

ספר פרויקט – שחמט

לכתוב בגוף שלישי

מכללה: רב תחומי עמל ב

שם פרויקט: שחמט

שם מלא: מהדב איתי

ת.ז: 215300807

שם מנחה: נילי נווה

מוגש

תאריך הגשה:

תוכן עניינים

[1. תקציר 2](#__RefHeading___Toc2219_661954165)

[2. תיאור הנושא 3](#__RefHeading___Toc2221_661954165)

[3. רקע תיאורטי 6](#__RefHeading___Toc2223_661954165)

[4. מושגים 7](#__RefHeading___Toc2225_661954165)

[5. תיאור הבעיה האלגוריתמית 8](#__RefHeading___Toc2338_661954165)

[6. סקירת אלגוריתמים בתחום הבעיה 10](#__RefHeading___Toc2340_661954165)

[7. מבנה נתונים 12](#__RefHeading___Toc28352_1472221104)

[8. תיאור האלגוריתם הנבחר 14](#__RefHeading___Toc2342_661954165)

[9. האלגוריתם הראשי בפסואוד קוד 16](#__RefHeading___Toc1239_3620615532)

[10. ארכיטקטורה של הפתרון המוצע בפורמט Top-Down Level Design 20](#__RefHeading___Toc1428_2728458605)

[11. תרשים מקרי שימוש UML Use cases 21](#__RefHeading___Toc10599_899343872)

[12. תרשים סביבת העבודה ושפת התכנות 22](#__RefHeading___Toc10601_899343872)

[13. התכנית הראשית פסאודו קוד 22](#__RefHeading___Toc1238_1222312374)

[14. תיאור ממשק גרפי 23](#__RefHeading___Toc28356_1472221104)

[15. תיאור הפונקציות והמחלקות הראשיות בפרויקט 23](#__RefHeading___Toc10605_899343872)

# 1. תקציר

# 2. תיאור הנושא

הנושא הנבחר הוא משחק השחמט, משחק לוח המשוחק על לוח בעל 64 משבצות.

לכל שחקן ישנם 16 כלים בעלי 6 סוגים שונים. כל סוג כלי יכול לזוז בהתאם לכללי התזוזה שלו.

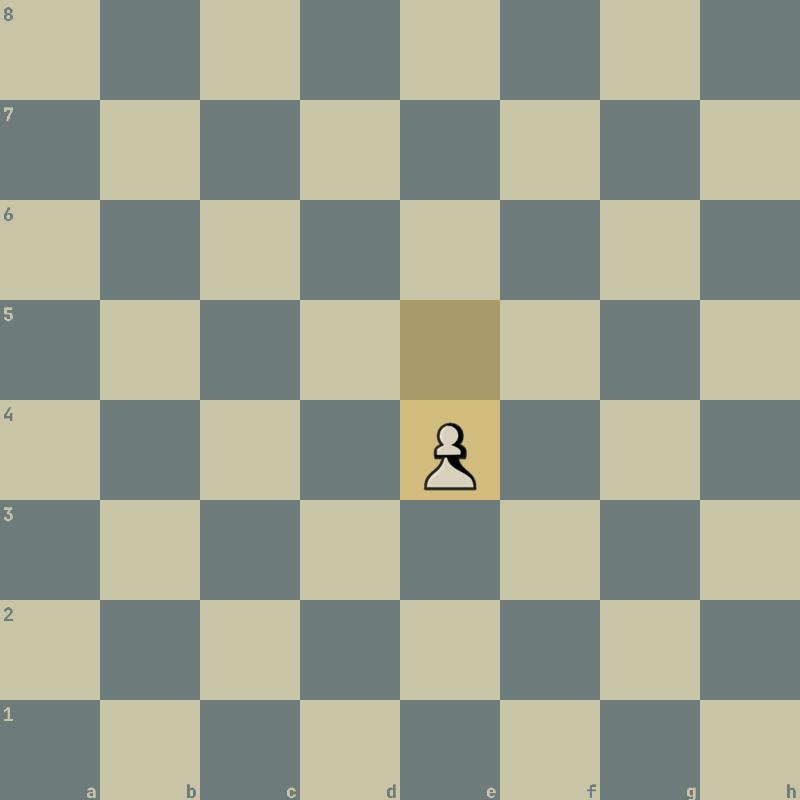
מטרתו של כל שחקן להזיז את כליו באופן שיגיע למט על היריב מבלי שיבצע מט עליו.

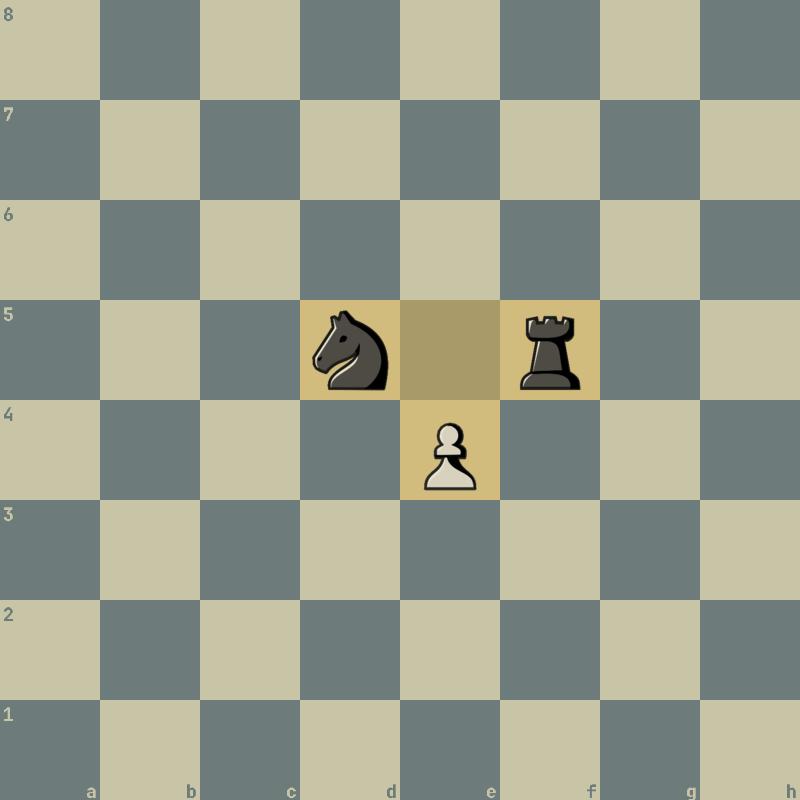
שחמט הוא משחק חשיבה וספורט תחרותי בין שני אנשים או יותר.

