תרגיל בית 2 מבוא לבינה מלאכותית

Ex2 Introduction to AI

Gobblet Gobblers

(enhanced TicTacToe)

הקדמה ואדמיניסטרציה

הנחיות כלליות

- תאריך הגשת התרגיל: **5.1.2023**
- את המטלה יש להגיש בזוגות בלבד – בקשות להגשה ביחידים באישור המתרגל האחראי בלבד (ספיר טובול).
- יש להגיש את המטלה מוקלדת בלבד – פתרון בכתב יד לא ייבדק.
- התשובות צריכות להיות כתובות בשפה העברית או באנגלית.
- אפשר לשלוח שאלות בנוגע לתרגיל דרך הפיאצה.
- המתרגלת האחראית על התרגיל: **אופק גוטליב**.
- בקשות דחיה מוצדקות יש לשלוח למתרגל האחראי בלבד.
- במהלך התרגיל ייתכן שיעלו עדכונים – הודעה תפורסם בהתאם במקרה זה.
- העדכונים מחייבים וזוהי אחריותכם להתעדכן לגביהם עד מועד הגשת התרגיל. עדכונים יופיעו בטופס בצבע צהוב.
- העתקות תטופלנה בחומרה.
- ציון המטלה כולל חלק יבש וחלק רטוב. בחלק היבש נבדוק שתשובתכם נכונה, מלאה, קריאה ומסודרת. בחלק הרטוב הקוד שלכם ייבדק בצורה מקיפה באמצעות בדיקות אוטומטיות – אשר ישוו את המימוש שלכם למימוש שלנו. חשוב לעקוב בזהירות אחר הוראות התרגיל מאחר שסטיות מהמימוש המבוקש עלולות להוביל לכשל בטסט האוטומטי (גם אם המימוש "נכון ברובו"). במהלך הבדיקה האוטומטית יינתן זמן סביר לכל הרצה – כך שכל עוד תעקבו אחר ההוראות, המימוש שלכם יעמוד בהגבלות הזמנים ויחזיר תוצאות טובות מספיק בשביל הטסט.
- מומלץ להסתכל בקוד בעצמכם. שאלות בסיסיות על פייתון שלא נוגעות לתרגיל כדאי לבדוק באינטרנט לפני שאתם שואלים בפיאצה. מומלץ לקרוא את הקוד הנתון על מנת להבין את אופן פעולתו – במקרה שישנם דברים לא מובנים . (לשם כך יש הערות רבות ואף הסבר מורחב על הסביבה!)
- מומלץ לא לדחות את התרגיל לרגע האחרון מאחר שהמימוש וכתיבת הדו"ח עלולים לקחת יותר זמן מהצפוי.
- התייחסות בלשון זכר, נקבה או רבים מתייחסים כלפי כלל המינים.
- **ציון התרגיל המקסימלי הינו 110 כיוון שיש בונוס - פירוט על הבונוס בסוף החלק הרטוב**

הוראות הגשה

בתוך קובץ זיפ עם השם : HW2_AI_id1_id2.zip

את הדו"ח היבש בפורמט הבא : id1_id2.pdf

ואת הקובץ: subbmision.py שבו אתם ממשים את האלגוריתמים

היכרות עם המשחק
Introduction with the game

לוח וכלי משחק:

לוח משחק של 3X3

2 שחקנים

לכל שחקן: 6 כלים

2 בגודל קטן

2 בגודל בינוני

2 בגודל גדול

המשחק בכללי:

גרסה משופרת של איקס עיגול, השחקן המנצח הוא זה שהצליח למלא שורה, טור או אלכסון בסימנים שלו (הגודל לא משנה).

שחקנים יכולים לזלול שחקנים אחרים ובכך לשנות את הצבע של משבצת קיימת.

