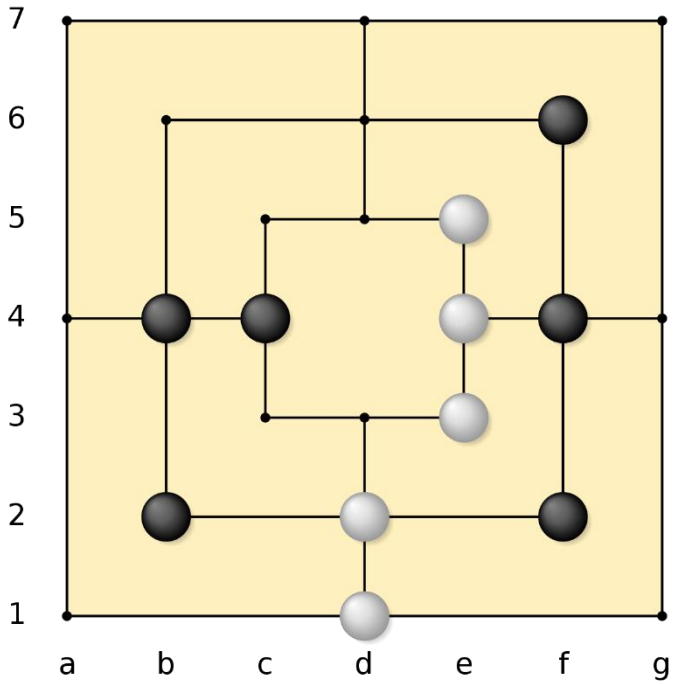
בס"ד

תרגיל בית 2 - חיפוש רב סוכני
טחנה - Nine Men’s Morris



פניות לסגל בנוגע לתרגיל בית יהיו דרך הפיאצה בלבד (לא יהיה מענה דרך המייל).
קישור לפיאצה: piazza.com/technion.ac.il/winter2022/236501

בנוסף, לפני ששואלים שאלה, יש לבדוק אם היא נשאלה בעבר. שאלות שנשאלו בעבר לא יקבלו מענה.

תיאור המשחק

עליכם לממש וריאציה של המשחק: 'טחנה'. זהו משחק אסטרטגיה מופשט לשני שחקנים, ששורשיו בעת העתיקה.

לוח המשחק

המשחק נערך על גבי לוח המכיל 24 צמתים (נמספר אותם 0-23) המצוירים על הלוח על ידי 3 ריבועים בגדלים שונים המחוברים על ידי 4 קווים היוצאים ממרכז צלעות הריבועים.

```
0 (00)------------------------ 0 (01)------------------------ 0 (02)
|                              |                              |
|                              |                              |
|                              |                              |
|        0 (08)--------------- 0 (09)--------------- 0 (10)   |
|       |                      |                      |        |
|       |                      |                      |        |
|       |                      |                      |        |
|       |         0 (16)----- 0 (17)----- 0 (18)    |        |
|       |         |                        |        |        |
|       |         |                        |        |        |
|       |         |                        |        |        |
0 (03)- 0 (11)--- 0 (19)                  0 (20)- 0 (12)--- 0 (04)
|       |         |                        |        |        |
|       |         |                        |        |        |
|       |         |                        |        |        |
|       |         0 (21)----- 0 (22)----- 0 (23)    |        |
|       |                      |                      |        |
|       |                      |                      |        |
|       |                      |                      |        |
|        0 (13)--------------- 0 (14)--------------- 0 (15)   |
|                              |                              |
|                              |                              |
|                              |                              |
0 (05)------------------------ 0 (06)------------------------ 0 (07)
```

מהלך המשחק

לכל שחקן יש 9 חיילים (נמספר אותם 0-8)

המשחק מחולק לשני שלבים.

<u>שלב ראשון</u>:

המשחק מתחיל בלוח ריק. בשלב הראשון מניח כל שחקן בתורו חייל בצומת פנוי. כאשר שחקן יוצר רצף של שלושה חיילים שלו באחד מהקווים הישרים שעל הלוח (רצף הקרוי "טחנה"), הוא מסיר מהלוח חייל של היריב . חייל זה יוצא מהמשחק.