**כלי המשחק**

**- חייל**

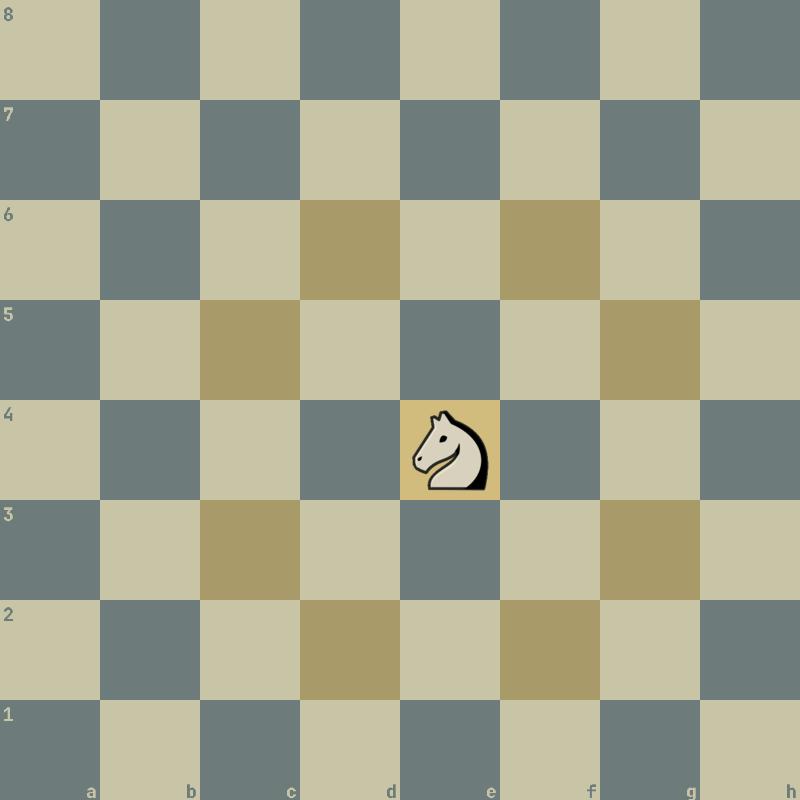
החייל יכול לזוז משבצת אחת קדימה בכל תור, למעט אם הוא עדיין לא זז ובמקרה כזה הוא יכול לזוז עד 2 משבצות קדימה. החייל יכול לאכול כל כלי הנמצא מלפניו באלכסון במרחק של משבצת אחת.





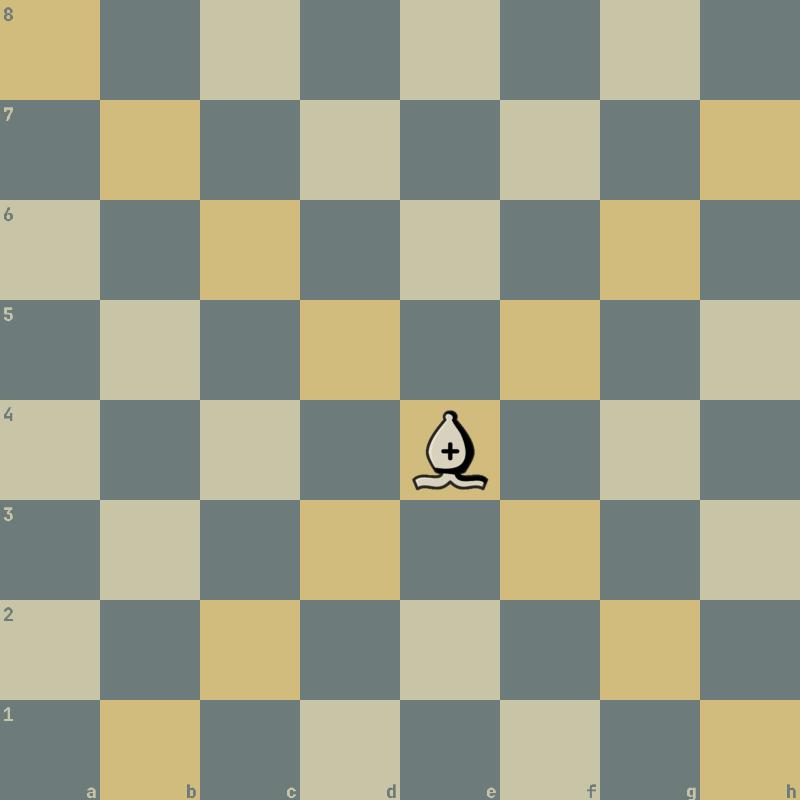
**- פרש**

הפרש יכול לזוז שתי משבצות בקו ישר ולאחר מכן משבצת אחד בפניה והוא השחקן היחיד שיכול לדלג מעל כלים אחרים. יכולת האכילה זהה ליכולת התזוזה.



