**תרגיל בית 2 מבוא לבינה מלאכותית**

**Ex2 Introduction to AI**

**Gobblet Gobblers**

**(enhanced TicTacToe)**

****

**הקדמה ואדמיניסטרציה**

**הנחיות כלליות**

* תאריך הגשת התרגיל: **29.12.2022**
* 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](https://www.youtube.com/watch?v=AxZlrnyag18)

**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](https://themindcafe.com.sg/wp-content/uploads/2018/07/Gobblet-Gobblers.pdf)

**מה קיבלתם?**

קיבלתם 4 קבצים מרכזיים:

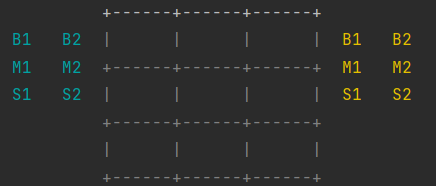
1. Gobblet\_Gobblers\_Env.py סביבת המשחק אותו ישחקו הסוכנים שלכם - הסבר מפורט בהמשך
2. submission.py קובץ שבו תממשו את הסוכנים שלכם וזהו גם הקובץ אותו אתם מגישים
3. game.py מכיל שתי פונקציות שמטרתן להריץ את המשחק ויעזרו לכם לבדיקה עצמית - הסבר מפורט עליהן בהמשך
4. main.py משמש גם כן לבדיקה עצמית

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



\*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: 



דוגמא לשימוש בפונקציה

המשמעות: נזיז את גובלין B1 לאריח מספר 7.



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

o שימו לב, כלים שנמצאים על כלים אחרים יסתירו את הכלים שמתחתיהם, כדי לדעת בדיוק איפה כל כלי ראו ()get\_state.

o 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 שלהן. 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: 

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.

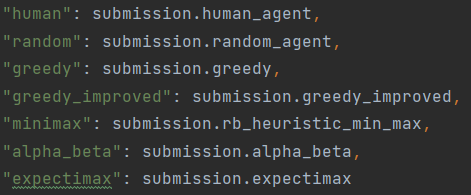
**()is\_final** - מבלבל אז חשוב לשים לב! פונקציה שמקבלת מצב מסוים ומחזירה :

**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!

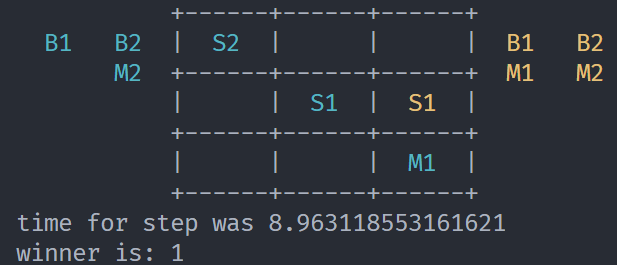
**2** - ניצחון של השחקן השני.

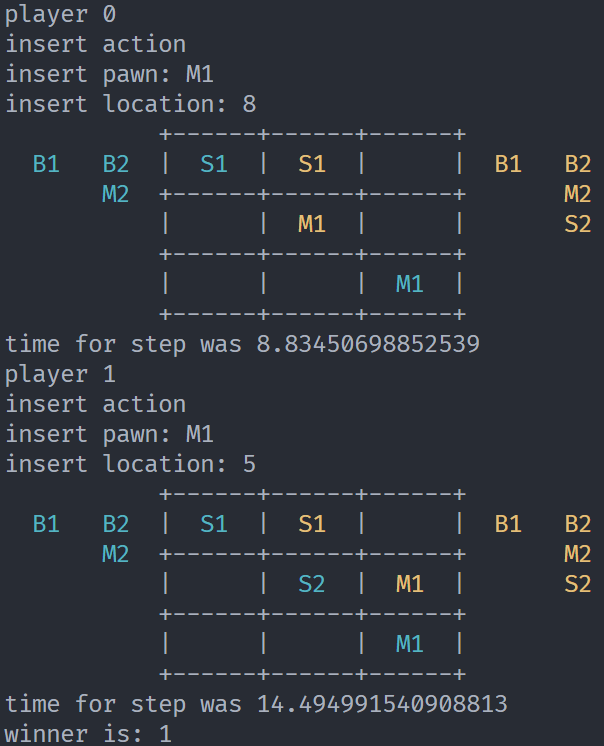
* **()play\_game** - פונקציה שמריצה משחק בודד. הפונקציה הנ"ל מקבלת מחרוזות שמתארות כל סוכן כפי שמתואר בתצלום של המילון בפונקציה הבאה. בנוסף יש דוגמא בהמשך. הפונקציה מחזירה מיהו המנצח ומדפיסה זאת למסך. מחזירה תוצאות כמו שמחזירה ()is\_final.
* **()play\_tournament** - פונקציה המריצה מספר play\_games לפי 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?