<u>שלב שני</u>:

לאחר שכל שחקן הניח על הלוח תשעה חיילים, מתחיל השלב השני של המשחק בו השחקנים מזיזים, כל אחד בתורו, חייל אחד. הזזת חייל נעשית מהצומת בו הוא נמצא לצומת סמוך[*] פנוי. גם בשלב זה שחקן שיצר טחנה מסיר מהלוח חייל של היריב. שחקן שנותר עם פחות משלושה חיילים או שאינו מסוגל לזוז בהגיע תורו מפסיד במשחק.

[*]צומת סמוך: באיור למעלה, בסמוך לצומת 0 יש את הצמתים 1 ו3. ובסמוך לצומת 9 יש את הצמתים 1, 10, 8, 17.

מטרת המשחק

להותיר בידי היריב פחות מ-3 חיילים (2 או פחות) או לחסום את היריב מלבצע מהלכים.

קישורים

לקריאה על המשחק: https://en.wikipedia.org/wiki/Nine_men%27s_morris

לשחק את המשחק: https://toytheater.com/nine-mens-morris/

(שימו לב, כשאתם קוראים על המשחק\ משחקים את המשחק היות ואנו ממשים ואריציה פשוטה יותר של המשחק, השלב השלישי, בו לאחד מהשחקנים יש פחות מ-3 חיילים לא תקף אצלינו.

הרצת משחקים

לפני תחילת מימוש הסוכנים, מומלץ להתנסות במשחק. הריצו את השורות הבאות ב- terminal בתיקייה בה נמצא הקוד המצורף לתרגיל:

כדי לשחק בתור שני הסוכנים, הריצו את השורה:

```
python main.py -player1 LivePlayer -player2 LivePlayer
```

לאחר הרצת השורות לוח המשחק יופיע בטרמינל ותוכלו להתחיל לשחק.
כדי לשחק נגד שחקן SimplePlayer (שחקן שהמימוש שלו נמצא בקבצי הקוד שקיבלתם) הריצו:

```
python main.py -player1 LivePlayer -player2 SimplePlayer
```

בהמשך נממש את השחקנים MinimaxPlayer, AlphaBetaPlayer ועוד נוספים. כדי להריץ משחק בין שני שחקנים כלשהם הריצו שורה דומה עם השמות של השחקנים המתאימים.

בנוסף אתם יכולים להגדיר:

- מגבלה על זמן מהלך על ידי הדגל move_time-
- הגבלת זמן גלובלי לכל תורות השחקן על ידי הדגל game_time-

חלק יבש (45 נק')

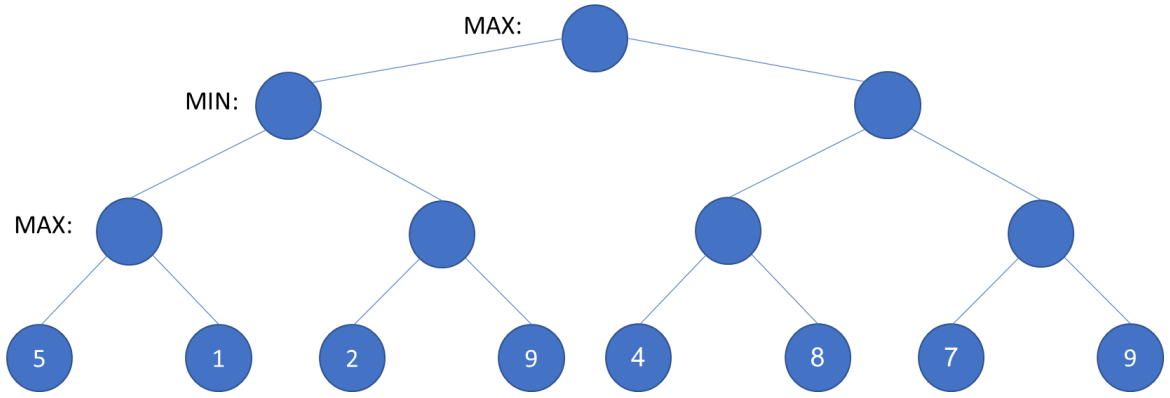
שאלות על הגדרת היוריסטיקה

1. (1.5 נק') נגדיר את היוריסטיקה הבאה:
בהינתן מצב s,
$h(s) = number\ of\ player\ incomplete\ mills - number\ of\ rival\ incomplete\ mills$.
הערך היוריסטי של מצב הוא מספר הקונפיגורציות בהן יש בשלישיה 2 חיילים (של הסוכן) + תא ריק.
מה החסרונות ומה ההיתרונות של היורסטיקה זו?

