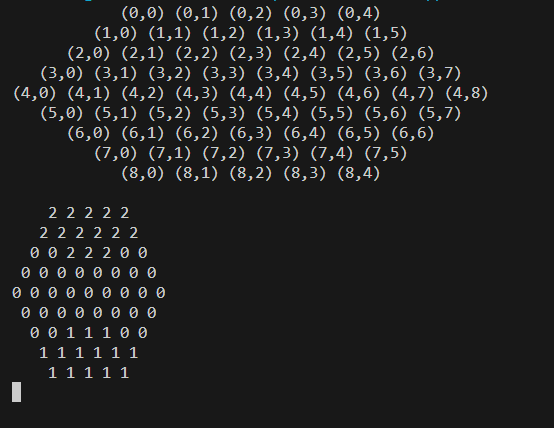
***אבלון***

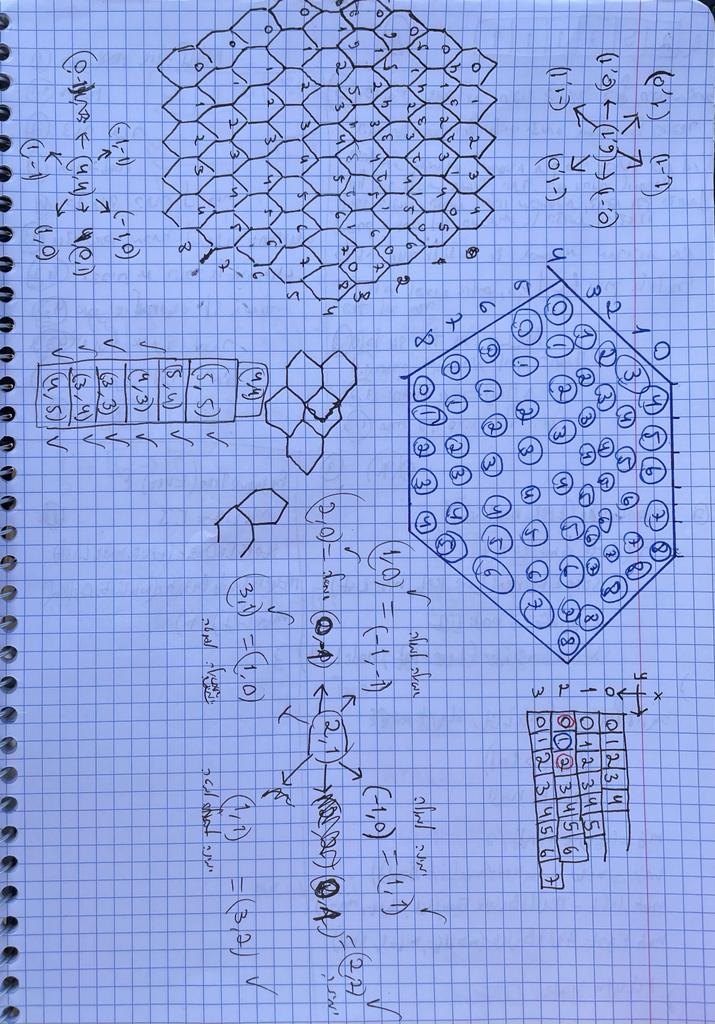
שם התלמיד: דניאל ליוש.

שם המנחה: אורי ואלון.