כן, אם השחקן מזיז כלי שכבר מונח על הלוח והוא שוכח שמתחתיו יש כלי של השחקן היריב אותו הוא אכל, הוא עלול להוביל למצב שנוצר מצב מנצח עבור השחקן היריב. לדוגמה:

1. (יבש: 1 נק') מהו המספר המקסימלי של אפשרויות לפעולות שונות (בהנחה שכל פעולה חוקית) שניתן לעשות בתור?

יש כלים שונים שהשחקן יכול להניח בכל אחד מ- המשבצות לכל היותר, ולכן המספר המקסימלי של אפשרויות לפעולות שונות הינו .

**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.

ראשית הגדרנו ערך לכל כלי משחק –

B\_PAWN = 5

M\_PAWN = 3

S\_PAWN = 1

מערכים אלה בנינו את לוח המשחק לפי המצב הנוכחי כאשר לוח המשחק הוא מערך numpy ממימדים כלומר מטריצה שבכל משבצת שלה יש וקטור מימדי שכל כניסה בו מיצגת **רובד** על הלוח. עבור משבצת כלשהי במטריצה, המקום הראשון בוקטור הוא הרובד החיצוני ביותר כך שהכלים בו הם הכלים שגלויים על הלוח, והמקומות השני והשלישי הם הרבדים הפנימיים יותר, כך שכאשר יש כלי שמוסתר תחת כלי אחר – הוא יופיע במקומות אלה.  
*משבצת לא מאוכלסת על לוח המשחק תיוצג בוקטור הפנימי במטריצה בעזרת הערך , שעל לוח המשחק ייוצג במקום המתאים במטריצה בעזרת הערך המתאים לו כפי שהוגדר למעלה – כאשר כלי של היריב ייוצג באמצעות ערך שלילי (זהה בגודלו) וכלי של הסוכן ייוצג באמצעות ערך חיובי.*

*נסמן את הלוח בעזרת האות .*

*בנוסף, הגדרנו משקל על כל משבצת בלוח המשחק כך שהמשבצת המרכזית קיבלה את המשקל הגדול ביותר, המשבצות בפינות הלוח קיבלו את המשקל הבא בסדר גודלו, ושאר המשבצות (אמצע כל שורה ועמודה חיצוניים) קיבלו את המשקל הנמוך ביותר.  
המשקלים הללו מיוצגים גם הם בעזרת מערך עם מימדים זהים לשל ונסמן את המערך הזה בעזרת האות (עבור ).*

*לבסוף, הגדרנו דעיכה של הערכים ברבדים השונים כך שהרובד החיצוני הגלוי לא ספג דעיכה כלל וקיבל משקל של , וככל שהרבדים הלכו והעמיקו הם קיבלו דעיכה שהלכה וגדלה כך שהמשקל במקומות אלה הוגדר להיות מספר כלשהו בין לבין .  
לדוגמה – הרובד החיצוני והגלוי מקבל משקל , הבא אחריו משקל ולבסוף ( כאשר הוא מספר השכבה בין לבין ).  
גם משקלים אלה מיוצגים בעזרת מערך עם מימדים זהים לשל ונסמן את מערך זה בעזרת האות (עבור ).*

*בנוסף עבור מצב שבו יש מנצח הגדרנו ערך קבוע שגדול יותר מכל ערך היוריסטי בערך מוחלט שניתן לקבל בכל מצב שאין בו מנצח.*

*הנוסחה הסופית לחישוב ההיוריסטיקה הוגדרה באופן הבא:  
( -* Hadamard product – *כפל איבר־איבר)*

1. (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 מחזירהאת הפעולה שהסוכן צריך בוחר לבצע כזוג סדור  
 (pawn, location) כפי שפורט בהסבר על הסביבה (זהה לקלט של מתודת ()step).