2. (2 נק') הגדירו היוריסטיקה המורכבת משילוב של לפחות ארבעה מרכיבים. (יסייע לכם לקבל תוצאות טובות בחלק הרטוב)
הציגו בצורה פורמלית הגדרה במסמך שלכם. הסבירו את המוטיבציה שלכם להגדרה היוריסטיקה, והסבר קצר על כל אחד מהפרמטרים שהגדרתם בהיוריסטיקה.

שאלות בנושא minimax ו-alpha-beta

3. א. (2 נק') מה היתרון בהוספת גיזום אלפא-בטא לאלוגריתם מינימקס, ואיך יתרון זה מושג?
ב. (2 נק') איך הייתם ממיינים את צמתי העץ על מנת לקבל גיזום מקסימלי?
ג. (2 נק') העץ למטה מייצג מצב במשחק. עכשיו תורו של max. סמנו כל עלה שיגזם ע"י אלפא בטא.
בנוסף, סמנו את המסלול האופטימלי להמשך המשחק בהינתן הידע שבעץ.

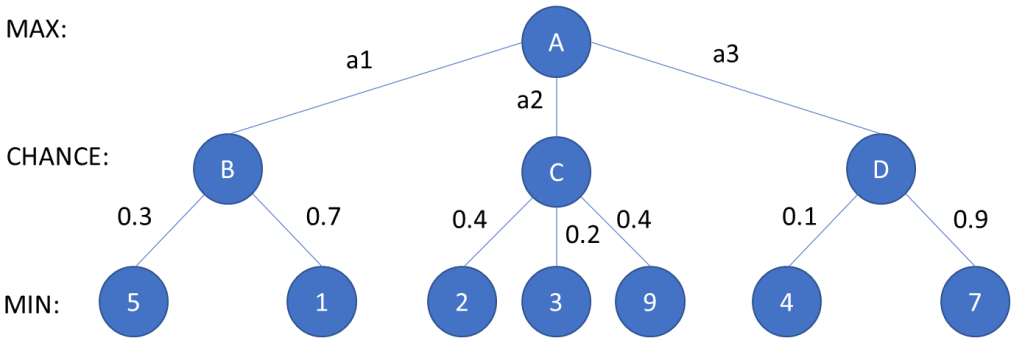


4. (2 נק') אמת או שקר: אם אתם משתמשים באלגוריתם alpha beta עם היוריסטיקה, האם יצרתם סוכן שמשחק עם אסטרטגיה אופטימלית? תנו דוגמא.

5. non zero-sum game: נניח משחק כלשהו שבו קיימים מצבים סופיים s1, s2, s3, s4.
1. (2.5 נק') הגדירו פונקציית utility עבור המצבים הסופיים שלא מגדירה משחק סכום אפס שלפיה מינימקס אופטימלי.
2. (2.5 נק') הגדירו פונקציית utility שלפיה minimax לא אופטימלי. תנו דוגמה לעץ חיפוש שבו אלגוריתם minimax אינו מתאים.

6. סטודנטית ממשת את האלגוריתם אלפא-בטא ואת הפונקציות הנחוצות למשחק 'טחנה'. במשחק שערכה נגד התוכנה שמה לב שלמרות שהמחשב יכל לנצח את המשחק בצעד הבא, הוא נמנע מלבצע את הצעד הזה ובחר צעד אחר. הסטודנטית בדקה ומצא שכל המימוש נכון ואין באגים בתוכנה.
א. (2.5 נק') איך יתכן מצב כזה? הסבירו בפרוט.
ב. (2.5 נק') הציעו שינוי לאלפא בטא שימנע מצבים כאלה.

7. א. (2 נק') חשבו את ערכי הExpectimax של הצמתים D, B, C. (הראו את החישוב שלכם.)