תאריך הגשה: 21/1/2024

הבעיה המרכזית שנתקלתי בה היא ייצוג לוח המשחק בגלל הצורה המיוחדת שלו, לוח בצורת משושה בשונה מלוחות בצורת ריבוע ומלבן, אי אפשר לייצג סתם ככה בעזרת מערך דו ממדי פשוט, מכמה סיבות, דבר ראשון, מספר התאים בכל שורה שונה, והתאים לא יושבים אחד מעל השני בצורה מסודרת כמו בלוחות מלבניים כך שתנועה על לוח שכזה היא מסובכת מאוד.   
אחרי לא מעט ימים של ניסיונות כושלים של כמה רעיונות הגעתי לפתרון שלי, רשימת סמיכויות, כל cell בלוח הוא איבר ברשימה הזו וכל איבר ברשימה מכיל עוד רשימה של הcell השכנים של התא המקורי, וכל cell מחזיק גם את רשימה של התאים השכנים שלו, הייתרון של הפתרון הזה הוא שניתן להגיע לתאים השכנים בקלות וכך לבצע צעדים וחשובים בצורה ממקודת. כל אובייקט מסוג cell מכיל את הקורדינציות שלו x ו y כך שזה כאילו מיוצג במערך דו ממדי, איקס מסמן את מספר השורה בין 1-9 ו y לעמודה. בנוסף כל cell מכיל משתנה של המצב שלו, האם הוא ריק – 0 שחקן אחד – 1 שחקן שתיים – 2. וכמו שאמרתי קודם גם רשימה של הcell השכנים שלו.   
כך נראה הלוח במצב ההתחלתי:   
  
בסוף החלטתי להשתמש במפה במקום ברשימה, כי ככה אני יכול לגשת לתא ביעילות ובקלות, אני משתמש במפה שהמפח שלה זה cell והערך של כל cell זה מפה של השכנים שלו כשהמפתח לכל הוא שוב פעם cell אבל הפעם הערך הוא Direction של השכן ביחס לתא המקורי, לכל תא שהוא לא בקצה יש ששה שכנים, שמאלה, ימינה, למעלה ימינה ושמאלה, ולמטה ימינה ושמאלה.   
זה מבנה הנתונים לייצוג הלוח, זה מדמה גרף לא מכוון מעגלי.  
  
השתמשתי ברשימה של cell במחלקה של הai שתחזיק את הגולות של המחשב, בחרתי ברשימה כי זה הדרך הכי קלה לעבור על כל איבר בה ולחשב דברים.   
  
השתמשתי בArrayList במחלקה של הAI כדי לשמור את כל המהלכים האפשריים ששחקן יכול לעשות, ואז לחשב את הציון של כל מהלך כדי לקבוע את המהלך הכי טוב.  
  
השתמשתי במחסנית שמחזיקה את שני המהלכים האחרונים כדי שיהיה אפשר לחזור אחורה   
  
השתמשתי בכל מיני מקומות בהאש מאפ וסט, אחד מהשימושים לדוגמא, שמירת השכנים של כל תא לפי הכיוון.  
  
מערך דו ממדי, להגדרת ששת הכיוונים האפשריים על הלוח.

סרטוטים:

  
  
A graph paper with writing on it

Description automatically generated

***Top Down Levels Diagram***

תיאור מילולי:  
לוח: היחידה הזו תבנה את מבנה הנתונים שמייצג את הלוח. כל תא בלוח הוא אובייקט מסוג cell   
  
מהלך: היחידה הזו מקבלת את של תאים שאמורים להכיל גולות של שחקן, ותא יעד שאמור להיות או ריק או עם גולה של היריב. היחידה תבדוק האם המהלך חוקי, אם כן ייקבע את סוג המהלך.   
  
לוגיקה: היחידה הזו תנהל את המשחק, את חילוף התורים בין השחקנים, ביצוע המהלכים, היא גם תשלוט על ממשק המשתמש, איתחול משחק, בדיקת ניצחון.

AI: היחידה תחשב את המהלך האופטימלי על ידי מכונת הכרעת מצבים, היחידה תקבע את מהלך המחשב ותשלח ללוגיקה.