1. (יבש : 2 נק') הסבירו את המוטיבציה לשינויים שביצעתם בהיוריסטיקה. האם לפי דעתכם סוכן חמדן המבוסס על היוריסטקיה שלכם (greedy\_improved) ינצח את הסוכן החמדן (greedy)? אם כן, פרטו מדוע.

ראשית שמנו לב שהיוריסטיקה שהיתה קיימת (נקרא לה יוריסטיקה "נאיבית") לא התחשבה כלל בכלים של היריב, לא תיעדפה השמה של כלים מסויימים על פני אחרים, לא תיעדפה משבצות כאלה ואחרות על פני אחרות והתעלמה לחלוטין מכלים שהיו מוחבאים תחת כלים אחרים. אנחנו חשבנו שחשוב להתייחס גם לנקודות האלה ולכן הגדרנו את היוריסטיקה שלנו כך שתתחשב גם בפרטים אלה.  
הגדרנו תועלת חיובית עבור כלים של הסוכן ועונש שלילי עבור כלים של היריב.  
הגדרנו משקל על כל כלי שמעודד השמה של כלים גדולים קודם על פני כלים קטנים יותר.  
הגדרנו גם אוסף משקלים על משבצות הלוח כך שתעודד השמה של כלים במשבצות שמשפיעות על יותר שורות/עמודות/אלכסונים (וכן ידוע לנו מאינטואיציה ממשחקי איקס-עיגול שאלה הן בדרך כלל משבצות טובות לשחק בהן).  
כמו כן, רצינו להתחשב גם בכלים שמוחבאים תחת כלים אחרים אך להקטין משמעותית את השפעתן ככל שהעומק שלהם הולך וגדל – ולכן גם התייחסנו לכך.

לדעתנו, מכל הסיבות שנאמרו לעיל, הסוכן שמבוסס על ההיוריסטיקה שלנו ינצח בקלות את הסוכן החמדן, שכן הסוכן החמדן שמשתמש ביוריסטיקה הנאיבית, כשם ההיוריסטיקה, נאיבי מדי.

1. (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.

היוריסטיקה קלה לחישוב תחושב בפחות זמן, וכאשר האלגוריתם מוגבל משאבים זמן החישוב הזה חשוב מאוד, שכן יהיה ניתן להעמיק בעץ יותר ולקבל החלטות שהוחלטו על סמך בחינה של אוסף גדול יותר של מצבים.

אמנם, אם ההיוריסטיקה הקלה לחישוב הזו לא מיודעת מספיק, ההחלטה שנקבל עדיין עלולה להיות רעה למרות שהיא התקבלה על ידי אוסף גדול יותר של מצבים שנבחנו, וזהו יתרון בשימוש בהיוריסטיקה מיודעת יותר (כמו, בשאלה – ההיוריסטיקה הקשה לחישוב).

1. (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.

לא בהכרח יש טעות באלגוריתם.

למשל, אם המצב הנוכחי הוא מצב שממנו עבור כל פעולה של הסוכן הוא יגיע למצב שגם ממנו קיים מסלול מנצח בעץ ה-Minimax כך שכל המצבים השכנים יקבלו את אותו הערך (ערך ה-Utility של מצב מנצח), האלגוריתם לא יתן עדיפות יתרה עבור אף אחד מהמצבים והוא עלול שלא לבחור בפעולה שתוביל לניצחון מיידית שכן אין לסוכן "מוטיבציה" לנצח כמה שיותר מהר.

1. (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?

גישת ה- מתמודדת עם הגבלת הזמן על ידי כך שהיא מחשבת את המהלך הטוב ביותר שניתן לחישוב תוך שניות – זמן מוגבל.