ב. (2 נק') איזה פעולה MAX יבחר? a1, a2, a3?
ג. (2 נק') גיזום באלגוריתם expectimax.
האם נוכל לגזום בexpectimax באותו אופן שבו אנחנו רוצים לגזום באלגוריתם אלפא בטא בלי לפגוע בנכונות האלגוריתם? יש להביא דוגמא.

8. נרצה שאלגוריתם אלפא בטא יגזום יותר צמתים ויבטיח לנו תשובה נכונה בקירוב הסוטה מהתשובה המדויקת בלכל היותר ϵ. לכן, נוסיף לו את השינוי הבא:
לכל מצב s במשחק, $|Algorithm(state) - minimax(state)| \leq \epsilon$
א. (2.5 נק') להלן פסאדו קוד לאלגורתם אלפא בטא:

```
alphaBeta(state):
        return maxValue(state, -INFINITY, INFINITY, 0)


maxValue(state, alpha, beta, depth):
        if cutoffTest(state, depth):
                return utility(state)

        value = -INFINITY
        for successor in state.getSuccessors():
                value = max(value, minValue(successor, alpha, beta, depth + 1))
                if value >= beta:
                        return value
                alpha = max(alpha, value)

        return value


minValue(state, alpha, beta, depth):
        if cutoffTest(state, depth):
                return utility(state)

        value = INFINITY
        for successor in state.getSuccessors():
                value = min(value, maxValue(successor, alpha, beta, depth + 1))
                if value <= alpha:
                        return value
                beta = min(beta, value)

        return value
```

הוסיפו לקוד שינוי על מנת שנקבל את ההתנהגות המבוקשת.
ב. (2.5 נק') תנו דוגמא עבורה האלגוריתם החדש מחזיר ערך שונה מערך המינימקס.

9. לפני תחרות חשובה במשחק Nine men's Morris, אחד השחקנים שיחד את השופט והצליח להשיג את פרוצדורת ההחלטה rival_move של היריב. הפרוצדורה מקבלת מצב s במשחק, מחשבת את הצעד שיבחר היריב, ומחזירה את המצב אליו יעבור.
לכל אחד מהשחקנים מוקצב זמן קבוע T לכל תור במשחק והוא מאפשר לעשות חיפוש מינימקס לעומק d1. הזמן שלוקח להפעיל את הפרוצדורה rival_move זניח ושוקל לאפס.
א. (2.5 נק') לאיזה עומק d2 נוכל לחפש בזמן T אם נשתמש בפרוצדורה rival_move?
ב. (2.5 נק') בהינתן צומת s, מה היחס בין הערך של המינימקס עם שימוש בפרוצדורה לבין הערך של המיניקס ללא שימוש בפרוצדורה כאשר שניהם מוגבלים לעומק d?

שאלות בנושא חיפוש לוקלי

10. סטודנט קיבל בעיית מקסימיזציה עם מרחב חיפוש בגודל 10^{12}. הוא בוחר לפתור את הבעיה בעזרת SAHC. הוא מריץ את האלגוריתם 1000 פעמים, כל פעם הוא מחליט להתחיל מנקודה רנדומלית אחרת. בכל 1000 הריצות של האלגוריתם, הערך הנמוך ביותר שהוא קיבל היה 0.9, הגבוה ביותר 5.8 והממוצע היה 3.2.
לוקח לאלוגריתם 5 צעדים בממוצע על מנת להתכנס ולהחזיר תוצאה. מכיוון שהסטודנט מרוצה מהחיפוש שלו הוא מדווח שהאופטימום ב5.8.
   א. (2.5 נק') האם הסטונדנט צודק שהוא מדווח ש5.8 האופטימום הגלובלי?
   ב. (2.5 נק') אם אתם הייתם צריכים לודא את תשובתו של הסטודנט על ידי ניסויים נוספים. באיזה אלגוריתם הייתם מעדיפים להתשמש?
   ג. (2.5 נק') צרו מרחב חיפוש בו SAHC לא מוצא פתרון אופטימלי בהסתברות גבוה, אך, האלגוריתם שבחרתם בסעיף ב' ימצא פתרון בהסתברות גבוה.

