**פרוייקט גמר**

שפת פרולוג והיבטים לבינה מלאכותית – 20596

**ארבע בשורה עם מודל אלפא-בטא**

מגיש: עמרי

תאריך: אוקטובר 2020

**תוכן עניינים**

הרצה ..................................................................................................... עמ' 3

חוקי המשחק .......................................................................................... עמ' 3

מבנה נתונים ........................................................................................... עמ' 4

אלגוריתמים ושיטות מרכזיות .................................................................. עמ' 5

צילומי מסך ............................................................................................ עמ' 9

**הרצה**

יש להריץ את המשחק ב-Prolog SWI, באמצעות הפקודה:

play.

המשחק פותח והורץ על SWI Prolog על המחשב נייח עם זיכרון של 24GB ומעבד Intel i5-4570 3.20GHz.

**חוקי** **המשחק**

המשחק הוא המשחק הקלאסי ארבע בשורה, בו על כל שחקן להגיע לרצף של ארבעת כלי המשחק שלו באמצעות הרכבתם בשורה, בעמודה באלכסון שמאלי או אלכסון ימיני בלוח המשחק.

הערות כלליות לגבי פיתוח המשחק:

* למשחק 6 רמות קושי אשר קובעות את רמת העומק של אלגוריתם עץ החיפוש.
* המשחק פותח עבור לוח כללי של NxN, ונבדק בעיקר על לוח סטנדרטי של 7x7.
* המשחק לא מצפה לקלט תקין, ואורך בדיקה על הקלט שנכנס ע"י המשתמש ונותן לו אפשרות לתקן את הקלט עם הודעת שגיאה מתאימה.
* בכל תור, הן של השחקן והן של המחשב, הלוח העדכני מודפס.

**מבנה נתונים**

מבנה הנתונים בו השתמשתי על מנת לייצג את לוח המשחק הוא רשימה המכילה בתוכה רשימות נוספות אשר כל אחת מהן מייצגת שורה בלוח המשחק. גודל הרשימה דהיינו, גודל לוח המשחק מתקבל על ידי המשתמש כאשר ההגבלה היחידה היא ליצור לוח בגודל של 4x4 לפחות. הלוח עצמו מיוצג ע"י המספרים הבאים:

* תא ריק
* התא מכיל כלי משחק של השחקן.
* -1 התא מכיל כלי משחק של המחשב.

בעת הדפסת הלוח, כלי משחק של המחשב מוחלף ב-x, וכלי המשחק של השחקן מוחלף ב-o.

**אלגוריתמים ושיטות מרכזיות**

השיטה turn:

שיטה זו אחראית לניהול התור של השחקן או של המחשב.

בעת ניהול תור השחקן, השיטה אחראית לקבל את העמודה אליה יוכנס כלי המשחק של השחקן וכמובן בדיקת תקינות הקלט דהיינו, בדיקה האם הקלט הוא מספר, העמודה שהתקבלה קיימת בלוח ואינה מלאה. אם כל הבדיקות תקינות, הכלי יוכנס לעמודה הנתונה, הלוח יודפס ותתבצע בדיקה האם במהלך האחרון הוביל לניצחון. אם כך הדבר, המשחק יסתיים, אחרת יועבר התור למחשב.

בעת ניהול תור המחשב, תורץ השיטה alphabeta שאחראית לחישוב התור הבא של המחשב. לאחר שהחושב מהלך, תתבצע בדיקה אם הוא הוביל לניצחון. אם כך הדבר, המשחק יסתיים, אחרת יועבר התור לשחקן.

בשני המקרים להלן, השיטה תבדוק האם הלוח כבר מלא לגמרי בכלי משחק ותכריז על תיקו בהתאם.

השיטה isFull:

השיטה בודקת האם הלוח מלא, זאת ע"י בדיקה האם 0 המייצג תא ריק מופיע בשורה הראשונה של הלוח, זוהי גם השורה הראשונה מבין כל הרשימות המייצגות את הלוח.

השיטה isColNotFull:

השיטה בודקת האם מספר העמודה שהתקבל מייצג עמודה לא מלאה ע"י בדיקה של האיבר המתאים בשורה הראשונה אשר צריך להיות 0.

השיטה putInCol:

שיטה תכניס לעמודה המתאימה של כלי המשחק שהתקבל ותחזיר את הלוח המעודכן. ראשית, השיטה תהפוך את סדר הרשימות המייצגות את הלוח כך שהשורה התחתונה ביותר תהיה ראשונה בסדר. לאחר מכן, השיטה משתמש בשיטת העזר putInFirstEmptyCell שעוברת על כל הרשימות עד שמוצאת את הרשימה הראשונה בה העמודה הנתונה היא 0, כלומר מייצגת תא ריק ותכניס אליו את כלי המשחק בנתון. לאחר מכן שיטת העזר תחזיר את הלוח המעודכן לשיטה אשר זו בתורה תהפוך שוב את סדר הרשימות בלוח לסדרן המקורי ותחזיר לוח זה.