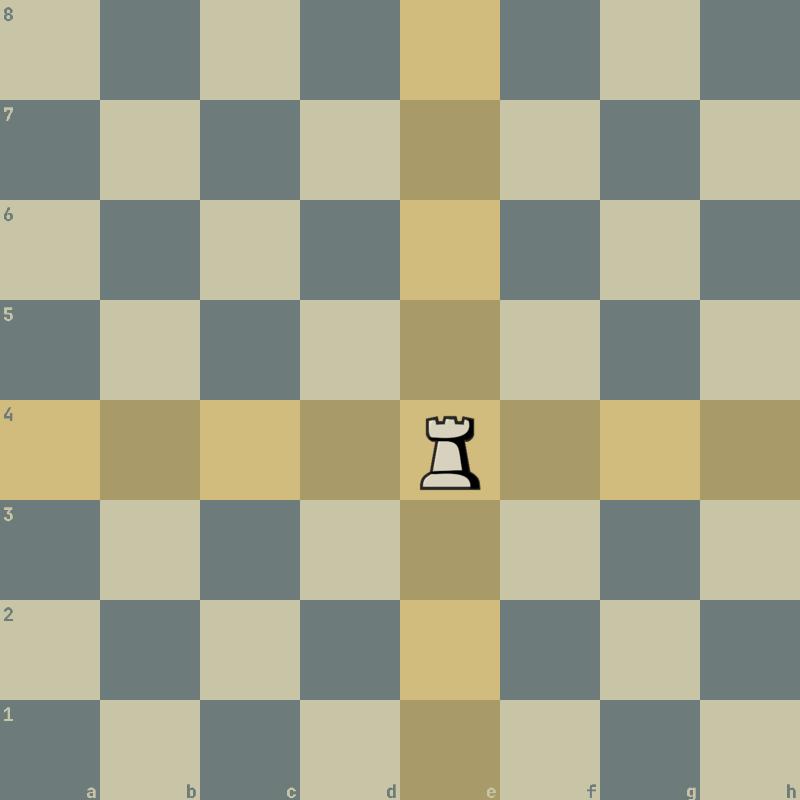
**- רץ**

הרץ זז באלכסונים לכל כיוון ואינו יכול לדלג על אף כלי. יכולת האכילה זהה.



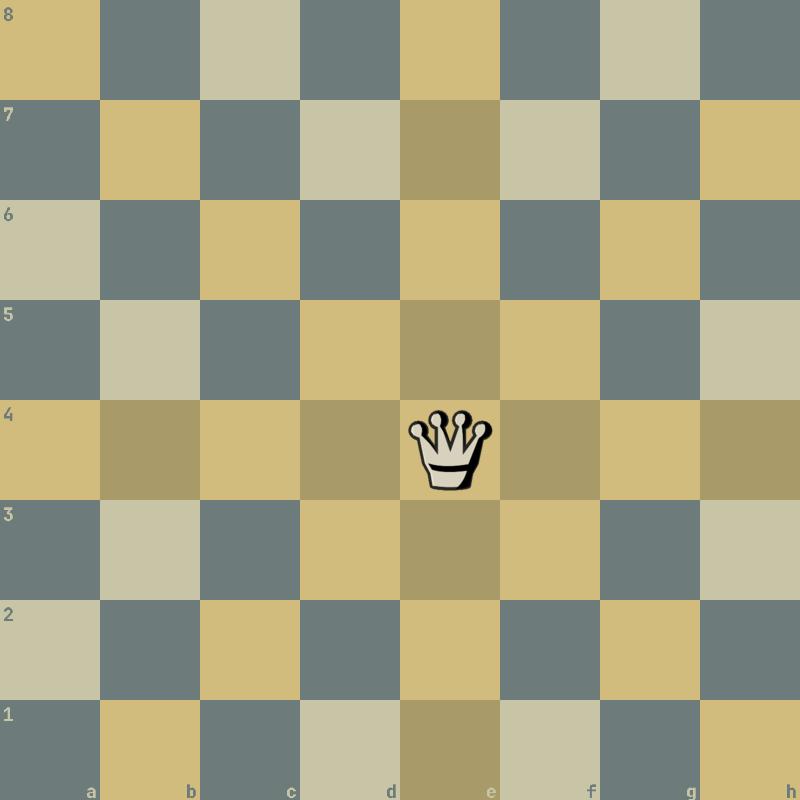
**- צריח**

הצריח נע בקווים ישרים לכל הכיוונים ואינו יכול לדלג על אף כלי. יכולת האכילה זהה.