חלק רטוב (55 נק')

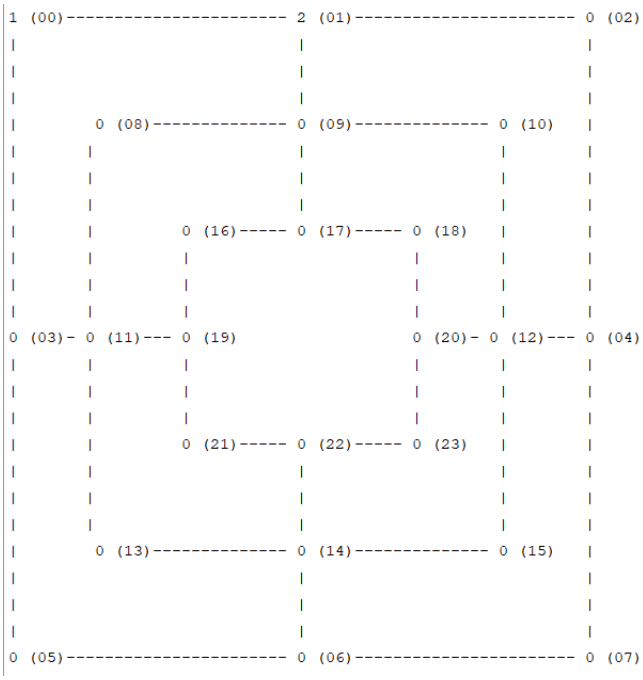
עליכם יהיה לממש סוכנים שמשחקים את המשחק.
כל שחקן ירש מהמחלקה AbstractPlayer.

בקובץ **AbstractPlayer**, מחלקה בשם Player, ובה הפונקציות הבאות:
אנא קראו אותן ואת תפקידן בעיון רב!

1. **Init-** איתחול השחקן. מקבלת פרמטר game_time אשר מהווה הזמן הכולל לכל תורות השחקן.
מאתחלת את הלוח להיות מערך באורך 24, ומאתחלת פונקציה directions אשר בהינתן מיקום של חייל, מחזירה את התאים הסמוכים שאליהם החייל יכול לזוז.

2. **set_game_params** - הגדר את הפרמטרים של המשחק הדרושים לשחקן זה.
פונקציה זו תקרא על ידי הקוד שמנהל את המשחק פעם אחת עבור כל שחקן בתחילת המשחק, ומטרתה להעביר לשחקנים את הלוח שבו יתנהל המשחק (הלוח ההתחלתי תמיד ריק).
פרמטרים: הלוח של המשחק, זהו np.array באורך 24 כאשר כל מספר מייצג משבצת על הלוח.

במהלך המשחק הלוח ישתנה וכל כניסה בלוח תכיל את אחד מהערכים הבאים:



- הערך 0 מסמל משבצת ריקה. (באיור מעל משבצות 2-23 ריקות)
- הערך 1 מסמל שבמשבצת זו יש חייל של שחקן מספר 1. (משבצת 0 באיור מעל)
- הערך 2 מסמל שבמשבצת זו יש חייל של שחקן מספר 2. (משבצת 1 באיור מעל)

3. **make_move-** פונקציה זו תקרא על ידי הקוד שמנהל את המשחק בכל תור של השחקן, ומטרתה לקבל מהשחקן את הצעד הבא שהוא מבצע.
הפונקציה מקבלת כפרמטר את הזמן, time_limit, שיש לשחקן לעשות את התור שלו, במידה והוא יחרוג ממסגרת זמן זו המשחק יסתיים והוא יפסיד במשחק.
פלט הפונקציה הוא הtuple הבא: (pos, soldier, dead_opponent_pos)
pos – מיקום על הלוח בו השחקן רוצה לשים חייל. (int בין 0-23)
Soldier – איזה חייל השחקן בוחר לשים במיקום זה (int בין 0-8). (אתם בוחרים איזה מספר כל חייל).
(שימו לב, זה חייב להיות חייל שיוכל להגיע למיקום זה. משמע, בשלב 1 של המשחק חייל שטרם הונח, ובשלב 2 של המשחק, חייל שנמצא במשבצת סמוכה לחייל זה.)
dead_opponent_pos – במידה והשחקן יצר טחנה, מחזיר את התא ממנו השחקן רוצה להסיר חייל של היריב. במידה והשחקן לא יצר טחנה, עליו להחזיר "-1".