ישנה בעיה נפוצה באיטרציה האחרונה. בממוצע, האלגוריתם יסתיים באמצע האיטרציה האחרונה ובמצב זה הערך שיחזור יהיה מהאיטרציה הקודמת, וכשה- גדול, זה אומר שרוב המשאבים בוזבזו עבור האיטרציה האחרונה אך נזרקו לפח מכיוון שהאלגוריתם נקטע באמצע, וזהו בזבוז גדול של משאבים.

פתרון אפשרי לבעיה זו כולל למיין את סדר פיתוח המצבים לפי היוריסטיקה כלשהי, ובאיטרציה האחרונה לפתח רק את המצבים העוקבים ה"טובים יותר היוריסטית", ועבור שאר המצבים העוקבים להשתמש בהיוריסטיקת הערכה.

באופן זה, האיטרציה האחרונה תסתיים תוך זמן סביר יותר ונוכל להשתמש גם במידע שלה כדי לקבל החלטה.

1. (יבש: 4 נק') נניח שבסביבה היו K שחקנים במקום 2 (תחשבו על משחק כללי לווא דווקא המשחק שלנו, אך עדיין משחק סכום אפס). אילו שינויים יהיה צריך לעשות במימוש סוכן Minimax?
   1. בהינתן שכל סוכן רוצה לנצח ולא אכפת לו רק ממכם

במצב זה כל שחקן ינסה להביא למקסימום את התועלת של עצמו ויניח שזו גם מטרתם של שאר הסוכנים במשחק. לכן מימוש סוכן ה-Minimax יצטרך להשתנות באופן הבא:

בעצם, בכל תור, כל שחקן יבחר את הצעד שיוביל אל התועלת הגדולה ביותר האפשרית במסלול כלשהו החל מהמצב הבא שאליו ילך.

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

במצב זה, כל שחקן יריב יבחר את הצעד שיביא למינימום את התועלת **שלי** מבלי להסתכל כלל על התועלת שלו, ובידיעה ומידול העולם בצורה הזו – אני אבחר את הפעולה שתמקסם את התועלת שלי כשאהיה מודע לכך שהיריב יחפש את הפעולה שתרע לי הכי הרבה.  
מימוש ה-Minimax נותר ללא שינוי.

1. (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

**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?

אנחנו צופים שהאלגוריתם שמימשנו בחלק זה ירוץ מהר יותר שכן הוא לא בהכרח יפתח את כל המצבים שהאלגוריתם מחלק ג' מפתח, והוא יכול לסיים את החישוב הרבה יותר מהר בעקבות כך.

מאחר ששני הסוכנים הם סוכנים שמוגבלים בזמן, אנחנו צופים שגם בחירת המהלכים תהיה שונה שכן באותו פרק הזמן סוכן ה-Alpha Beta יצליח לסיים יותר איטרציות העמקה (iterative deepening) וכן יקבל החלטות יותר מיודעות לגבי המהלך הבא שלו.

1. (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?

בשיטה זו נזכור רצפי משחק/תוכניות פעולה למצבים נפוצים, ונשתמש בהם אם נגיע למצב שהוא חלק מרצף כזה.

שימוש בתוכניות פעולה שמוגדרות מראש חוסכות חישוב של תתי עצים במהלך חיפוש ה-Minimax שכן אם מגיעים למצב נפוץ ניתן לבצע ממנו פעולה באופן שנשמר מראש בספריית הפתיחה/הסיום מבלי לבצע חישובים נוספים.

1. (יבש: 3 נק') למדתם על גישה נוספת שנקראת טבלאות מצבים. מה השיטה מבצעת, איך היא שומרת על נכונות, ומדוע היא עוזרת "לגזום" מהעץ?

השיטה מנצלת מקום כדי לייעל חישוב חוזר של מצב שנצפה בשנית.

כאשר מחשבים ערך של מצב כלשהו שומרים את ערכו בטבלת מצבים ביחד עם העומק שבו המצב נצפה. כאשר פוגשים באותו המצב בשנית בעץ החיפוש, משתמשים בערך שנשמר בטבלה במקום לחשבו שוב.

באופן זה חוסכים ירידה לתתי עצים של מצבים שכבר ירדנו לתתי העצים שלהם בעבר במהלך תהליך החיפוש ובמקום זאת משתמשים בערכים שכבר חושבו עבורם בעבר.

כדי לשמור על נכונות הערכים שמבצעים בהם שימוש חוזר עלינו להיות זהירים ולוודא שמשתמשים בהם רק כאשר המצב שנתקלנו בו בשנית הוא בעומק שגדול או שווה לעומק שבו היה המצב בפעם הראשונה שנתקלנו בו והוספנו אותו לטבלת המצבים, כי בסיטואציה זו הערך חושב על ידי תת עץ של המצב בעומק שהוא לפחות כמו העומק של המצב שנצפה בשנית.

אם נשתמש בערך מהטבלה כאשר המצב התגלה בשנית בעומק נמוך יותר, הערך יהיה מיודע פחות שכן תת העץ עכשיו גדול יותר מתת העץ שבו חושב ערך s, אבל אם נשתמש בו נחסוך הרבה חישובים ולכן זהו tradeoff, וצריך לקבל החלטה לגבי מהי העדיפות.

1. (יבש: 2 נק') בהרצה של 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.

נסמן את ערך המינימקס מהריצה האחרונה על העץ ב- ונשים לב כי הוא חסם עליון של וחסם תחתון של לאחר ריצת האלגוריתם.

לכן, כעת במקום לאתחל את למינוס אינסוף ואת לאינסוף ניתן לאתחל אותם באופן הבא:

*נגדיר קטן כלשהו כך ש- ו- יאותחלו בסביבת של ערך המינימקס. יש לשים לב לא לבחור קטן מדי כדי שלא יגזום את המסלול הרלוונטי משורש העץ, אבל כן מספיק קטן כדי שהגיזום לפיו יגזום כמה שיותר תתי עצים. מצד שני, אם יהיה גדול מדי, הריצה החוזרת של האלגוריתם עלולה לא לגזום מספיק תתי עצים והריצה החוזרת תיקח משך זמן דומה לשל הריצה הקודמת.*

*בעצם, נאתחל:*

*ונריץ את האלגוריתם המתוקן עם השלבים שמחזירים את הצעד המומלץ.*

*באופן זה, במקרה הכללי יגזמו כמה שיותר תתי-עצים שמיותר לפתחם וגם נצליח לקבל חזרה את הצעד המומלץ שהיה אמור לחזור בהרצה הקודמת (או לפחות צעד שיוביל אל מצב עם ערך זהה).*

*במקרה הגרוע ביותר, סדר הפיתוח של המצבים יגרום לכך שלא יתבצע גיזום כלל, או כמעט בכלל.*

*במקרה הטוב ביותר, פיתוח המצבים מצומת מקסימום השכנים יסודרו בסדר מהגדול לקטן ומצומת מינימום יסודרו מהקטן אל הגדול. באופן זה כמות המצבים שיגזמו היא מיטבית.*

**חלק ה - Expectimax**

**(20 נק')**

1. (יבש: 2 נק') סוכן Minimax (עם או בלי שיפורים) מניח כי היריב בוחר בכל צעד בפעולה האופטימילת עבורו, מה בעייתי בגישה הזו ואיך אלגוריתם Expectimax מתגבר על הבעייתיות שתיארתם?

הסקה שמבוססת תמיד על המקרה הגרוע ביותר היא שמרנית מדי וגם לא בהכרח מציאותית – שכן היריב לא בהכרח תמיד יבחר את הפעולה האופטימלית עבורו. שימוש באלגוריתם Expectimax מתגבר על הבעייתיות הזו בכך שהוא מבצע הסקה על סמך המקרה הממוצע – בעזרת התוחלת של התועלת הטובה ביותר.

1. (יבש: 3 נק') בהנחה ואתם משתמשים באלגרותים Expectimax נגד סוכן שמשחק באופן רנדומלי לחלוטין באיזה הסתברות תשתמשו? ומדעו?

אם ידוע שהיריב ישחק תמיד בצורה אקראית לגמרי, הרי שהבחירות שלו נדגמות מתוך מרחב התפלגות אחיד (Uniform), ולכן נמדל את צמתי ה-Chance גם בעזרת פונקציית הסתברות אחידה ובכך נוכל לשערך באופן הכי קרוב לאיך שהיריב שלנו משחק מהו המקרה הממוצע הטוב ביותר.

1. (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
2. איך ניתן לבצע גיזום לאלגוריתם זה? תארו בצורה מפורטת את תנאי הגזימה, והסבירו את הרעיון מאחוריו.

עבור ההיוריסטיקה נסמן: .

עבור צומת נסמן את המצבים השכנים בעזרת הקבוצה:

חישוב תוחלת התועלת של צומת הסתברות כלשהו יסומן:

*ונסמן בנוסף את החישוב החלקי של תוחלת התועלת עבור צומת הסתברות באופן הבא:*

*נבחן מתי נגזום בכל אחד מהמקרים , לפי סוג צומת האב של צומת ההסתברות הנוכחי – בין אם הוא צומת מינימום או מקסימום.*

*אתחול: .*

*אם צומת האב של צומת ההסתברות הוא צומת מקסימום:*

*נשים לב שבמקרה הטוב ביותר לאחר חישוב נוכל להוסיף לחישוב החלקי הזה את הערך:*

*וזאת כדי שהערך המחושב יהיה הכי קטן שאפשר. אמנם, אם ערך זה גדול או שווה ל-, כלומר – עדיין גדול יותר מהחסם העליון על ערכי המקסימום,* ***עלינו לגזום אותו****. (שכן אף צומת מינימום לא יבחר בו במעלה העץ)*

*בנוסף, נעדכן את באופן הבא בצומת המקסימום:*

*באופן דומה ואנטי־סימטרי נעבוד בצומת המינימום:*

*נשים לב שבמקרה הטוב ביותר לאחר חישוב נוכל להוסיף לחישוב החלקי הזה את הערך:*

*וזאת כדי שהערך המחושב יהיה הכי גדול שאפשר. אמנם, אם ערך זה קטן או שווה ל-, כלומר – עדיין קטן יותר מהחסם התחתון על ערכי המינימום,* ***עלינו לגזום אותו****. (שכן אף צומת מקסימום לא יבחר בו במעלה העץ)*

*בנוסף, נעדכן את באופן הבא בצומת המינימום:*

1. הציגו דוגמה להיוריסטיקה כזאת עבור המשחק בתרגיל שלנו וצרפו דוגמא ללוח עבור כל אחד מהמצבים הבאים:   
    - מצב המקיים

* מצב המקיים

נגדיר את ההיוריסטיקה הבאה:

כך ש- הוא מספר הכלים החשופים של הסוכן על הלוח במצב , ו- הוא מספר הכלים החשופים של היריב על הלוח במצב .

נשים לב כי לכל שחקן יש כלים בסך הכל ולכן ההיוריסטיקה חסומה בין לבין .

|  |  |  |
| --- | --- | --- |
| **B2** | **M2** |  |
| **S2** |  | **M1** |
|  | **S1** | **B1** |

מצב שמקיים :

|  |  |  |
| --- | --- | --- |
| **B2** | **M2** |  |
| **S2** |  | **M1** |
|  | **S1** | **B1** |

מצב שמקיים :

*(כלי הסוכן ב****כחול****, כלי היריב ב****אדום****).*

*נציין שככל הנראה לא ניתן להגיע למצב כזה במשחק רגיל, אבל זהו עדיין מצב תקין במרחב המצבים ולכן הדוגמה תקינה.*

1. (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:
   1. לפעולות שבהן יכול לאכול חייל אחר יש הסתברות גבוהה פי 2 משאר הפעולות
   2. 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**