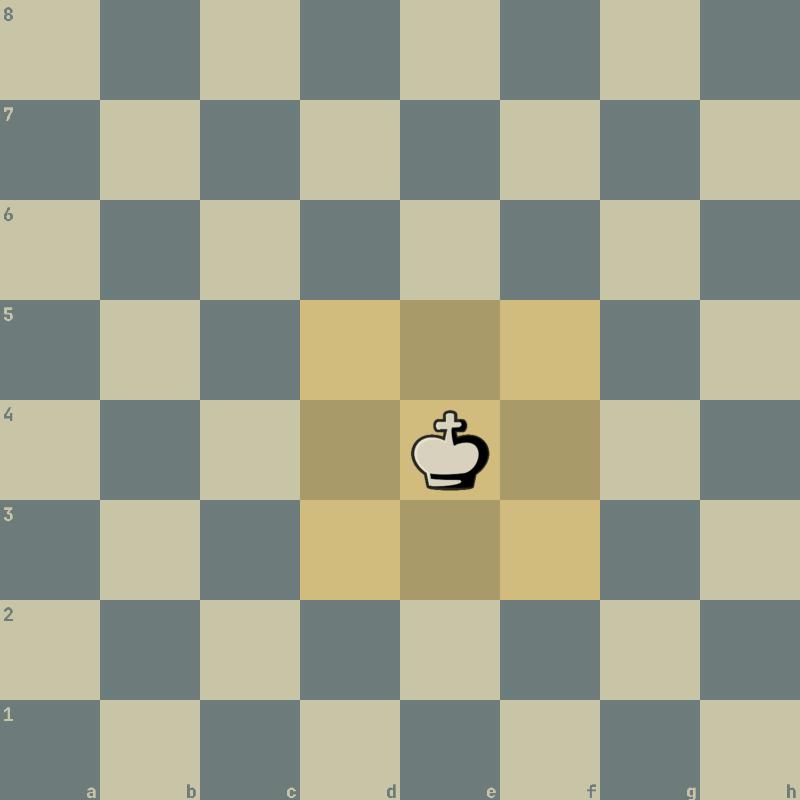
**- מלכה**

המלכה יכול לזוז או באלכסון לכל כיוון או בקו ישר לכל כיוון. אפשר להתייחס למלכה כאל חיבור של צריח ורץ. המלכה אינה יכולה לדלג על אף כלי .יכולת האכילה זהה.

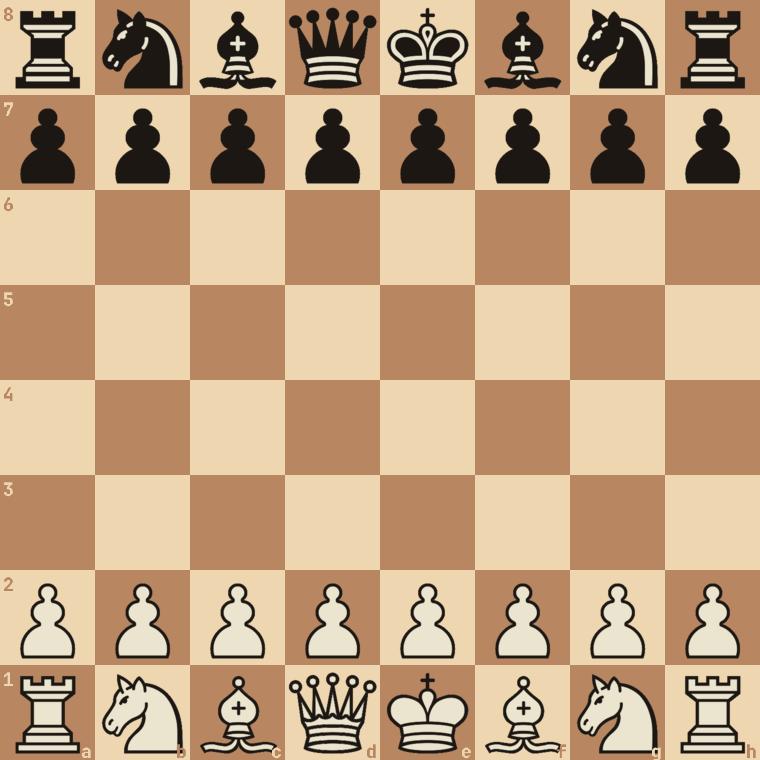


**- מלך**

המלך יכול לזוז משבצת אחת לכל הכיוונים כל עוד היא לא מאוימת ע"י השחקן השני. יכולת האכילה זהה.



מצב התחלתי של הלוח:



מהלך אחד

הfen של המצב ההתחלתי נראה כך:

**rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w**

כאשר:

- ‘/’ מייצג סוף שורה בלוח

- כל אות מייצגת כלי וצבעו (אות גדולה כלי לבן, אות קטנה כלי שחור)

- מספר מייצג את כמות המשבצות הריקות מאותו מיקום

- w כאשר תורו של השחקן הלבן, b כאשר תורו של השחקן השחור

לfen הרשמי חלקים נוספים כמו האם שחקן יכול להצריח, ולחוק "הכאה אגב הלוכו" שמתאר אכילה מיוחדת בין שני כלי חייל.

# 3. רקע תיאורטי

ככל הנראה שחמט מתבסס על משחק לוח ההודי צטורגנה אך אינינו בטוחים. המשחק התפשט באזור המאה השביעית ורק לאחר מכן במאה ה15 חוקי השחמט המוכרים לנו כיום הומצאו.

לאחר שהשחמט שאנו מכירים היום התפשט, שחקנים החליטו לארגן טורנירים שבהם שחקן אחד מתחרה עם שחקן אחר ומטרתו של כל אחד הוא לנצח. כיום אנו קוראים לשחקן שחמט מקצועי "שחמטאי". יש לציין כי השחמט המקצועי משוחק לצד שעון כך כשלכל שחקן יש זמן מוגבל לחשוב ולבצע מהלך. השחמט נחשב למשחק חשיבה מורכב שעל השחקנים להסתכל כמה צעדים קדימה כדי להחשיל את האויב ולנצח. שם המשחק נקרא ע"י כיתור המלך - שח וע"י כיתור המלך שאין לו אפשרות לבצע לברוח – "מט". ברגע ששחקן מבצע שחמט הוא מנצח.

לאחר אלן טיורינב

לשחמט ומדעי המחשב ישנה היסטוריה מעניינת, ב1948 אלן טיורינג ושמפרנאון יצרו את תכנית השחמט "*Turochamp"* המסוגלת לנצח בני אדם ברמה נמוכה ע"י חישוב האפשרויות ובחירת מהלך בהתאם, אלגוריתם זה נקרא minimax והוא משמש בלמציאת המהלך הטוב ביותר במגבלות המכונה. מאוחר יותר בשנות התשעים פותח"Deep Blue" מחשב מיוחד ע"י IBM כשמטרתו לשחק שחמט ברמה הגבוהה ביותר, המחשב ניצח בשני משחקים מתוך שישה נגד "גרי קספרוב", שחמטאי ברמה עולמית, וכעבור שנה פותח המחשב וניצח את גרי בשני משחקים מתוך שלושה כשאר במשחק השלישי היה תיקו. "Deep Blue" הוא המחשב הראשון שהביס שחמטאי ברמה עולמית.

כיום "stock fish" נחשב למנוע השחמט החזק ביותר.חלק מחשיבים את "alpha zero" שפותח ע"י גוגל למנוע שחמט הממוחשב החזק ביותר כאשר הוא ניצח את stockfish, אבל יש לזכור שalpha zero מבוסס על למידת מכונה שיושבת על מחשב על בעוד שstockfish היא תוכנה שניתן להריץ על כמעט כל מחשב הנמצא בשוק. יש לציין כי ל stockfish יש לא מעט גרסאות שמחולקות לפי תוכנה לוקאלית (על המחשב) ותוכנה בענן (על שרת), כאשר לתוכנה בענן היכולת לשחק שחמט ברמה גבוהה יותר כי יש לה גישה ליותר כוח עיבוד וגישה לנתונים רבים. למשל כשאר ישנם שבעה כלים על הלוח (לא משנה איזה סוג הכלים) לstockfish יש בסיס נתונים בעל 20 טרה המכיל את כל האפשרויות של המשחק ובמצב זה הוא יכול לנצח ב100 אחוז מהפעמים.

# 4. מושגים

**לוח המשחק 8\*8 –** מערך של מאסקים

**כלי** - ביט במאסק

**Fen** **–** מחרוזת המתארת את מצב הלוח ואת תורו של השחקן הנוכחי

פעולות:

**מהלך** - הזזה של כלי ממשבצת מקור למשבצת יעד בהתאם לכללי התזוזה של הכלי. כאשר במשבצת היעד אין כלי בעל אותו הצבע של הכלי המוזז.

**כלי מחליק** - כלי הנע אינסוף משבצות לכיוונים מסוימים עד לסוף הלוח או עד למשבצת של כלי יריב או עד משבצת לפני של כלי לא יריב.

**חוסם** - כלי הנמצא במשבצת שמגבילה את יכולת התזוזה של **כלי מחליק**

**אכילה** - **מהלך** הנגמר בהוצאת כלי מהמשחק.

**קידום - מהלך** הנגמר בהחלפת חייל למלכה כאשר משבצת היעד היא בשורה האחרונה של הלוח בהתאם לצד של כל שחקן

מצבים:

**איום** – מצב כאשר כלי בעל היכולת **לאכול** כלי אחר **במהלך** אחד.

**שח** – **איום** על המלך. במצב זה לשחקן בעל המלך המאוים מותרים רק מהלכים המסירים את האיום.

**מט** – **מהלך** הגורם ל**שח** ובלתי אפשרי להסירו לאחר **מהלך** אחד.

**תיקו** – כאשר אין **שח** ואין **מהלכים** חוקיים לאחד מהשחקנים או כאשר אין כלים על הלוח מלבד שני המלכים.

**מזלג** – שני **איומים** או יותר המתבצעים ע"י כלי אחד.

**הצמדה – איום** על **חוסם** שכאשר יזוז יחשף **חוסם** נוסף.

**תמיכה** – כאשר כלי יכול להגיע במהלך אחד למיקום של כלי אחר מאותו הצבע.

**פיתוח** – מהלך הגורם לכלי לעלות את **דרגת ההשקפה** שלו.

**דרגת השקפה** – מספר **המהלכים** שכלי יכול לבצע ברגע נתון

**החלפה** – מהלך או יותר הנגמר בהורדת סכום משקל כלים שווה בין שני השחקנים.

# 

# 5. תיאור הבעיה האלגוריתמית

**Model**

* ייצוג הלוח
* ביצוע מהלך
* בדיקה האם מהלך חוקי
* זיהוי קידום
* ביצוע קידום
* זיהוי מצב התחלה
* זיהוי שח
* זיהוי מט
* זיהוי תיקו
* זיהוי מצב הגנה
* זיהוי מצב התקפה
* זיהוי מצב פיתוח
* זיהוי מצב מזלג
* זיהוי מצב הצמדה
* זיהוי איום
* מיפוי הלוח למצבים ותיעדופם
* תגובה למצב התקפה
* תגובה למצב הגנה
* תגובה למצב התקפה
* תגובה למצב פיתוח
* תגובה למצב מזלג
* תגובה למצב הצמדה
* תגובה למצב איום
* אתחול הלוח לפי fen
* אתחול הלוח לפי מצב התחלתי
* חישוב מהלך אופטימלי

**viewer**

* ייצוג הלוח
* קליטת מהלך מהשחקן
* ביצוע מהלך
* ביצוע קידום
* קליטת fen מהמשתמש
* בדיקה האם fen חוקי
* אתחול הלוח לפי fen

# **6. סקירת אלגוריתמים בתחום הבעיה**

ישנם מספר אלגוריתמים ושיטות לבניית מנוע שחמט. מנוע שחמט טוב מנצל את כל האלגוריתמים האלה ומגיע להחלטה הכי נכונה ככל האפשר. אינו משתמש באלגוריתמים האלה

יעילות

**אלגוריתם minimax:**

המטרה במשחק השחמט היא למצוא את המהלך היקרב את השחקן לניצחון ככל שניתן. באלגוריתם זה אנו בונים עץ אפשרויות המכיל את כל האפשריות של הלוח כאשר כל רמה בעץ היא חצי מהלך (מהלך של שחקן אחד). לאחר מכן מבצעים ניתוח של הלוח ובוחרים במהלך שהוביל ללוח בעל סיכוי הכי גבוה לניצחון בשבילנו בהנחה שהשחקן היריב יבחר במהלך הכי טוב לטובתו. ככל שגובה העץ (עומק החיפוש) יגדלו, האלגוריתם ימצא מהלכים טובים יותר או יוודא שהמהלך שבחר הוא מהלך טוב.

**Alpha beta pruning:**

אלפה בטא פרונינג הינו תוספת לאלגוריתם המינימקס שבא למטב את החיפוש כך שנבדוק פחות מהלכים עבור אותו עומק או עבור אותו מספר מהלכים נגיע לעומק גדול יותר. כדי לצמצם את האפשרויות האלגוריתם גוזם את תתי העץ שלא יובילו למהלך טוב ובכךממטב את החיפוש ומעלה את גובה העץ עבור אותה יכולת עיבוד.

**Algorithm Based-Heuristic:**

אלגוריתם זה מעריך את לוח השחמט ופולט לאיזה שחקן יש יתרון ובכמה. אלגוריתם זה הוא קריטי בשיטת הminimax כדי להכריע האם המהלך הוא הוסיף לנו יתרון או הוריד. אלגוריתם זה עושה זאת ע"י ספירת הכלים על הלוח כפול משקל של כל כלי. משקל של מלכה יהיה יותר ממשקל של חייל. בנוסף האלגוריתם מנתח את הפוזיציה של כל כלי ונותן להם ערך. למשל המלך יקבל ניקוד גבוה אם הוא יהיה בקצה של הלוח שם הוא יותר מוגן (אלא אם זה סוף משחק ואז המלך יקבל ניקוד גבוה כאשר הוא קרוב לאמצע הלוח), הפרש יקבל ניקוד גבוה כאשר הוא באמצע הלוח כי אז יש לו דרגת השקפה גבוהה יותר וכך הלאה. ככל שהאלגוריתם של הערכה יהיה יותר מדויק, אנו נוכל להסתפק בגובה נמוך יותר של עץ האפשרויות. ברמה תיאורטית אם הפונקציה מושלמת אנו נוכל לחפש בגובה עץ 1 ולא יותר כדי למצוא את המהלך הכי טוב.

**Monte Carlo Tree Search:**

בדומה לאלגוריתם מינימקס גם במונטי קרלו אנו בונים עץ אפשרויות אך טיפה שונה. במקום לבצע brute force לכל האפשרויות כדי למצוא את המהלך הטוב ביותר אנו נשתמש בהסתברויות. תחילה נבנה עץ אפשרויות בעומק 1, לאחר מכן עבור כל בן נבצע סימולציית משחק רנדומלי שלם ונוסיף את תוצאת המשחק לצומת. האלגוריתם יבחר איזה מהלכים באים לחקור ע"פ נוסחה מתמטית בין הערך שבכל צומת לבין מספר הפעמים שחקרנו אותו ואז נבצע את אותו תהליך עד מספר חזרות מסויים. בסוף האלגוריתם נבחר את המהלך שבעל שילוב של הערך הכי גבוה וכמות הסימולציות שהורצו עבור תת העץ של אותו מהלך (ככל שחקרנו צומת יותר והסתברות הניצחון שלו גבוהה אז הוא כנראה מהלך טוב).

יש לציין שאלגוריתם זה אינו זקוק לפונקציה המעריכה את הלוח והוא מתאים למשחקים שבהם יש קושי לכתוב פונקציה המעריכה את הלוח כמו שחמט.

**בסיס נתונים:**

שמירת מצבי לוח והמהלך הנכון ביותר למצב בבסיס נתונים, ושליפת המהלך מבסיס הנתונים לפי מצב הלוח במהלך המשחק.

כיוון שיש יותר מדי מצבי לוח, בלתי אפשרי לשמור את כולם. נהוג לשמור מצבי פתיחה ומצבי סוף. כאשר מתחילת המשחק ועד ל5 מהלכים, ומ5 כלים על הלוח עד לסיום המשחק בהתאמה.

יתרונות: בחירת המהלכים הכי נכונים בדיוק של 100% בסיבוכיות לינארית התלויה בגודל בסיס הנתונים.

חסרונות: אלגוריתם זה פותר רק את מצב הפתיחה ומצב הסוף.

**מכונת מצבים:**

בשיטה זו אנו ננתח את משחק השחמט, נמצא אסטרטגיות שמעלות את הסיכוי לניצחון ונבחר את האחת המתאימה לפי המצב בלוח, ישנו דמיון בין שיטה זו לבין האופן שבו שחמטאי בוחר מהלך, הרי שחמטאי אינו מחשב בראש עץ אפשרויות ואין לו מבנה נתונים בגודל של נאסא בגישה של o(1). הוא מזהה דפוסים על הלוח ולפיכך מגיע להכרעה.

כדי לממש את השיטה הזו, יש לנתח את משחק השחמט ולמצוא דפוסי משחק והתגובה המתאימה אליהם.

דפוסים בשחמט לדוגמה: מזלג, הצמדה, שח, מט, תמיכה בכלי, פיתוח, החלפה, התקפה, הגנה.

# **7. מבנה נתונים**

**גרף:**

אפשר לייצג את הלוח ע"י גרף מכוון, כך שכל משבצת בלוח היא קודקוד, לכל קודקוד יהיו שכנים כך שכל שכן הוא נמצא במרחק של מהלך אחד ממנו לפי חוקי התזוזה של כל כלי. זאת אומרת שיהיה סוג לקשת, בין אם היא קשת של מלכה או קשת של חייל.