השיטה isItAWin:

שיטה זו, בעזרת שיטות עזר, בודקת האם ישנו ניצחון במשחק בהתאם לכלי המשחק שהתקבל ועליו מתבצעת הבדיקה. ישנה שיטת עזר לכל בדיקה שצריך לעשות, כלומר שיטת העזר הבודקת ניצחון בשורה, שיטת עזר הבודקת ניצחון בעמודה ושתי שיטות עזר הבודקת ניצחון באלכסון ראשי ומשני. שיטות העזר משתמש בשיטה append כדי לבדוק האם תנאי הניצחון מתקיימים:

* ניצחון בשורה – שיטת ה-append הראשונה רצה על הלוח ועל כל שורה בודקת האם בשורה זו יש 4 כלי משחק זהים ברצף.
* ניצחון בעמודה – שיטת ה- append הראשונה מחפשת 4 שורות אשר מקיימות ( ב-4 ה-appends תחתיו) שהמרחק בין כלי המשחק הנתון לתחילת השורה שווה בין ארבעתן.
* ניצחון באלכסון – באופן דומה לניצחון בעמודה, גם בניצחון באלכסון בודקים את המרחק מתחילת השורה לכלי המשחק הנתון, רק שהפעם, בהתאם לכיוון האכלסון, המרחק צריך להיות קטן או גדול בין כל כלי משחק.

השיטה alphabeta:

שיטה זו היא השיטה אשר התקבלה לידינו מאת המנחה רועי רחמני ולא בוצעו בה שינויים מלבד הפונקציה היוריסטית בה היא משתמשת והמהלכים המומלצים שהיא מקבלת, אותם הציג בהמשך. על כן, לא אפרט כאן את אופן פעולת השיטה.

השיטה move:

שיטה זו, בהינתן לוח משחק תחזיר לוח מעודכן עם מהלך של המחשב, ולוח נוסף עם המהלך זהה רק עם כלי משחק של השחקן. שיטה זו תחזיר כמובן מהלכים תקינים, כלומר כלי המשחק נכנסו לעמודות לא מלאות.

השיטה recommended\_moves:

שיטה זו תחזיר את כל המהלכים המומלצים ללוח הנתון באופן הבא:

* השיטה משתמשת בשיטה move ובודקת האם קיים מהלך מבין כל המהלכים השונים במרחק צעד אחד מהלוח הנתון אשר מובילים לניצחון של המחשב. אם קיים מהלך כזה, זה המהלך היחיד שיוחזר ע"י השיטה. זאת כדי לוודא שזה יהיה המהלך שיבחר ואין טעם להמשיך לחפש מהלכים נוספים.
* אם אין מהלך שמוביל לניצחון המחשב, השיטה תשתמש בשיטה move ותבדוק האם קיים מהלך מבין כל המהלכים השונים במרחק צעד אחד מהלוח הנתון אשר מובילים לניצחון של השחקן. אם קיים מהלך כזה, נתעלם מהמהלכים האחרים ונחזיר את המהלך שימנע (יחסום) את המהלך של השחקן ע"י הצבת כלי משחק של המחשב. לצורך חישוב כל המקרים שלהלן השיטה ראשית, הופכת את הלוח כך שכל 1 יהפוך ל- -1 ולהפך. לאחר מכן תבדוק האם המהלך שהתקבל מוביל לניצחון של -1 שהוא בעצם ניצחון של 1 כי הפכנו את הלוח. אם אכן יש ניצחון, ניקח את הלוח השני שהשיטה move החזירה אשר מייצג לוח שבו הוכנס 1 במקם -1, כיוון שמדובר בלוח הפוך המהלך זה מייצג בעצם את המחשב בלוח המקורי. על כן, ניקח לוח זה ונפעיל עליו היפוך נוסף ואותו השיטה תחזיר.
* אם שני המקרים לעיל לא מתקיימים נחזיר את כל מהלכים האפשריים במרחק של צעד אחד מהלוח הנתון.

הפונקציה היוריסטית - השיטה value:

תפקיד שיטה זו הוא להעריך את הלוח שהתקבל באמצעות מספר. השיטה פועלת באופן הבא:

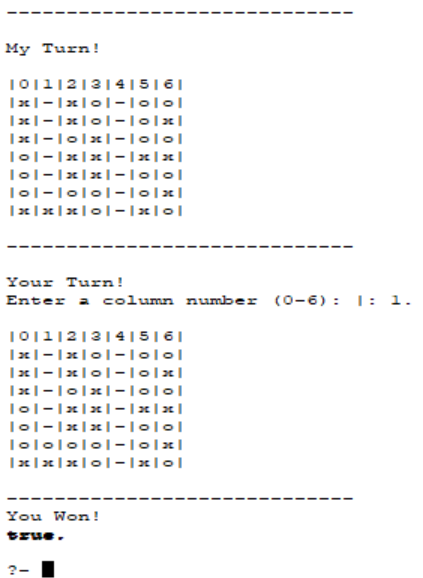
* ראשית נבדוק האם הלוח מוביל לניצחון של המחשב. אם כך הדבר תחזיר את המספר 1000 בחיבור עומק הלוח הנוכחי. כיוון שכל ריצה של alphabeta הערך של depth יורד, ערך גבוה של depth מתאר שלב מוקדם יותר בעץ, כלומר מהלך קרוב יותר למהלך הנוכחי. לכן נעדיף ניצחון קרוב מאשר ניצחון רחוק. בצורה זו נקבל ערך גדול יותר עבור ניצחון קרוב.
* במקרה ואין ניצחון למחשב, נבדוק האם יש ניצחון לשחקן בלוח. אם כך הדבר, נחזיר -1000 ונחסר ממנו את העומק של הלוח הנוכחי. בדומה למה שהוסבר בתת הסעיף הקודם, כאן נעדיף הפסד רחוק יותר.
* אם שני המקרים לעיל לא מתקיימים, נחזיר ערך רנדומלי בין 0 ל- 100 עבור הלוח שהתקבל בכדי לערבב מהלכים, כך שכל משחק יהיה שונה.

**צילומי מסך**

תחילת המשחק:



ניצחון השחקן במשחק:



הפסד השחקן במשחק:

