דו"ח מסכם

תרגיל בית 2

יונתן בתן

302279138

Yonibettan@gmail.com

נדב אליהו

303086854

Neliahu@gamil.com

חלק א':

מטעמי אסתטיקה, הודבקו התוצאות עבור ריצת התכנית ללא הדפסות.

```
(py3) ybettan@localhost hw2 (devel) $ python run_game.py 2 10 5 n simple_player random_player
The winner is O random
(py3) ybettan@localhost hw2 (devel) $ python run_game.py 2 10 5 n simple_player random_player
The winner is X simple
(py3) ybettan@localhost hw2 (devel) $ python run_game.py 2 10 5 n simple_player random_player
The winner is X simple
(py3) ybettan@localhost hw2 (devel) $ python run_game.py 2 10 5 n random_player simple_player
The winner is O simple
(py3) ybettan@localhost hw2 (devel) $ python run_game.py 2 10 5 n random_player simple_player
The winner is O simple
(py3) ybettan@localhost hw2 (devel) $ python run_game.py 2 10 5 n random_player simple_player
The winner is O simple
(py3) ybettan@localhost hw2 (devel) $
```

לפי התכנית השחקן שמוכנס ראשון (מבין השניים) הינו X.

ניתן להבחין כי ב- 3 ההרצות הראשונות כיוון ש- simple הוא הראשון שהוכנס אזי הוא מקבל את התווית X ו-random את התווית O.

ב-3 הריצות האחרונות נראה כי simple הוא שקיבל את התווית O

השחקן X תמיד משחק ראשון ולכן לצורך ההרצות פשוט שינינו את סדר הפרמטרים בשורת הפקודה.

הפלט מייצג את השחקן המנצח בריצה הספציפית.

התוצאות הינן שונות בין הריצות מכיוון שהשחקן random תמיד מגריל מצב אקראי מבין המצבים האפשריים העומדים לרשותו.

חלק ב':

התשובות לא לפי סדר הסעיפים לצורך נוחות ההסבר אך הם עונים על כולם.

אנו נבחר יוריסטיקה אשר לה 2 מרכיבים עיקריים:

- **תנועה:**
  - בוחנת לכל משתתף את מספר המהלכים האפשריים
  - בודקת מתוכם כמה מהלכים הם בעלי פוטנציאל לכל משתתף, כלומר משבצות אשר בקרבתן יש חייל יריב אחד לפחות

- **ניקוד:**
  - מבוסס על מספר החיילים שיש לכל משתתף
  - ניתן דגש על פינות ופאות
  - מתוך הפאות ניתן דגש על החיילים בפאות שהינם במצב "יציב" כלומר לא ניתנים להפיכה (למשל חלק משורה המסתיימת בפינה)

ולבסוף נשלב את 2 מרכיבים אלו בעזרת משקלים שונים אשר משתנים לפי שלבי התכנית, כלומר בהתחלה ניתן דגש גדול יותר על תנועה שגם התנועה משמעותית בהתחלה וגם אנחנו רחוקים מהפינות ובשלב כלשהו נתחיל להוריד את המשקל של התנועה לטובת הניקוד שכן אנו מתקרבים לסוף המשחק והתנועה יורדת משמעותית ל-2 השחקנים.

עבור מצב סופי היוריסטיקה תחזיר את הפרש מספר החיילים שלנו ושל היריב.

אנחנו אכן צופים שיפור משמעותי בתוצאות שכן היוריסטיקה הינה מיודעת יותר.

בלוח 8X8 כמו אצלנו יש 64 משבצות שמתוכם 4 כבר תפוסות במצב התחלתי, מה שמשאיר אותנו עם 60 מהלכים

מכיוון שבשלב זה אנחנו משנים רק את היוריסטיקה (כל השאר זהה ל- simple player) ולא פשוט להתחיל לספור באיזה מהלך אנחנו מבלי לשנות את נתוני המחלקה, אנו ניתן פקטור זהה של 1 עבור כל 60 המהלכים בשלב זה.

עבור שחקן התחרות ניתן משקלים כדלהלן:

- ב- 20 המהלכים הראשונים ניתן פקטור 1.5 לטובת התנועה ו-0.5 לטובת הניקוד
- ב- 20 המהלכים האמצעיים ניתן פקטור 1 לשני המרכיבים
- ב- 20 המהלכים האחרונים ניתן פקטור 0.5 לטובת התנועה ו-1.5 לטובת הניקוד

ועכשיו קצת נוסחאות:

<u>ניקוד:</u>

$$\begin{aligned} result = &(\#myCorners - \#oponentCorners)\cdot CORNER\_FAC + \\ &(\#myStableSides - \#oponentStableSides)\cdot STABLE\_EDGE\_FAC + \\ &(\#mySides - \#oponentSides)\cdot EDGE\_FAC + \\ &(\#myInter - \#oponentInter)\cdot INTER\_FAC \end{aligned}$$

