**Othello**

**Avyal Ron & Dor Zohar**

**מבנה נתונים:**

Board:

מבנה הנתונים שבחרנו להשתמש ללוח הוא רשימה של רשימות, אשר כל רשימה ברשימות מייצג שורה בלוח, וכל תא בשורה מייצג את תוכן התא שנמצא. לדוגמא, בתא הBoard[3][2] יהיה הערך (black,white or empty) של התא של שורה 3 עמודה 2.

Position:

בשיטת האלפא-בטא ניתן למצוא את מבנה הנתונים Position שהוא המבנה של p(Board,Player).

מבנה נתונים זה מאחסן את מצב הלוח, ותור מי עכשיו (Player=black/white).

Move:

צעד יראה במבנה הנתונים c(X,Y), שמה שהוא מייצג זה את התא שבו נרצה לשים את המטבע הבא.

**אלפא בטא:**

אלפא בטא שלנו מקבל את Position, אשר הוא מבנה המתואר למעלה, וכמו כן אלפא ובטא יהיו בטווח 100 למינוס 100.

הוא מחזיר את הצעד שאותו המחשב יבצע:

נבדוק אם העומק שאליו הגענו הוא 0, במידה וכן, נעבור ל"עלי עץ ההחלטה" שזאת פונקצית value, שמבצעת את פונקצית היוריסטיקה שלנו בהינתן מצב הלוח.

במידה והעומק גדול מ0, נקבל את כל המהלכים האפשריים עבור הפוזיציה (Position) שלנו, ועבור מהלכים אלה (קיים לפחות אחד כאשר אנחנו נכנסים לאלפא בטא בהתחלה, זאת מכיוון שבפרדיקט choose\_move אשר קורא לאלפא-בטא אנחנו בודקים האם קיימים צעדים אפשריים), נקרא לbestmove שיחזיר לנו את המהלך הרצוי.

Bestmove: פרדיקט זה מקבל מהלכים אפשריים, ועבור כל מהלך מבצע אותו על הפוזיציה שלנו, משנה את השחקנים ומבצע אלפא-בטא על הפוזיציה החדשה.

לאחר מכן הוא מבצע גזימה (cutoff).

ומחזיר לבסוף את הערך של אלפא ובטא.

Cutoff: מקבל ערך ובודק אם הוא גדול מאלפא, במידה ולא, ימשיך לשאר העץ שכן אין גזימה, במידה ולא הוא יבדוק אם קטן מבטא, במידה ולא, הרי שיש גזימה ויחזיר את הערך. אחרת, נקרא לbesmove ונמשיך בחיפוש הערך של תת עץ המהלכים.

**פונקצית היורסטיקה:**

לאחר מחקר של קריאת מאמרים שונים, הרבה ניסוי ותהייה, בחרנו בפונקציית היוריסטיקה הבאה (הפרדיקט value):

נסכום את הערכים הבאים:

1. **CoinsVal :**

זהו הערך של הפרדיקט coinValue, שמה שהוא מבצע זה הפרש בין מספר המטבעות של השחקן שיש בפוזיציה לעומת היריב שלו, נחלק את ההפרש בסכום של כל המטבעות על הלוח, זאת כדי לקבל את היחס של השחקן לעומת היריב (לדוגמא האם יש לו פי שתיים מטבעות מהיריב, מצבו מצויין לעומת אם יש לו פחות מטבעות מהיריב). נכפיל את היחס שקיבלנו ב100 על מנת שנקבל מספר לאלפא-בטא

1. **CornersVal:**

זהו ערך הפרדיקט cornersValue,

פינות הם התאים החשובים ביותר בלוח, שאותם אי אפשר לאכול אך יכולים לאכול פוטנציאלית הכי הרבה, לכן מה השהפרדיקט עושה הוא סופר את מספר הפינות ששייכים לשחקן בהפרש פינות היריב, על מנת שנקבל יחס חשוב בהיוריסטיקה השלמה נכפיל את מספר הפינות ב25 (מספר זה נבחר מכיוון שמקסימום הפינות הוא כאמור 4, לכן מקיסמום המספר שנחזיר הוא 100, שזה המקסימום אצלנו באלפא-בטא

1. **MobilityVal:**

זהו הערך שמחזיר הפרדיקט mobilityValue.

הפרדיקט בודק כמה אנחנו "ניידים" על הלוח, כלומר, כמה מהלכים פוטנציאלים יש לנו לעשות. במצב שאין מהלכים לבצע, הרי שאנחנו במצב מאד רע לעומת אם יש לנו מבחר גדול של מהלכים, שהרי זה מראה על מצב מאד מבוסס על הלוח, ופוטנציאל גדול לאכילות.

לכן מה שנחזיר הוא היחס של המהלכים שלנו לעומת המהלכים של היריב. חישוב היחס מתבצע כמו שביצענו אותו בcoinValue.

1. **RiskySpotsValue:**

זהו ערך שמחזיר הפרדיקט riskySpotsValue.

תאים מסוכנים ששחקן רצוי מאד שלא יכנס אליהם אלו התאים שצמודים לפינות, וזאת מכיוון שליריב יהיה את האופציה להשתלט על הפינות במידה ויאכל מהזווית הנכונה, ופינות הסברנו כמה הם חשובות לנו.

לכן, הפרדיקט יחשב את מספר התאים המסוכנים שהיריב נמצא בהפרש שלנו (ולא הפוך, שכן אנחנו נרצה שהיריב יהיה בתאים אלו ולא אנחנו), ונכפיל ב25 מאותו הסבר שעשינו בפרדיקט cornersValue.

**שגיאות:**

* במידה והשחקן (אתה) מכניס ערכים שגויים, אותיות, או תא שאי אפשר לבצע אליו מהלך, התכנית תזהה, תזרוק שגיאה על קלט שגוי, ולאחר מכן אפשרות לבצע שוב את המהלך.
* הכנסה של עומק שגוי:

כאשר מפעילים את המשחק (לדוגמא: play(3)) אז עומק המשחק הוא שלוש. זהו כמובן העומק שאלפא-בטא נכנס בעץ ההחלטות, וככל שהעומק גדול יותר כך המשחק קשה יותר..

לעומק יש מגבלה (אנחנו בחרנו 6 מכיוון שבמחשבים אצלנו המשחק רץ מהר עד העומק הזה. לאחר מכן המשחק עובד לאט).

במידה והכנסנו לדוגמא play(8) יזרק שגיאה על עומק שגוי.