יעילות המבנה לצורך בדיקה האם מהלך חוקי:

בדיקה יכולת התזוזה: O(V) בממוצע עבור כלים מחליקים, וO(1) עבור כלי לא מחליק. בהנחה שבדיקה האם שני קודקודים שכנים היא בO(1).

בדיקת האם שח: כמות השכנים לקודקוד \* בדיקת יכולת התזוזה = O(V\*E).

בדיקת מהלך חוקי כוללת בדיקה האם הכלי יכול להגיע למשבצת היעד, האם אין אף כלי בדרך לשם אם זה כלי מחליק, והאם השחקן אינו גורם לאיום על המלך שלו.

**הגדרה בסביבת העבודה:**

typedef char Squares;

typedef struct {

Squares vertexs[8][8];

int rook\_edges[8][8];

int bishop\_edges[8][8];

int pawn\_edges[8][8];

int king\_edges[8][8];

int knight\_edges[8][8];

} Board;

**Bitboard:**

ייצוג הלוח ע"י 15 מספרים בעלי 64 סיביות. כאשר כל מספר מתאר את הלוח ‘מימד’ אחר של הלוח בעזרת 1 המייצג תפוסה, ו0 שמייצג אין תפוסה.

- חיילים לבנים/שחורים

- פרשים לבנים/שחורים

- רצים לבנים/שחורים

- צריחים לבנים/שחורים

- מלכות לבנות/שחורים

- מלך לבן/שחור

- כל הכלים של לבן/שחור

- כל הכלים על הלוח

בדיקת האם כלי יכול לזוז למשבצת מסוימת היא בO(1)

בדיקה האם שח - (מספר הכלים)O

בדיקת האם מט - (מספר המהלכים החוקיים \* בדיקה האם שח)O

**הגדרה בסביבת העבודה:**

typedef enum Player {

BLACK,

WHITE,

} Player;

enum BoardType {

WHITE\_PAWNS, WHITE\_KNIGHTS, WHITE\_BISHOPS,

WHITE\_ROOKS, WHITE\_QUEENS, WHITE\_KINGS,

BLACK\_PAWNS, BLACK\_KNIGHTS, BLACK\_BISHOPS,

BLACK\_ROOKS, BLACK\_QUEENS, BLACK\_KINGS,

WHITE\_ALL, BLACK\_ALL, ALL,

};

typedef struct {

u64 board[15];

Player current\_player;

} Bitboard;

# 8. תיאור האלגוריתם הנבחר

אני בחרתי במכונת מצבים כדי למצוא את המהלך האופטימלי עבור השחקן הממוחשב, אני בחרתי באלגוריתם זה כיוון שהוא אלגוריתם יעיל ומינימלי שביצועיו לא תלויים בחומרה שהוא רץ עלייה

נרכיב גרף מצבים כאשר כל קודקוד מכיל פונקציה של מהלך ומכיל תנאי שהלוח חייב לקיים כדי לעבור לקודקוד זה. בגרף זה הקשתות מכוונות אך לא ממושקלות.

מצב פתיחה

- בחר מהלך מספר הפתיחות

מצב שח

- אם השח הוא מ2 כלים ומעלה הזז את המלך אחרת הסר את האיום עם הכלי בעל החשיבות הנמוכה ביותר