4. **set_rival_move** – פונקציה זו תקרא על ידי הקוד שמנהל את המשחק לאחר כל תור של השחקן היריב, ומטרתה לעדכן את השחקן שלכם בצעד שהשחקן היריב ביצע.

הפונקציה מקבלת כפרמטר את הצעד move שהוא הtuple: (rival_pos, rival_soldier, dead_my_pos) המתייחס לצעד שהיריב עשה.

5. **isMill-** פונקציה אשר בודקת אם השחקן יצר טחנה (mill). הפונקציה מקבלת את התא שבו השחקן רוצה לבדוק האם הוא יצר טחנה ולוח (אופצונאלי, ראה קוד). שימו לב, על הלוח שלכם להיות מעודכן לפני הקריאה לפונקציה זו.
הפונקציות checkNextMill, isPlayer הן פונקציות עזר של פונקציה זו.
שימו לב: הפונקציה **isMill** ממומשת.

הנכם יכולים להוסיף פרמטרים ופונקציות נוספות לקובץ זה.

בקובץ **SearchAlgos**:

שימו לב! קובץ זה הוא קובץ שלכם. הינכם יכולים לשנות את הפרטרים של הפונקציות הקיימות בקובץ זה ולהוסיף אליו כל מחלקה שתרצו.

מחלקה בשם SearchAlgos. ולה פונקציה init.

1. **Init-** מקבלת:
utility– פונקציית תועלת
succ- פונקציית העוקב
perform_move – פונקציה שמבצעת צעד (רובכם כנראה לא תשתמשו בפרמטר זה והוא מאותחל לnone)
goal- פונקציה שבודקת אם אנו במצב מטרה
2. **Search-** מקבלת:
State – מצב נוכחי
depth- הגבלת עומק של אלגוריתם המינימקס
maximizing_player- ערך בוליאני אם אני הmaximizer ערכו יהיה true.

המחלקות Minimax, וAlphaBeta יורשות מSearchAlgos

חלק א' - מימוש שחקן Minimax

בחלק זה נבנה שחקן Minimax מוגבל משאבים בווריאצית Anytime contract.
את השחקן עליכם לממש בקובץ MinimaxPlayer.py, במחלקה ששמה Player.

על השחקן שתבנו יש לממש את הפונקציות set_game_params, make_move, set_rival_move.

הפונקציה __init__ מקבלת פרמטר game_time בו אין שימוש בשחקן זה, לכן יש להתעלם מערך זה.
יש לממש את אלגוריתם ה-Minimax בקובץ SearchAlgos.py תחת המחלקה Minimax.

set_game_params – כפי שהסברנו עליה למעלה.

set_rival_move- כפי שהסברנו עליה למעלה.

make_move- פונקציה זו תקרא על ידי הקוד שמנהל את המשחק בכל תור של השחקן שלכם, ומטרתה לקבל מהשחקן שלכם את הצעד הבא שהוא מבצע.
שימו לב, לפונקציה make_move יש מגבלת זמן, time_limit, שיש לשחקן לעשות את התור שלו, במידה והוא יחרוג ממסגרת זמן זו המשחק יסתיים והוא יפסיד במשחק. דרך טובה להתמודד עם מגבלת הזמן היא באמצעות iterative deepening.

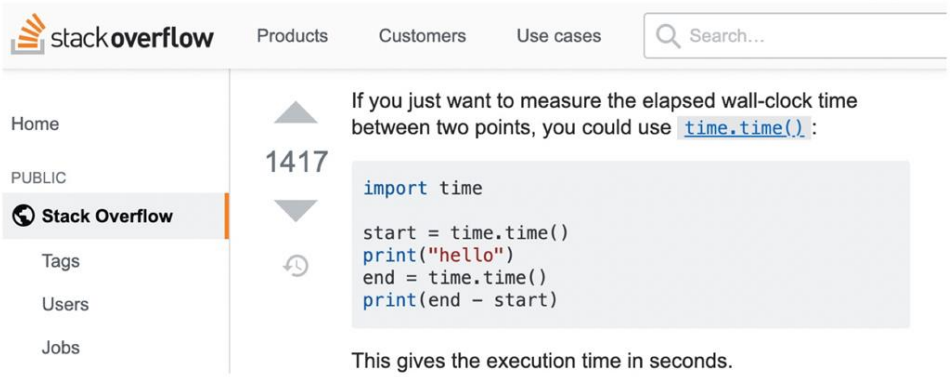
יש לבצע את המימוש של אלגוריתם ה-Minimax בקובץ SearchAlgos.py בתוך המחלקה Minimax.