להלן סרטון אשר עוזר להבין בכלליות את מהות המשחק:

How to Play Gobblet Gobblers

המשחק בספציפי:

- כל שחקן בתורו מניח גובלין על הלוח, או על משבצת ריקה או על גובלין קטן יותר
- במקום להניח גובלין חדש על הלוח יכול שחקן בתורו לבחור גובלין שלו שנמצא על הלוח ולהזיז אותו למשבצת חוקית
- גובלין יכול לזלול כל גובלין שקטן ממנו (גם אם הם בצבעים זהים)
- ניתן להניח 3 גובלינים אחד על השני (גדול על בינוני כאשר הבינוני על קטן)
- דגש חשוב: במשחק המקורי אין הגבלת צעדים ובגרסה שלנו אנו מגבילים את מספר התורות במשחק. (אם אין נצחון ונעבור את הגבלת הצעדים נחשיב זאת בתור תיקו)
- דגש חשוב: במשחק המקורי חשוב לזכור מה שיש מתחת לכל גובלין שעל הלוח, אצלנו יודעים תמיד (למחשב יש זכרון טוב 🙂)

להלן ספר החוקים הרישמי של המשחק: Gobblet gobblers rules

מה קיבלתם?

קיבלתם 4 קבצים מרכזיים:

1. Gobblet_Gobblers_Env.py סביבת המשחק אותו ישחקו הסוכנים שלכם - הסבר מפורט בהמשך
2. submission.py קובץ שבו תממשו את הסוכנים שלכם וזהו גם הקובץ אותו אתם מגישים
3. game.py מכיל שתי פונקציות שמטרתן להריץ את המשחק ויעזרו לכם לבדיקה עצמית - הסבר מפורט עליהן בהמשך
4. main.py משמש גם כן לבדיקה עצמית

היכרות עם הסביבה

הסביבה איתה תעבדו ממומשת בקובץ Gobblet_Gobblers_Env.py היא מבוססת על סביבה של gym כמו הסביבה איתה עבדתם בתרגיל בית 1.

היא מכילה את המתודות הבאות שקשורות לתפקוד הסביבה:

- **init()** - הקונסטרקטור של הסביבה (המחלקה). ניתן להשתמש בו באופן הבא:

```
env = GridWorldEnv()
```

- **reset()** - מאפסת את לוח המשחק. שמה את כל הכלים בצדדים ולוח המשחק ריק. משתמשים בה באופן הבא `state = env.reset()` הפונקציה מחזירה מצב (כדי להבין מה המשמעות של להחזיר מצב תקראו את הפירוט על הפונקציה get_state()), להלן הלוח במצב מאותחל:

```
        +------+------+------+
B1   B2 |      |      |      |  B1   B2
M1   M2 +------+------+------+  M1   M2
S1   S2 |      |      |      |  S1   S2
        +------+------+------+
        |      |      |      |
        +------+------+------+
```

*האותיות מייצגות את גודל השחקן (B - גדול, M - בינוני, S - קטן)

- **step()** - פונקציה שמקבלת action ומבצעת את הפעולה. הפעולה צריכה להיות tuple בפורמט (pawn,location) כאשר pawn מייצג את השחקן שתרצו להזיז מבין האופציות {B1,B2,M1,M2,S1,S2} ו-location את המקום על הלוח בו תרצו להניח את הכלי.
  location הוא מספר בין 0 ל - 8 שמייצג את המשבצת על הלוח באופן הבא:

```
+-----+-----+-----+
| 0   | 1   | 2   |
+-----+-----+-----+
| 3   | 4   | 5   |
+-----+-----+-----+
| 6   | 7   | 8   |
+-----+-----+-----+
```

דוגמא לשימוש בפונקציה

```
env.step(("B1", 7))
```

המשמעות: נזיז את גובלין B1 לאריח מספר 7.

- **render()** - מדפיסה את לוח המשחק דוגמא לשימוש בפונקציה

```
env.render()
```

  - שימו לב, כלים שנמצאים על כלים אחרים יסתירו את הכלים שמתחתיהם, כדי לדעת בדיוק איפה כל כלי ראו ()get_state.
  - שימו לב שיפתח לכם חלון pygame עם גרפיקה נוחה של המשחק ובו תוכלו לראות את המצב של הלוח לאחר שתבצעו (ליתר דיוק הסוכן שלכם יבצע 🙂) צעדים, בנוסף יש הדפסות לקונסולה שתוכלו להיעזר בהן בדיבוג.

* ישנן פונקציות פנימיות רבות נוספות, מוזמנים לקרוא את התיאור שלהם חלקן אפילו יכולות לעזור לכם בכתיבת היוריסטיקה בהמשך, אך אין צורך להתעמק בהן.

בנוסף הסביבה מכילה את הפונקציות הבאות שקשורות לאינטגרציה שלה עם הסוכנים שאתם הולכים לממש בתרגיל זה:

get_state() - פונקציה שמחזירה את המצב הנוכחי של הסביבה, משתמשים בה באופן הבא:

```
env.get_state()
```

והצורה שבה המצב של הסביבה מוחזר הוא במבנה של State (שמוגדר בקובץ Gobblet_Gobblers_Env.py).

get_neighbors() - פונקציה שמקבלת טיפוס מסוג state ומחזירה רשימה שמכילה את כל השכנים שלו (כל המצבים שיווצרו מכל הפעולות החוקיות שאפשר להפעיל על אותו מצב) ברשימה בעצם כל איבר הוא tuple של (action, state) כך תוכלו לעבור על המצבים ולהחזיר בקלות את הפעולה עבור המצב שתבחרו מרשימת השכנים.

is_final() - מבלבל אז חשוב לשים לב! פונקציה שמקבלת מצב מסוים ומחזירה :

None – עבור מצב לא סופי.

0 - אם יש תיקו, אם שני השחקנים ניצחו באותו צעד (מצב תקין) או אם עברו 100 תורות (50 לכל שחקן) ואין הכרעה.

1 - ניצחון של השחקן הראשון. שימו לב! במהלך המשחק, התור של השחקן הראשון מצוין בתור 0 ולא 1!

2 - ניצחון של השחקן השני.

- **play_game()** - פונקציה שמריצה משחק בודד. הפונקציה הנ"ל מקבלת מחרוזות שמתארות כל סוכן כפי שמתואר בתצלום של המילון בפונקציה הבאה. בנוסף יש דוגמא בהמשך. הפונקציה מחזירה מיהו המנצח ומדפיסה זאת למסך. מחזירה תוצאות כמו שמחזירה is_final().

```
"human": submission.human_agent,
"random": submission.random_agent,
"greedy": submission.greedy,
"greedy_improved": submission.greedy_improved,
"minimax": submission.rb_heuristic_min_max,
"alpha_beta": submission.alpha_beta,
"expectimax": submission.expectimax
```

- **play_tournament()** - פונקציה המריצה מספר play_games לפי ערך num_games שתתנו לה (אנו נבדוק את הקוד שלכם עם num_games = 50) ובכך מחזירה באחוזים כמה ניצחונות יש לכל שחקן וכמה תיקו היו במשחק. הפונקציה הנ"ל מקבל מחרוזות שמתארות כל סוכן כפי שמתואר בתצלום של המילון הנ"ל. דוגמא בהמשך.

* num_games הוא לא מספר המשחקים שרצים בפועל, בפועל רצים פי 2 משחקים. הוא מייצג כמה משחקים יש בהם כל שחקן הינו השחקן הראשון. (תורו לשחק ראשון)

מתחילים לכתוב!

חלק א - היכרות עם הסביבה (4 נק')

1. (יבש: 1 נק') כנסו לקובץ main והריצו את השורה שמסומנת בהערה ומעליה כתוב #PART1 השורה מריצה את המשחק עם שני סוכנים אנושיים, שחקו אחד נגד השני עד לניצחון וצרפו את ההדפסה של המצב הסופי.
2. (יבש: 2 נק') האם ניתן לגרום בתור של סוכן מסוים לביצוע פעולה שתגרום לסוכן האחר לנצח? אם כן הסבירו מדוע מצב שכזה יכול לקרות וצרפו את שני הצעדים האחרונים במשחק שכזה, אם לא, הסבירו מדוע לא יתכן מצב כזה?
3. (יבש: 1 נק') מהו המספר המקסימלי של אפשרויות לפעולות שונות (בהנחה שכל פעולה חוקית) שניתן לעשות בתור?

חלק ב - Improved Greedy

(15 נק')

בקובץ submission.py שקיבלתם *ממומש* עבורכם סוכן greedy *שמשתמש* בהיוריסטיקה לא חכמה שנקראת `dumb_heuristic2` היא מחשבת את כמות השחקנים הגלויים (לא נמצאים מתחת לשחקן אחר) שיש לסוכן שלה על הלוח.

*שימו לב! *ממומשת* ב-submission היוריסטיקה שנקראת `dumb_heuristic1` איננו משתמשים בה בשום מקום אך מה שהיא מבצעת הוא : מחזירה 0 - אם המצב אינו מצב סופי. מחזירה 1 אם ניצחנו ו-1- אם הפסדנו או שהיה תיקו. אתם מוזמנים להסתכל עלייה בכדי להבין כיצד להשתמש ב - ()is_final.

1. (יבש: 6 נק') הגדירו היוריסטיקה משלכם להערכת מצבי המשחק, כתבו נוסחה מפורשת עבור היוריסטיקה. מוזמנים להוסיף תרשים או פירוט מפורט של מה היוריסטיקה עושה בהניתן מצב הלוח והשחקן שמשחק. בחרו בהסבר שמות ברורים ומשמעותיים.
2. (רטוב: 5 נק') *ממשו* בקובץ submission.py את `greedy_improved` וממשו את היוריסטיקה החכמה שלכם תחת הפונקציה `smart_heuristic.` (אל תשנו חתימות של פונקציות זה יכשיל את הטסטים!)

   greedy_improved הינה פונקציה המקבלת:
   - curr_state - מצב נוכחי.
   - agent_id - השחקן *שמשחק* כעת - אם זהו השחקן הראשון יועבר 0 אם זהו השחקן השני יועבר 1.

- time_limit - הגבלת הזמן (בשלב זה אתם יכולים להתעלם ממנה, היא תיהיה רלוונטית באלגוריתמים הבאים).
  הפונקציה מחזירה את הפעולה שהסוכן צריך בוחר לבצע כזוג סדור (pawn, location) כפי שפורט בהסבר על הסביבה (זהה לקלט של מתודת step()).

3. (יבש : 2 נק') הסבירו את המוטיבציה לשינויים שביצעתם בהיוריסטיקה האם לפי דעתכם סוכן חמדן המבוסס על היוריסטקיה שלכן (greedy_improved) ינצח את הסוכן החמדן (greedy)? אם כן, פרטו מדוע.
4. (רטוב: 2 נק') הריצו את השורות שנמצאות בהערה ומעליהן #PART2 וצרפו את התוצאות שיודפו למסך כתוצאה מכך. אתם בעצם תריצו:
   - חמדן נגד אקראי (random)
   - חמדן משופר נגד אקראי (random)
   - חמדן נגד חמדן משופר

חלק ג - RB heuristic MiniMax
(20 נק')

1. (יבש: 2 נק') מה היתרונות והחסרונות של שימוש בהיוריסטיקה קלה לחישוב לעומת היוריסטיקה קשה לחישוב בהינתן שהיוריסטיקה הקשה לחישוב יותר מיודעת מהקלה לחישוב (נותנת אינפורמציה טובה יותר לגבי השאלה מהו מצב טוב)? בהינתן שאנו בmin-max מוגבל משאבים.
2. (יבש: 3 נק') דני מימש את האלגוריתם RB-heuristic-MiniMax והריץ אותו ממצב s בו קיימת פעולה בודדת שמביאה לניצחון. דני נדהם לגלות שהאלגוריתם לא בחר בפעולה זו. האם בהכרח יש טעות באלגוריתם שדני כתב? אם כן, הסבר מה הטעות, אחרת, הסבר מדוע האלגוריתם פעל באופן זה.
3. (יבש: 3 נק') למדתם בהרצאות ובתרגולים גישה שנקראת anytime search. כיצד היא מתמודדת עם הגבלת הזמן ? איזה בעייה נפוצה יש באיטרציה האחרונה ואיך פותרים אותה?
4. (יבש: 4 נק') נניח שבסביבה היו K שחקנים במקום 2 (תחשבו על משחק כללי לוא דווקא המשחק שלנו, אך עדיין משחק סכום אפס). אילו שינויים יהיה צריך לעשות במימוש סוכן Minimax?
   a. בהינתן שכל סוכן רוצה לנצח ולא אכפת לו רק ממכם

b. בהנחה שכל סוכן שונא אתכם והדבר היחיד שאכפת לו זה שלא תנצחו

5. (רטוב: 8 נק') עליכם לממש את הפונקציה `rb_heuristic_min_max` בקובץ submission.py. שימו לב כי הסוכן מוגבל משאבים, כאשר המשתנה time_limit מגביל את מספר השניות שהסוכן יכול לרוץ לפני שיחזיר תשובה.(בסעיף זה אסור להשתמש בגיזום כדי לפתור את הבעייה - אל תדאגו בחלק הבא יהיה לכם גיזום).

* מומלץ להריץ את הסוכנים שלכם אחד נגד השני כמו בחלק א ולראות כי אתם מקבלים תוצאות הגיוניות

* שימו לב שהגבלת הזמן השתנתה מ-1 שנייה ל - 80 שניות

*השינויים הם בקובץ game.py

חלק ד - Alpha_Beta
(20 נק')

1. (רטוב: 10 נק') כעת אתם ממשו סוכן אלפא בטא, בקובץ submission.py ממשו את הפונקציה `alpha_beta` כך שתבצע גיזום כפי שנלמד בהרצאות ובתרגולים.
2. (יבש: 3 נק') האם הסוכן שמימשתם בחלק ד' יתנהג שונה מהסוכן שמימשתם בחלק ג' מבחינת זמן ריצה ובחירת מהלכים?
3. (יבש: 2 נק') למדתם מספר שיטות לשיפורים של alpha_beta אחת מהשיטות הללו נקראת ספריות פתיחה / סיום. מה שיטה זו מבצעת ומדוע היא עוזרת לגזום נתח מהעץ?
4. (יבש: 3 נק') למדתם על גישה נוספת שנקראת טבלאות מצבים. מה השיטה מבצעת, איך היא שומרת על נכונות, ומדוע היא עוזרת "לגזום" מהעץ?
5. (יבש: 2 נק') בהרצה של $Minimax - \alpha\beta$ המתכנת מחק בטעות את הצעד המומלץ ונשאר רק עם ערך המינימקס של העץ. כיצד ניתן לשפר את יעילות ההרצה החוזרת באמצעות שינויים פשוטים באלגוריתם? תארו את התנהגות

האלגוריתם המתוקן. הסבירו כיצד יתנהג במקרה הטוב ביותר, במקרה הרע ביותר, ובמקרה הכללי.

חלק ה - Expectimax
(20 נק')

1. (יבש: 2 נק') סוכן Minimax (עם או בלי שיפורים) מניח כי היריב בוחר בכל צעד בפעולה האופטימילית עבורו, מה בעייתי בגישה הזו ואיך אלגוריתם Expectimax מתגבר על הבעייתיות שתיארתם?
2. (יבש: 3 נק') בהנחה ואתם משתמשים באלגרותים Expectimax נגד סוכן שמשחק באופן רנדומלי לחלוטין באיזה הסתברות תשתמשו? ומדעו?
3. (יבש: 5 נק') עבור משחקים הסתברותיים כמו שש בש, בהם יש מגבלת משאבים, משתמשים באלגוריתם RB-Expectimax. הניחו כי ידוע שהפונקציה היוריסטית h באלגוריתם Expectimax-RB מקיימת $\forall s: -6 \leq h(s) \leq 6$
   a. איך ניתן לבצע גיזום לאלגוריתם זה? תארו בצורה מפורטת את תנאי הגזימה, והסבירו את הרעיון מאחוריו.
   b. הציגו דוגמה להיוריסטיקה כזאת עבור המשחק בתרגיל שלנו וצרפו דוגמא ללוח עבור כל אחד מהמצבים הבאים:
      - מצב המקיים $h(s) = 6$
      - מצב המקיים $h(s) = -6$
4. (רטוב: 10 נק') כעת תממשו סוכן expectimax ע"י כך שתממשו בקובץ submission.py את הפונקציה `expectimax`, תחת ההנחה כי היריב שלנו אוהב אקשן ומחליט שאת כל הפעולות מבצע בהסתברות שווה למעט שתי סוגי פעולות:
   a. לפעולות שבהן יכול לאכול חייל אחר יש הסתברות גבוהה פי 2 משאר הפעולות
   b. הוא מעדיף חיילים קטנים (בגודל S) ולכן לפעולה שמערבת שחקן S יש גם כן הסתברות גבוהה פי 2 משאר הפעולות