**תיאור מערכת התוכנה**

* **GameBoard** – מחלקה שתפקידה הוא ליצור ולהחזיק את מבנה הנתונים של לוח המשחק.

|  |  |
| --- | --- |
| כותרת הפונקציה | הסבר |
| **initializeBoard()** | הפעולה יוצרת את לוח המשחק על ידי יצירת תאים וקביעת הסמיכויות שלהם. |
| **getCellAt(int x, int y)** | מחזירה את אובייקט התא לפי הקואורדינטות |
| **isValidPosition(int x, int y)** | פעולה שמקבלת קואורדינטות וקובעת אם זה תא שקיים בלוח או לא |
| **determineInitialState(int x, int y)** | מקבל קואורדינטות ומחזיר את המצב ההתחלתי שהתא הזה צריך להיות, שחקן אחד שתיים או ריק |
| **getCells()** | מחזיר אוסף של כל התאים בלוח |
| **getBoard()** | מחזיר את מפת הסמיכויות של הלוח |

* **Cell** – מחלקה שמייצגת תא בלוח

|  |  |
| --- | --- |
| כותרת הפונקציה | הסבר |
| **getNeighborsMap()** | מחזיר מפה של השכנים |
| **setState(int State)** | מגדיר את מצב התא,ריק 0 , שחקן 1, שחקן 2 |
| **isBorder()** | האם התא בקצה הלוח |
| **getState()** | מחזיר את מצב התא |
| **getX() / getY()** | מחזיר את ערך הX או הy של התא |

* **Move** – מחלקה שמקבלת רשימה של תאים ותא יעד ושחקן, ובודקת אם המהלך חוקי, את סוג המהלך, ביצוע המהלך וביטול מהלך ועוד.

|  |  |
| --- | --- |
| כותרת הפונקציה | הסבר |
| **isValid()** | הפעולה מחזירה אמת אם המהלך חוקי אחרת שקר |
| **executeMove()** | הפעולה מבצעת את המהלך אם הוא חוקי |
| **undoMove()** | הפעולה מחזירה את הלוח למצב שלפני ביצוע הפעולה |
| **getMarbles()** | מחזיר רשימה של הגולות שהיו חלק מהמהלך |
| **determineMoveType(List<Cell> selectedMarbles, Cell dest)** | קובע את סוג המהלך לפי הגולות שנבחרו והיעד |

* **GUI -** המחלקה היא הקונטרולר של ממשק המשתמש והיא מנהלת את המשחק גם.

|  |  |
| --- | --- |
| כותרת הפונקציה | הסבר |
| **initialize()** | מאתחל את רכיבי הGUI ומקשר אותם למודל הנתונים |
| **updateBoard()** | מעדכן את הGUI |
| **turn(Cell cell)** | קולט מהמשתמש לחיצות עכבר ומבצע בהתאם ללחיצה |
| **undoMove()** | מבצע undo לתור של המחשב ושל השחקן ומעדכן את הGUI |
| **executeTheTurn(Move move)** | מקבל מהלך מהשחקן או מהמחשב ומבצע אותו |
| **computerPlay()** | תור של המחשב |
| **endGame()** | פעולה שמקפיצה על המסך חלון של סוף משחק עם הודעה על מי ניצח ושואל אם רוצה לשחק שוב או לצאת |
| **changePlayer()** | מחליף את התור ביו המחשב לשחקן האנושי ולהפך |
| **restartGame()** | מאתחל את הקונטרולר וכל הנתונים ומתחיל משחק חדש |

Computer – מחלקה שמקבלת מצב לוח ומחשבת את המהלך האופטמילי עבור השחקן המבוקש מבלי לחשב כמה מהלכים קדימה, בעזרת מכונת הכרעת מצבים.

|  |  |
| --- | --- |
| כותרת הפונקציה | הסבר |
| **cellsToMoveTo()** | מעדכן את הרשימה של התאים אליהם כל שחקן יכול לזוז |
| **updateMarblesList()** | מעדכן את הרשימה של הגולות של כל שחקן |
| **getAllPotentialMovesForBoth()** | מעדכן את הרשימה של כל המהלכים החוקיים של כל שחקן |
| **computerTurn()** | מחזיר את המהלך האופטימלי לשחקן |
| **evaluatesBoardState()** | מחשב על ידי מכונת הכרעת מצבים את הציון של כל מהלך |