בשלב זה ניתן את הערך הנ"ל לקבועים אך יתכן ונשנה אותם בעתיד לצורך שיפור ביצועים לתחרות

$$CORNER\_FAC = 100$$
$$STABLE\_EDGE\_FAC = 20$$
$$EDGE\_FAC = 5$$
$$INTER\_FAC = 1$$

<u>תנועה:</u>

$$result = (\#myLegalMoves - \#oponentLegalMoves)\cdot MOBILITY\_FAC$$

בשלב זה ניתן את הערך הנ"ל לקבועים אך יתכן ונשנה אותם בעתיד לצורך שיפור ביצועים לתחרות

$$MOBILITY\_FAC = 1$$

<u>מצב סופי:</u>

עבור מצב סופי נחזיר את $result = \#myInter - \#oponentInter$

להלן התוצאות:

```
(py3) ybettan@localhost hw2 (devel) $ python run_game.py 2 10 5 n AI2_302279138_303086854.better_player simple_player
The winner is X better
(py3) ybettan@localhost hw2 (devel) $ python run_game.py 2 10 5 n AI2_302279138_303086854.better_player simple_player
The winner is X better
(py3) ybettan@localhost hw2 (devel) $ python run_game.py 2 10 5 n AI2_302279138_303086854.better_player simple_player
The winner is X better
(py3) ybettan@localhost hw2 (devel) $ python run_game.py 2 10 5 n simple_player AI2_302279
138_303086854.better_player
The winner is O better
(py3) ybettan@localhost hw2 (devel) $ python run_game.py 2 10 5 n simple_player AI2_302279138_303086854.better_player
The winner is O better
(py3) ybettan@localhost hw2 (devel) $ python run_game.py 2 10 5 n simple_player AI2_302279138_303086854.better_player
The winner is O better
(py3) ybettan@localhost hw2 (devel) $
```

ניתן לראות שבכל 6 הריצות הסוכן better_player שמימשנו מנצח את הסוכן simple_player

חלק ג':

- מומש בקוד
- השתמשנו ביוריסטיקה של simple_player על מנת לא ליצור תלות בין החלקים השונים של התרגיל

חלק ד':

- מומש בקוד
- ברור שהאלגוריתם מינימקס ומינימקס עם גיזום אלפא-בטא מחזיר את אותו ערך, הרי הגיזום אמור לייעל את זמן החיפוש אך לא אמור לפגוע בנכונות של אלגוריתם ללא גיזום.
  ההבדל נובע מהעמקה הדרגתית, כלומר מכיוון שלא תמיד יש לנו מספיק משאבים כדי להריץ את האלגוריתם ללא גיזום בשלמותו לפעמים נצטרך לעצור אחרי עומק מסוים ולכן להתפשר על ערך יוריסטי.
  בעקבות כך ככל שהעומק יהיה גדול יותר היוריסטיקה שלנו תהיה מיודעת יותר ולכן במידה ונגזום ענפים נוכל במקרה הטוב לחפש לעומק פי 2 יותר גדול במקרה הטוב.

  לכן נצפה לקבל ביצועים טובים יותר עבור השחקן אלפא-בטא בתנאי שהזמן המוקצב לכל שחקן הינו קטן מספיק בכדי לאלץ אותו להתפשר ולא להצליח להגיע לכל עץ החיפוש ובנוסף צריך להתקיים שסדר הבנים בעץ החיפוש מאפשר גיזום (במקרה הגרוע אלפא-בטא לא יגזום שום ענף ונקבל ביצועים זהים).

חלק ה':

סעיף 1:

התשובה לשאלה זו אינה חד משמעית, שכן בשלב זה עוד לא ביצענו fine tuning , ונגלה אותה בהמשך ולכן ניתן קווים מנחים והסברים לכמה תוצאות אפשריות.

- בהנחה והענקנו משקלים "טובים" למרכיבים של הפונקציה היוריסטית של better נצפה לביצועים טובים מאוד ממנו שכן הוא הכי מיודע ולמרות שהוא חושב רק צעד אחד קדימה אי הדיוק של היוריסטיקה של שאר השחקנים עלולה להביא לתוצאות בינוניות למרות אלגוריתמי החיפוש
- במידה ופספסנו במעט את המשקלים של היוריסטיקה של better נקבל יתרון לטובת min_max and alpha_beta עם יתרון ל- alpha_beta למרות שיתכן שהמצב לא יהיה כך בעקבות יוריסטיקה לא מדויקת בכלל.

סעיף 2:

א. עבור תוספת של selective_deepening היה ניתן לתת ערך "יציבות" עבור כל מצב ולהחליט להעמיק את אלגוריתם החיפוש רק עבור צמתים "לא יציבים", שכן העמקה בצמתים יציבים לא מוסיפה הרבה תועלת אך נחסוך בזמן חיפוש לטובת מצבים אשר קיים בהם פוטנציאל לשיפור.
לכן ברמת האלגוריתם עבור כל ריצה (עומק מקסימלי כלשהו) , ברגע שהגענו לעומק המקסימלי מבצעים העמקה סלקטיבית על כל הצמתים ה- "לא יציבים" כאשר העומק של ההעמקה הנוספת נקבע באופן דינמי ביחס לכמות הצמתים הלא יציבים (ככל שיש פחות לא יציבים נאפשר העמקה עמוקה יותר) בכדי לא לפגוע בזמן הריצה של האלגוריתם בצורה משמעותית.