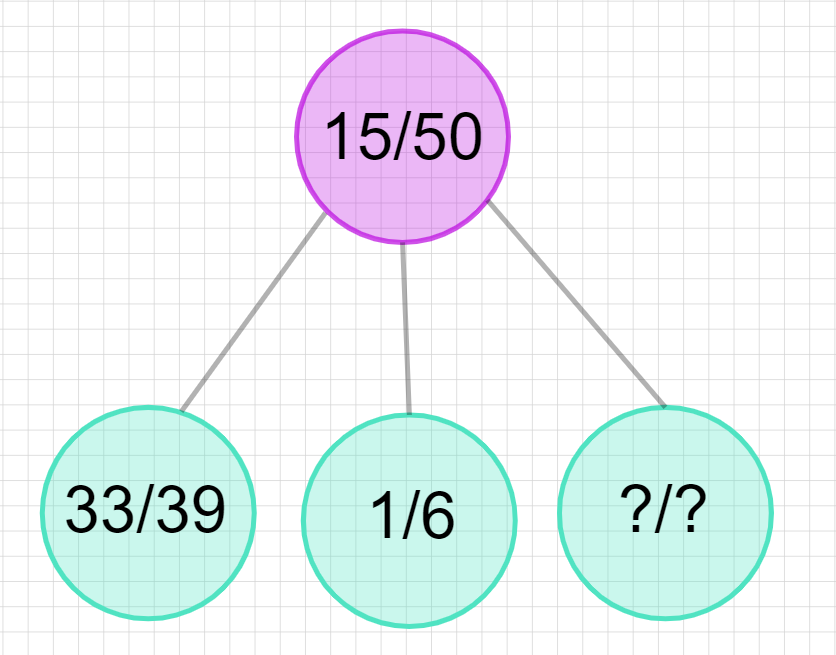
כפי שאתם יודעים האלגוריתמים שמימשתם הינם אדברסיאלים, משמע מתחרים אחד בשני, ולכן אנו מזמינים אתכם לכתוב סוכן שיתחרה בסוכנים של שאר הסטודנטים בקורס עליכם לממש אותו תחת הפונקציה supre\_agent יש לכם יד חופשית בבחירת האופן בה תממשו את הסוכן (יכולים לבחור אפילו אחד מהאלו שמימשתם בתרגיל). שני הזוגות שינצח מול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

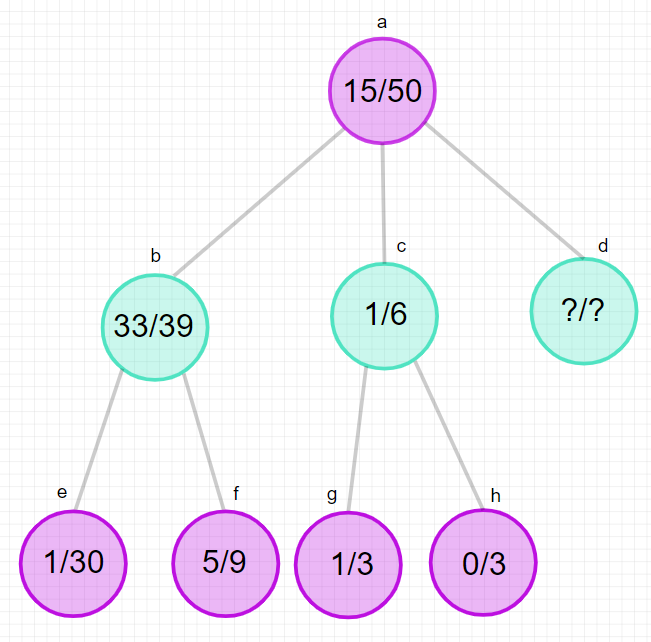
**שאלה פתוחה - Monte Carlo Tree Search**

**(21 נק')**

1. (2 נק') הסבירו את המונחים הבאים:
   1. exploitation
   2. 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)i (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)

1. (2 נק') עבור העץ הנ"ל שמתאר שני שלבים במשחק עבור העלה עם ה- (?) השלימו מהו הערכים שאמורים להיות במקום הסימני שאלה.
2. (7 נק') חשבו את עבור הצמתים הכחולים עם ערך , השוו בין הערכים שיצאו לכם בחישוב ה - , הסבירו מה המשמעות שלהם. האם יש צמתים (רק מהכחולים) שהערך שלהם שונה מהותית אחד מהשני אך ה - דומה? אם כן הסבירו מדוע מצב כזה קורה ע"פ העקרונות של 

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

1. (4 נק') מצאו וציינו מיהו הצומת הבא שיפותח
2. (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.