פונקציית התועלת - היוריסטיקה

כפי שלמדנו בהרצאות, סוכן Minimax מוגבל משאבים משתמש בהיוריסטיקה כדי להעריך את טיב העלים בעץ שאותו פיתח ולקבוע את אסטרטגית המשחק על פי ערכים אלו.

עליכם להגדיר ולממש היוריסטיקה עבור שחקן ה-Minimax, על היוריסטיקה להיות מורכבת מלפחות 4 מרכיבים, כלומר עליכם להגדיר לפחות 4 ערכים שהם פונקציה של המצב אותו אנחנו רוצים להעריך, כך שהיוריסטיקה תהיה מרוכבת מקומבינציה שלהם.

טיפ
כדי להתמודד עם מגבלת הזמן, ניתן להשתמש בספריה time:



חלק ב׳ – מימוש סוכן α-β (12 נק׳)

בחלק זה נשפר את סוכן ה-Minimax שבנינו לסוכן α-β.

את השחקן יש לממש בקובץ ששמו AlphabetaPlayer.py במחלקה ששמה Player.
השחקן יכול להיות זהה לשחקן ה-Minimax, מלבד לכך שיממש את אלגוריתם α-β.

הפונקציה __init__ מקבלת פרמטר game_time בו אין שימוש בשחקן זה, לכן יש להתעלם מערך זה.

על השחקן לממש את הפונקציות set_game_params, make_move, set_rival_move כפי שתוארו למעלה, ולממש את אלגוריתם ה-alpha-beta בקובץ SearchAlgos.py תחת המחלקה AlphaBeta.py.

חלק ג׳ – מימוש סוכן α-β עם מגבלת זמן גלובלית

בחלק זה לסוכן ה- α-β, במקום מגבלת זמן עבור כל תור, ישנה מגבלת זמן כללית עבור כל תורות השחקן ביחד, לכן יש לתכנן אסטרטגיה שתפרוס את הזמן בצורה חכמה על פני תורות השחקן.

את השחקן יש לממש בקובץ ששמו GlobalTimeAB.py במחלקה ששמה Player.

הפונקציה __init__ מקבלת פרמטר game_time אשר מהווה הזמן הכולל לכל תורות השחקן.

על השחקן לממש את הפונקציות set_game_params, make_move, set_rival_move כפי שתוארו בחלק ב׳, מלבד השינוי בפונקציה make_move אשר מתייחס לזמן הגלובלי ולא לפרמטר time_limit.

בשביל לבדוק סוכן זה יש להגדיר בשורה שמריצה את המשחק, את פרמטר game_time בגדלים שונים, ואת הפרמטר move_time כך שיהיה גדול או שווה לפרמטר game_time.

חלק ד׳ – תחרות

כל זוג נדרש לממש שחקן יחיד שייצג אותו בתחרות, בה ישתתפו כל הזוגות. הסטודנטים ששחקניהם יצטיינו בתחרות יזכו בבונוס לציון הסופי של הקורס:
5 נקודות לזוג המנצח, 3 נקודות למקום השני, ו2 נקודות למקום השלישי.
התחרות מוגדרת כך שיש לכל שחקן זמן גלובלי לכל התורות ביחד, כמו שהוגדר בסעיף ד׳.
את שחקן התחרות שלכם יש לממש בקובץ ContestPlayer.py, במחלקה בשם Player.
על המחלקה לממש את הפונקציות , make_move, set_rival_move.
אתם יכולים להגיש את השחקן שמימשתם בחלק ד׳ או לממש שחקן חדש.
במידה והשחקן שתגישו לא ירוץ, יורדו לכם נקודות בתרגיל. ההשתתפות בתחרות היא חובה.

