נבדק עם המצב הבריאותי שלו עבור כל אחת מהן,במידה והוא נשא לפחות לאחת מהן וקיימת התאמה בינו לבין המשתמש הראשון תוצג הודעה להכנסת פרטי עץ המשפחה,במידה ולא קיימת התאמה תוחזר הודעה מתאימה.

במידה ולא קיימת התאמה

```
Enter the path to the FASTA file: C:\Users\ADMIN\Documents\gene_project\fasta files\Xbreak\Sample_Fragile_X_Carrier.fasta

Carrier status:
Sickle Cell Anemia: Not Carrier
Tay Sachs: Not Carrier
Pku: Not Carrier
Cystic Fibrosis: Not Carrier

This individual is not a carrier for those diseases.

There is no matching carrier status between the two individuals.
```

במידה וקיימת התאמה

```
Enter the path to the FASTA file: C:\Users\ADMIN\Documents\gene_project\fasta files\Cystic Fibrosiss\Carrier_CFTR4.fasta

Carrier status:
Sickle Cell Anemia: Not Carrier
Tay Sachs: Not Carrier
Pku: Not Carrier
Cystic Fibrosis: Carrier

This individual is a carrier for at least one disease.

Warning: Both individuals are carriers for the following diseases:
- Cystic Fibrosis
In order to calculate the results, please enter data about your family.
```

במידה והייתה קיימת התאמה-המשתמשים ימלאו את עץ משפחתם בהתחלה הם יכניסו את הפרטים על האחים שלהם בסוף יוחזר החישוב של האחים

```
user number 1 please enter the ditails for your family:
Let's start with your siblings -
Please enter the number of your siblings (if any):
3
Enter the number of your *carrier* siblings:
1
Enter the number of your *sick* siblings:
2
The total chance contribution from siblings is: 0.0833333
```

לאחר מכן, הם ימלאו את הפרטים על ההורים,הסבים והסבתות ולבסוף התקבל החישוב הסופי-

```
Input information for your father- press 0 (healty) / 1 (carrier) / 2 (sick)
0
isAcceptingState: 0
Updated chance: 0
Current state: 1
Input information for your grandfather- press 0 (healty) / 1 (carrier) or U to change your last choice
1
isAcceptingState: 0
Updated chance: 0.02
Current state: 5
Input information for your grandmather- press 0 (healty) / 1 (carrier) or U to change your last choice
1
isAcceptingState: 1
Updated chance: 0.04
Current state: 14
Input information for your mather- press 0 (healty) / 1 (carrier) / 2 (sick) or U to change your last choice
2
isAcceptingState: 0
Updated chance: 0.2
Current state: 3
Input information for your grandfather- press 1 (carrier) / 2 (sick) or U to change your last choice
2
isAcceptingState: 0
Updated chance: 0.3
Current state: 10
Input information for your grandmather- press 1 (carrier) / 2 (sick) or U to change your last choice
2
isAcceptingState: 1
Updated chance: 0.4
-----------------------------------
the final calculation is: 67.875%
-----------------------------------
```

מדריך למשתמש

דבר ראשון יש לבחור קובץ מתוך המחשב של רצף הגנים ולהכניס לתוכנה. במקרה שהתקבל והוא נשא הוא יכניס את הקובץ של המשתמש השני.

במקרה וקיימת נשאות המשתמשים ימלאו פרטים על בני משפחתם.

פיתוחים עתידיים

- אפשרות של הנדסה גנטית ושינוי המוטציות בגוף על ידי שהמערכת תגיד היכן הבעיה
- להוסיף ממשק משתמש נח
- להוסיף אבטחת מידע

ניתוח יעילות

במהלך כתיבת האלגוריתמים השתדלתי ככל האפשר לכתוב אותם ביעילות ולדאוג שגם הסיבוכיות זמן ומקום תיהיה יעילה.

בנוסף דאגתי שהקוד לא ימשיך לעבוד כאשר נמצא פתרון, וכן שהתוכנה לא תיפול כאשר המשתמש מכניס קלט שגוי-כמו העלאת קובץ ריק או הכנסת קלט לא חוקי וכו

אבטחת מידע

מידע שמשתמש מזין למערכת אינו מידע אישי וסודי כמו: סיסמא, תעודת זהות ועוד... הצורך אבטחה .בנוסף לכך, כאשר המשתמש עוזב את התוכנה כל המידע נאבד ואינו נשמר בשום מקום, ולכן אין צורך באבטחת מידע.

מסקנות

במבט לאחור לזמן בו החלתי לתכנן את הפרויקט שלי התחושה שליוותה אותי הייתה שהפרויקט הינו דבר עצום ובלתי עביר .הוא דרש ממני סבלנות והשקעה רבה בכל חלק וחלק ואפילו הקטן ביותר, בתכנון הפרויקט שלב אחר שלב ומימוש של כל אחד מהחלקים .היו ימים שבהם הכל זרם והלך בקלות אך היו זמנים שבדרך לא תמיד הייתה הצלחה, לא תמיד התוצאות באו בקלות ופעמים אף עבדתי חודשים על קודים בניסיון לפתור את הבעיה ובסוף הם לא נכללו באלגוריתם הסופי שלי .התוצאה שהגעתי אליה הוכחה לי שהמאמץ היה שווה וכל רגע היה כדי להגיע לשלב המרגש הזה .כמובן שעשיית הפרויקט תרמה לי המון במובן המקצועי ובהכנה לחיים עצמם. למדתי להציב לעצמי מטרה ולא לזוז ממנה עד שאני מצליחה ולא להתייאש אפילו שזה היה מייגע ולא ראיתי את הסוף, אבל עכשיו אני מבינה ורואה שהיה שווה המאמץ הגדול הזה. יש לי הרבה סיפוק ממנו הספקתי כל מה שרציתי שהצבתי לעצמי. בהתחלה הפרויקט היה נראה לי הר שאי אפשר לעבור ואז חילקתי את זה למטרות קטנות ולכן היה לי יותר קל.

השקעתי בפרויקט הזה את כל כולי והמסקנה שלי- "יגעת ומצאת תאמין."

בבילוגרפיה

ויקיפדיה

Python

genebank

gitHub

Stack Overflow

ויקירפואה

מכון דוידסון לרפואה

המכלול

דור ישרים

תודות

דבר ראשון תודה לבורא העולם שבלעדיו לא היה לי פרויקט והוא ליווה וילווה אותי תמיד.

תודה למנחת הפרוייקט המורה תמר זקס על הכוונה מראשית עד אחרית,הענקת תחושת ביטחון בעצמי ובפרוייקט, סיוע בכל הפרטים הגדולים והקטנים, מענה בכל שעה כמעט והכל במאור פנים, בשמחה וברוגע .תודה לרכזת המורה אלה גליק על אכפתיות ודאגה בכל המישורים, גם בזמנים קשים ומרכבים, על הקשבה לכל בעיה והכלה .תודה לריבקי ואיילה המתרגלות, על עידוד פרגון ודאגה,ותודה לחברותי למגמה שהיו שם תמיד בשבילי, בכשלונות ובהצלחות, בכאב ובשמחה. תודה על השתתפות והכלה, על עידוד ותמיכה, בלעדיכם לא הייתי מצליחה לעבור את הדרך הארוכה הזאת .יש לכן חלק גדול מאוד בפרוייקט שלי. תודה!