בס"ד תאריך הגשה: 22.12.22

תרגיל 2 – משחקים

Connect 4

הקדמה:

בתרגיל זה תבנו סוכנים עבור משחק "4 בשורה". לאורך הדרך, תממשו את האלגוריתמים minmax ו - Alpha-Beta Pruning. בנוסף, תלמדו ותממשו את Expectimax.

הורידו את הקוד מה classroom, אשר נמצא ב - ex2_connect4.rar.

קבצים לעריכה:

multiAgents.py – הקובץ בו תממשו את הסוכנים שלכם.

שימו לב, אין לשנות חתימות של הפונקציות שם כלל! עליכם להשלים את המימושים היכן שנדרש.

קבצים שכדאי להסתכל עליהם:

Connect4.py - הקובץ הראשי שמריץ את המשחק. קובץ זה מתאר את ה GameState, שבו נשתמש בפרויקט זה. וכן מכיל את לוגיקת המשחק.

gameUtil.py – קובץ זה מכיל משתנים קבועים למשחק. חלק מהמשתנים האלו (כמו גודל הלוח) נוכל לשנות במהלך הבדיקה לבדיקות שונות.

test.py – בקובץ זה תוכלו להשתמש על מנת לבדוק את עצמיכם. (הסבר בהמשך)

קבצי תמיכה שניתן להתעלם מהם:

graphics.py – גרפיקה.

util.py – פונקציות עזר, ניתן להתעלם מקובץ זה.

הגשה

במהלך התרגיל תערוכו חלקים של **multiAgents.py.** עליכם לשלוח קובץ זה בלבד עם הקוד והערות שלכם. נא לא לשנות את הקבצים האחרים או לשלוח אף אחד מהקבצים המקוריים מלבד קובץ זה. **בראש הקובץ נא לכתוב את שם הסטודנט ות.ז.**

בנוסף לקובץ זה, עליכם להגיש קובץ פרטים אישיים בשם detail.txt והוא יכיל את שם הסטודנט בשורה הראשונה ות.ז. בשורה השניה.

ברוך הבא ל Multi-Agent Connect 4 !

בקובץ connect4.py ישנם 4 פרמטרים בעזרתם ניתן לשחק במשחק, עליהם נסביר כעת:

```
# graphicMode - True- for graphic mode. False- for textual mode
# gameMode  - get the value 2- for player vs. player or 1- for palyer vs. AI_agent
# depth - the max depth to explore the minimax tree
# type - the name of the agent will play as AI_agent (one of "BestRandom", "MinimaxAgent", "AlphaBetaAgent", "ExpectimaxAgent")
```

graphicMode – משתנה בוליאני המאפשר לשחק את המשחק באמצעות גרפיקה יפה או דרך ה command
gameMode – משתנה זה יקבל אחד משני הערכים 2 – למשחק של שני שחקנים, 1 – למשחק של שחקן מול סוכן.
depth – משתנה זה יכיל את העומק המקסימלי בחקירת עץ המשחק
type – יכיל את שם הסוכן איתו תרצו לשחק במידה ונבחר game_mode = 2. סוגי הסוכנים השונים הם: "BestRandom", "MinimaxAgent", "AlphaBetaAgent", "ExpectimaxAgent"

בקובץ connect4.py כבר מוגדרים לכם ערכי ברירת מחדל באמצעותם אתם יכולים להתנסות במשחק. שינוי של ערכי המשתנים יפעילו את המשחק בדרכים השונות.

שימו לב, רק כאשר game_mode=1 תוכלו לשחק מול אחד הסוכנים.

שאלה 1 – Minimax (35 נקודות)

בשאלה זו תממשו שחקן ממוחשב. את המימוש יש לכתוב במחלקה המסופקת לכם MinmaxAgent. המחלקה נמצאת בקובץ multiAgents.py.

הקוד שלכם צריך להרחיב גם את עץ המשחק לעומק שרירותי, (משתנה depth) כשאת ניקוד העלים תחשב בעזרת הפו' self.evaluationFunction. מימוש ברירת המחדל מוגדר כ - scoreEvaluationFunction.
(הפונקציה שמחשבת את הציון בעלים כבר ממומשת במחלקה gameState)
שימו לב כי MinmaxAgent מרחיבה את MultiAgentSearchAgent ולכן ישנה גישה ל self.depth ול - self.evaluationFunction.
אנא ודאו שקוד ה Minmax שלכם מתייחס לשני המשתנים האלו במידת הצורך מכיוון שהערכים שלהם עשויים להשתנות בתגובה לאפשרויות ההרצה השונות.

טיפים:

- ממשו אלגוריתם רקורסיבי תוך שימוש בפו' עזר.
- ישום נכון של minmax יוביל לכך שהסוכן יפסיד בחלק מהמשחקים. זו לא בעיה מכיוון שזו התנהגות נכונה ותעבור את הבדיקות.
- פונקציית ההערכה לconnect4 בחלק זה כבר כתובה (self.evaluationFunction). אין לשנות את הפונקציה הזו.
- AI_agent הוא תמיד סוכן 1.
- כל המצבים במינימקס צריכים להגיע מ GameStates, להעביר אותן ל- getAction או להפיק אותן באמצעות GameState.generateSuccessor.

הוראות הרצה:

בכדי להתנסות בהרצת שחקן ה minmax, עליכם לשנות את המשתנה:

gameMode =1
type="MinimaxAgent"

ערך המשתנה depth חייב להיות לפחות 3.

שאלה 2 - Alpha-Beta Pruning (35 נקודות)

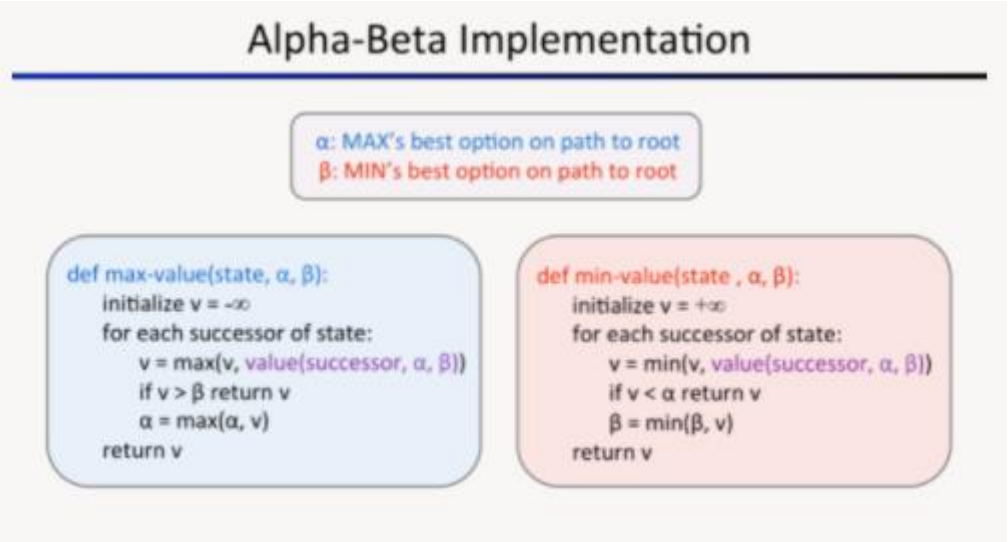
צורו סוכן חדש שמשתמש ב Alpha-Beta Pruning כדי לחקור ביעילות רבה יותר את עץ המינימקס, ב- AlphaBetaAgent. אתם אמורים לראות שיפור במהירות.

ערכי המינימקס ב AlphaBetaAgent צריכים להיות זהים לערכי המינימקס ב MinimaxAgent, אם כי הפעולות שהוא בוחר יכולות להשתנות בגלל התנהגות שונה של שבירת קשרים.

שימו לב: מכיוון שאנו בודקים את הקוד כדי לקבוע אם הוא בוחן את המספר הנכון של מצבים, חשוב שתבצע גיזום אלפא-בטא מבלי לסדר ילדים מחדש. במילים אחרות, המצבים ישארו בסדר שהוחזר על ידי GameState.getLegalActions.