חוקים:
- אסור לשחקן להשתמש ברשת האינטרנט או ברשת מקומית כלשהי.
- אסור לשחקן להשתמש בכל סוג של קוד מקבילי.
- כל שחקן שינסה לרמות או לשבש את קוד היריב או המערכת ייפסל, ויורדו למגישים נקודות בתרגיל.
- חל איסור להשתמש במידע שעובד מראש ונשמר בקובץ.
- אסור להשתמש בחבילות/ספריות שאינן מובנות בפייתון, מלבד networkx, collections והחבילות אשר מבוצע בהם import בקוד שסופק לכם.
- כל חישוב לשם קביעת הצעד הבא של הסוכן צריך להתבצע אך ורק בקריאה לפונקציה.make_move

חלק ה׳ – כתיבת דו״ח

ענו על הסעיפים בהמשך לשאלות בחלק היבש
1. (2 נק׳) תארו את היוריסטיקה שקבעתם עבור שחקן ה-Minimax, הסבירו.
2. (2 נק׳) הסבירו את אופן פעילותו של שחקן התחרות שלכם. אם הגדרתם פונקציה היוריסטית חדשה, הסבירו עליה.
3. (2 נק׳) תארו איך ניהלתם את זמן ריצת הפונקציה make_move. עבור שני מקרי הגבלת הזמן.
(זמן מוגבל לתור וזמן מוגבל גלובלי).

חלק ו׳ – ביצוע ניסויים

הוסיפו את תשובותיכם לסעיפים הבאים לדוח:

4. (1 נק׳) הריצו משחקים בין סוכן ה-Minimax לסוכן ה-AlphaBeta (בעלי מגבלת זמן לתור) שמימשתם כאשר תתנו כל פעם זמן הגבלת תור שונה. מה התוצאות שקיבלתם? האם התוצאות מתאימות לציפיותיכם?

השוואות עומק בין שחקני שונים:

5. (4 נק׳) הגדירו שני שחקנים חדשים, LightABPlayer.py, ו- HeavyABPlayer.py, ללא התייחסות למגבלות זמן, ועליהם לחפש לעומק קבוע. היוריסטיקות של השחקנים יוגדרו באופן הבא:

- HeavyABplayer יוגדר להיות שחקן עם היוריסטיקה מתוחכמת (ניתן להשתמש ביוריסטיקה שמימשתם לשחקן שלכם)
- LightABPlayer יוגדר להיות שחקן עם היוריסטיקה פשוטה.

עליכם לבצע את הניסוי הבא:

על השחקן HeavyABplayer להיות מוגדר לעומק 3 באופן קבוע, כאשר עומק החיפוש בשחקן LightABPlayer משתנה בכל שלב בניסוי.

ישנם 3 שלבים, כאשר בשלב ה- i עומק חיפוש השחקן LightABPlayer יהיה $i - 1 + search\_depth(HeavyPlayer)$. כלומר בשלב הראשון עומק חיפוש של שלוש.
בכל שלב בניסוי ציון השלב יוגדר להיות מספר ניצחונות של HeavyABplayer.

יש להציג את תוצאות הניסוי בגרף, כאשר ציר ה-x מוגדר להיות ההפרש בין עומקי החיפוש של LightABPlayer ו-HeavyABplayer וציר ה-y מוגדר להיות ציון השלב.

יש לבצע את הניסוי פעם נוספת עבור חיפוש בעומק 2 של HeavyABplayer (כעת LightABPlayer יחפש לעומקים 2,3,4)

הסבירו את תוצאות הניסוי, האם יש שוני בין תוצאות שני הניסויים? אם כן מה לדעתכם יכולה להיות סיבה לכך?

הוראות הגשה

יש להגיש את כל הקבצים(קוד ודו״ח) בקובץ zip יחיד ששמו AI2_<id1>_<id2> כאשר במקום <id1> ו-<id2> יש לרשום את מספרי תעודת הזהות של המגישים.
מלבד לדו״ח, קובץ ה-zip צריך את כל קבצי הקוד של השחקנים שאתם כתבם(כל השחקנים מלבד LivePlayer.py, SimplePlayer.py, AbstractPlayer.py), וקבצי הקוד SearchAlgos.py ו- utils.py.
כאשר היררכיית התיקיות קיימת בקובץ ה-zip (השחקנים בתיקיית players, והקבצים SearchAlgos.py ו- utils.py מחוץ לתיקייה players