בונוס בונוס בונוס בונוס בונוס

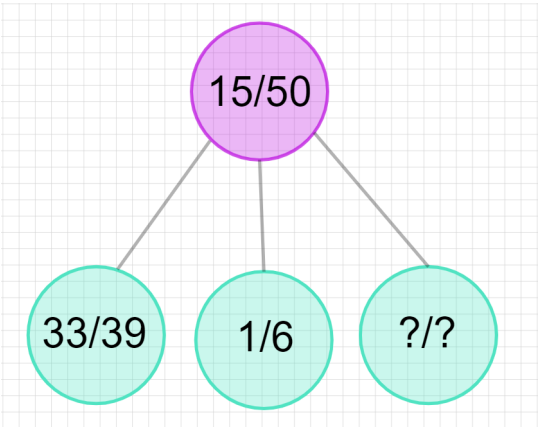
כפי שאתם יודעים האלגוריתמים שמימשתם הינם אדברסיאלים, משמע מתחרים אחד בשני, ולכן אנו מזמינים אתכם לכתוב סוכן שיתחרה בסוכנים של שאר הסטודנטים בקורס עליכם לממש אותו תחת הפונקציה supre_agent יש לכם יד חופשית בבחירת האופן בה תממשו את הסוכן (יכולים לבחור אפילו אחד מהאלו שמימשתם בתרגיל). שני הזוגות שינצח מול הכי הרבה קבוצות אחרות יקבלו 10 נק' בונוס לציון הסופי של תרגיל הבית הנ"ל. ארבעת הזוגות שאחריהם יקבלו 5 נק' בונוס.

שאלה פתוחה - Monte Carlo Tree Search
(21 נק')

1. (2 נק') הסבירו את המונחים הבאים:
   a. exploitation
   b. exploration

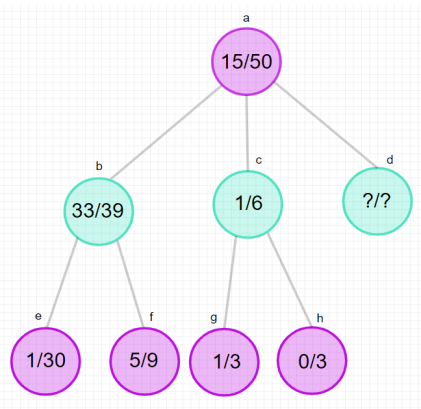
אנדריי ואופק החליטו לשחק קאטאן (נניח שזה משחק לשני אנשים), הם החליטו שהמשחק פשוט מידי והחליטו להוסיף את ההרחבות: "ערים ואבירים" ו"יורדי הים". אחרי שהבינו שהגזימו וכעת אינם בטוחים מה הצעד החכם ביותר ובגלל שה-branching factor גדול מאוד החליטו להשתמש ב- MCTS (Monte Carlo Tree Search) (דגש - זהו משחק סכום אפס שנגמר בניצחון של אחד הצדדים)

להלן עץ שמתאר שני שלבים במשחק. כחול זה תור של אופק וורוד זה תור של אנדריי (פחות רלוונטי לשני הסעיפים הראשונים)



2. (2 נק') עבור העץ הנ"ל שמתאר שני שלבים במשחק עבור העלה עם ה- (?) השלימו מהו הערכים שאמורים להיות במקום הסימני שאלה.
3. (7 נק') חשבו את $UCB_1(s)$ עבור הצמתים הכחולים עם ערך $C = \sqrt{2}$, השוו בין הערכים שיצאו לכם בחישוב ה - $UCB_1(s)$, הסבירו מה המשמעות שלהם. האם יש צמתים (רק מהכחולים) שהערך שלהם שונה מהותית אחד מהשני אך ה - $UCB_1(s)$ דומה? אם כן הסבירו מדוע מצב כזה קורה ע"פ העקרונות של UCB_1

כעת חושפים לנו המשך של שני השלבים שראינו והעץ כעת נראה כך:



4. (4 נק') מצאו וציינו מיהו הצומת הבא שיפותח
5. (6 נק') בהנחה ולו יש בן יחיד שלאחר סימולציה ממנו מתקבל הפסד של אופק. ציירו את העץ החדש הנוצר (שנו את ערכי שאר הצמתים בהתאם) לאחר שנפעפע את המידע על הסימולציה שהתרחשה.

- **להאריך את ההגבלת זמן**
- **בונוס: 3' נק מדוע הוא לא**

- **5 משחקים מול 5 משחקים ?**