שימו לב 2: לא לגזום במצבי שוויון! (בניגוד למה שנלמד בכיתה, שם כן גזמנו במצבי שיוויון)

קוד הפסאודו להלן מייצג את האלגוריתם שעליך ליישם עבור שאלה זו.



יישום נכון של Alpha-Beta Pruning יוביל לכך שהסוכן יפסיד בחלק מההרצות. זו לא בעיה, מכיוון שזו התנהגות נכונה והיא תעבור את הבדיקות.

הוראות הרצה:

בכדי להתנסות בהרצת שחקן ה minmax, עליכם לשנות את המשתנה:

gameMode =1
type="AlphaBetaAgent"

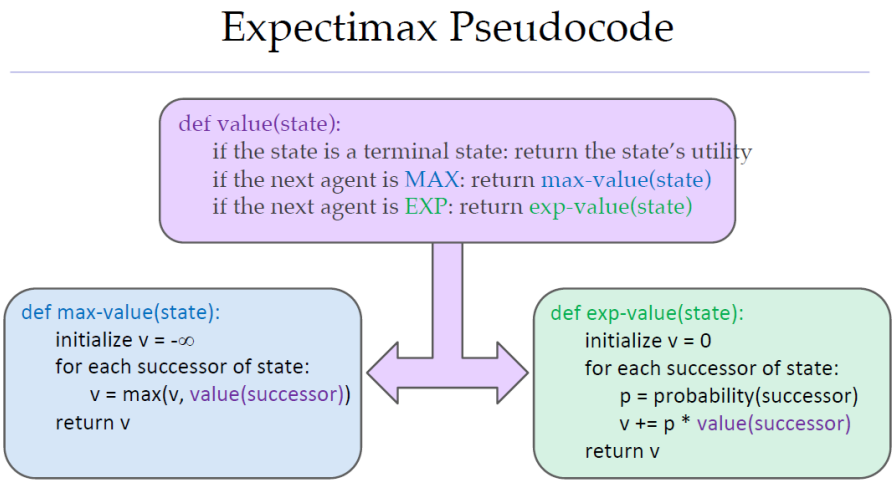
ערך המשתנה depth חייב להיות לפחות 3.

שאלה 3 – Expectimax (30 נקודות)

הסוכנים Minimax ו- alpha-beta נהדרים, אבל שניהם מניחים שאתה משחק נגד יריב שמקבל החלטות אופטימליות. כפי שכל מי שניצח אי פעם איקס-עיגול יכול לומר לך, זה לא תמיד המקרה. בשאלה זו תממשו את ExpectimaxAgent, שהוא שימושי ליצירת מודלים של התנהגות הסתברותית של סוכנים שעשויים לעשות בחירות לא אופטימליות. בהמשך הקורס נפגוש את האלגוריתם כשנרצה למדל חיפוש בסביבה לא ודאית.

חומרים נוספים על Expectimax ניתן לראות כאן, או בחיפוש ברשת.

קוד הפסאודו להלן מייצג את האלגוריתם שעליך ליישם עבור שאלה זו.



כדי לפשט את הקוד שלך, הנח שאתה משחק נגד יריב שבוחר מבין getLegalActions שלו באופן אחיד באקראי.

גם כאן, יישום נכון של Expectimax יוביל לכך שהסוכן יפסיד בחלק מההרצות. זו לא בעיה, מכיוון שזו התנהגות נכונה והיא תעבור את הבדיקות.

הוראות הרצה:

בכדי להתנסות בהרצת שחקן ה minmax, עליכם לשנות את המשתנה:

```
gameMode =1
type="ExpectimaxAgent"
```

ערך המשתנה depth חייב להיות לפחות 3.

בדיקות

לנוחיותכם, הוספנו קובץ test.py המאפשר לכם לבדוק את המימושים השונים.

מצורפת תיקיה בשם test_yourself בה נמצאים מספר קבצי בדיקה.

על מנת להריץ את הבדיקות, עליכם להריץ את הקובץ test.py. הבדיקה תחזיר לכם משוב על קבצי הבדיקה השונים.

בהצלחה רבה!!