מצב הגנה

- אם אני יכול להסיר את הכלי המאיים הסר אותו

- ברח במידה ואתה לא חושף כלי בעל חשיבות גבוהה יותר

- אם אני במזלג הסר את המזלג

- אם אני בהצמדה הסר את ההצמדה

- אם אני יכול לעשות מזלג לא כושל, בצע ותעבור למצב התקפה

- אם אני יכול לעשות הזמדה לא כושלת, בצע ותעבור למצב התקפה

מצב התקפה

- בצע מט

- בצע שח

- תאכל כלי עם ערך גדול יותר

- בצע מזלג לא כושל

- אם כלי שלי מאוים עם כלי בעל ערך קטן יותר עבור למצב הגנה

מצב פיתוח

- אם אני בשח עבור למצב שח

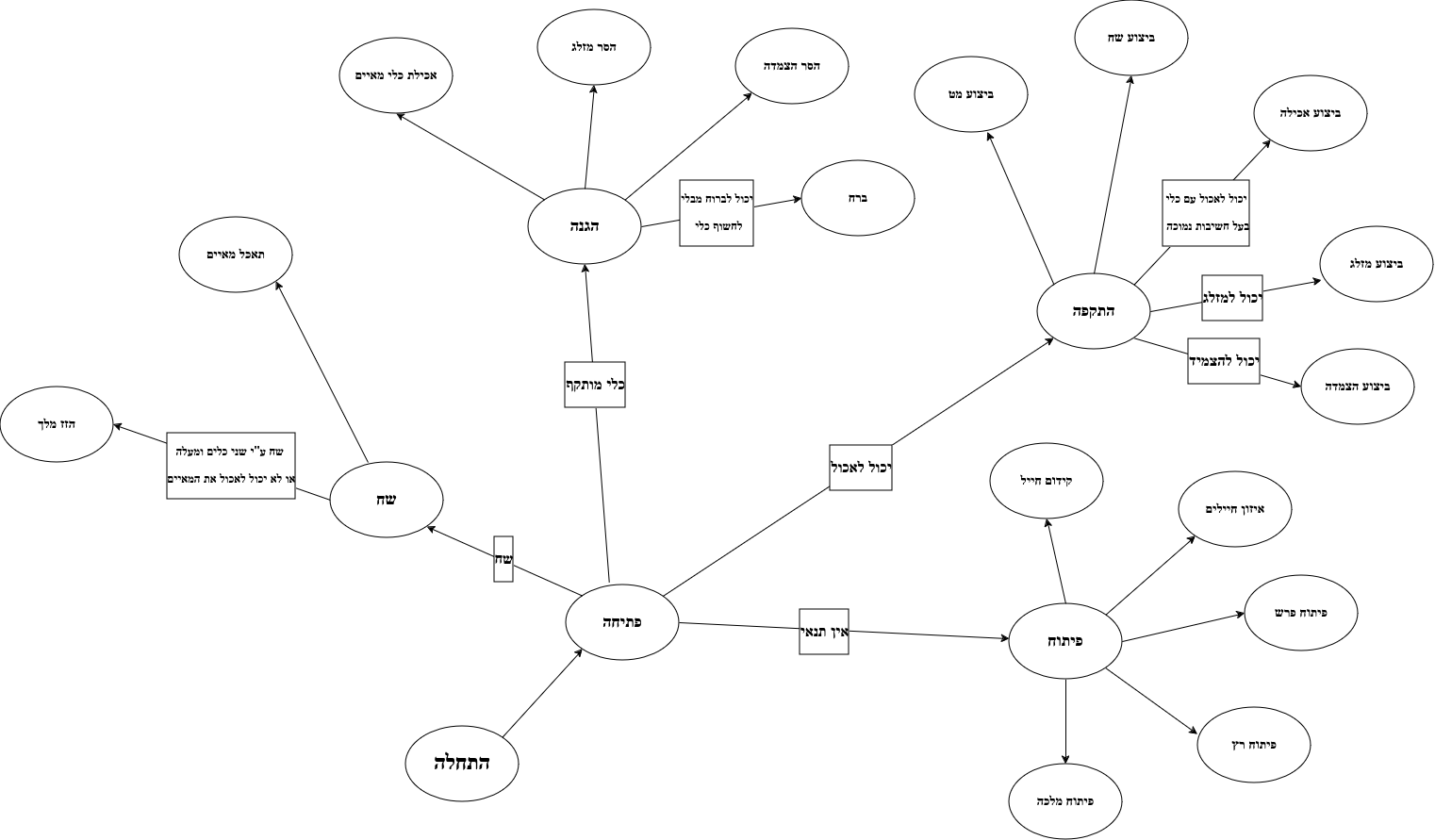
- אם אני יכול לבצע שח עבור למצב התקפה

- אם אני יכול לאכול כלי בעל ערך גדול יותר עבור למצב התקפה

- אם כלי שלי מאויים ע"י כלי בעל חשיבות נמוכה עבור למצב הגנה

- פיתוח כלי

נארגן את קבוצת המצבים לגרף ונקבל:



כעת כדי למצוא את המהלך האידיאלי לפי מכונת המצבים הנתונה, האלגוריתם יבצע DFS ואם יגיע לעלה הוא יבצע את תוכנו של הקודקוד.

# 9. האלגוריתם הראשי בפסואוד קוד

Function DFS(Graph G, board)

{

bool done = false

create stack s

create visited array

push start node to s

while s not empty and not done {

node = pop from s

if visited[node] == false {

visited[node] = true

if node is leaf {

call node.action

done = true

}

for v in node.adjucents {

if visited[v] == false && v.condition(board) == true {

push v to s

}

}

}

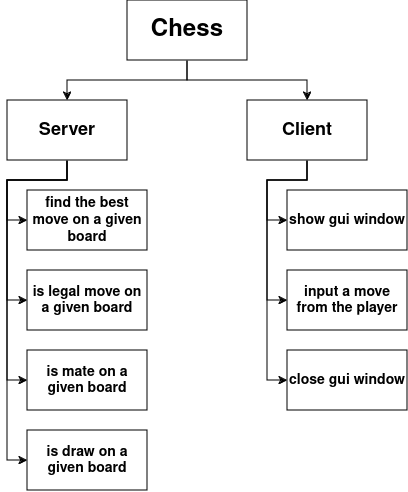
}

}

# 10. ארכיטקטורה של הפתרון המוצע בפורמט Top-Down Level Design

במצב אידיאלי פרויקט זה ימומש ע"י ארכיטקטורה שרת-לקוח, אך פרויקט זה הינו אבטיפוס ולכן הוא מומש בארכיטקטורה standalone.

להסיר חיצים



# **11. תרשים מקרי שימוש UML Use cases**

# 12. תרשים סביבת העבודה ושפת התכנות

**שפת תכנות:** c

**סביבת עבודה:** gcc

# 13. התכנית הראשית פסאודו קוד

Bitboard bitboard

Move move

Bool draw = false

Bool mate = false

GuiInitWindow(fen)

BitboardInit(bitboard, fen)

mate = BitboardIsMated(&bitboard, bitboard.current\_player)

draw = BitboardIsDraw(&bitboard)

while (!WindowShouldClose() && !draw && !mate) {

GuiDrawWindow()

if (GuiGetUserMove(move)) {

if (BitboardIsValidMoveAlgebraicNotation(bitboard, move, bitboard.current\_player)) {

BitboardMakeMoveAlgebraicNotation(bitboard, move)

GuiMakeMove(move)

mate = BitboardIsMated(bitboard, bitboard.current\_player)

draw = BitboardIsDraw(bitboard)

}

}

}

GuiCloseWindow();

# 14. תיאור ממשק גרפי

ספריית raylib הינה ספריית גרפיקה פשוטה הכתובה בc.

השתמשתי בספרייה לצורך:

* הצגת החלון
* ניגון שמע של אכילה ותזוזה
* הצגת הלוח
* הצגת הכלים
* קליטת מהלך מהמשתמש

החלון מורכב ממטריצה של ריבועים כאשר כל משבצת יכולה להכיל כלי

# 

# 15. תיאור הפונקציות והמחלקות הראשיות בפרויקט

# 