ב. לא ניתן לקבל שיפור משמעותי שכן זה בדיוק מה שקורה גם ללא הפונקציה הזאת.
בסוף כל תור כל שחקן מחשב את הזמן הנותר לו עד סוף רצף המהלכים ומחסיר 1 ממספר המהלכים הנותרים.
כלומר אם המהלך הראשון לדוגמא מתוך 5 התבצע מאוד מהר (יותר מהר מחמישית מהזמן שהוקצה ל-5 מהלכים) , אותו הפרש מתחלק בצורה שווה לכל המהלכים הנותרים.

חלק ו':

סעיף 1:

- שימוש בטבלת ערבול המכילה מהלכים וחסמים על התוצאה – מאפשר גיזום אלפא-בטא מהיר יותר בחלק מהמקרים
- ניצול זמן החשיבה של היריב לחישובים
- שימוש ב- opening book עליו נרחיב בסעיף הבא

סעיף 2:

1. Opening book הוא קובץ אשר מאכסן בתוכו עבור כל ריצת משחק את רצף המהלכים שנעשו והתוצאה הסופית.
ע"י קובץ זה ניתן לדרג "מהלכים חמים" כלומר לראות מהלכים שחוזרים על עצמם לעיטים קרובות (כלומר נבחרו ע"י האלגוריתם פעמים רבות) ולכן ניתן להסיק מכך שהמהלך הינו מהלך טוב ולכן נשתמש בו במקום לחשבו מחדש ע"י האלגוריתם ונרוויח זמן חישוב למטרות אחרות – כמו העמקה גדולה יותר בחיפוש עבור מהלכים שיש לנו עליהם פחות מידע.

2. להלן 5 המשחקים הנפוצים ביותר, כאשר המספר בעמודה משמאל מייצג את מספר ההופעות של משחק זה בקובץ המקורי.

```
ybettan@localhost hw2 (devel) $
ybettan@localhost hw2 (devel) $ cat book.gam | cut -c-30 | sort | uniq -c | sort -nr | head -n5
  13493 +d3-c5+f6-f5+e6-e3+c3-f3+c4-b4
  13493 +d3-c5+f6-e3+c3-f5+e6-f3+c4-b4
  13493 +d3-c5+e6-f5+f6-e3+c3-f3+c4-b4
   7432 +d3-c5+f6-f5+e6-e3+d6-f7+g6-e7
   7432 +d3-c5+e6-f5+f6-e3+d6-f7+g6-e7
ybettan@localhost hw2 (devel) $
ybettan@localhost hw2 (devel) $
ybettan@localhost hw2 (devel) $
```

3. קוד

4. החיסרון העיקרי בבנייה הנוכחית הוא שעלינו לחלץ מתוך state נתון את המהלך שהיריב ביצע על מנת לתחזק את ה- prefix של $n-1$ מהלכים קודמים מה שמשפיע על זמן הריצה של שחקן.

5. דרך חלופית, אשר בה לא נשתמש כרגע, היא ליצור מילון מתוך הקובץ בצורה כזאת שעבור כל מהלך מייצרים state ואת המילון בונים בצורה כזאת שה- key הינו state וה- value המוחזר הינו המהלך הבא (move). דרך זו מכבידה על הבנאי אך משפרת את זמן הריצה.

ה- design שאנחנו בחרנו מתחזק מחרוזת, בפורמט של ה- opening book, של המהלכים שבוצעו עד כה כך שהמהלך של היריב מחולץ ע"י חישוב ה- diff בין המצב הנוכחי למצב האחרון שבוצע.
בכל בחירת מהלך נשתמש במחרוזת זאת כ- key ובמידה והנ"ל קיים במילון נבחר על פיה את המהלך הבא אחרת המצב הבא יבחר לפי הפונקציה היוריסטית כפי שהיה קודם לכן.

מכיוון שהמהלך הראשון ב- opening book הינו תמיד d3 והנ"ל אינו מהלך חוקי עבור השחקן X שאצלנו תמיד משחק ראשון, הוגדרו 4 זוגות של פונקציות המתאימות את משבצות הלוח ל- opening book, book2reality_i and reality2book_i עבור $1 \le i \le 4$.
במידה והשחקן שלנו משחק ראשון אזי תמיד בוחרים את הגרסה הראשונה של הפונקציות אחרת מחכים שהיריב יבצע את המהלך הראשון ולפי המשבצת אותה בחר מתאימים את פונקציית המיפוי (שתתאים את משבצת היריב ל- d3).

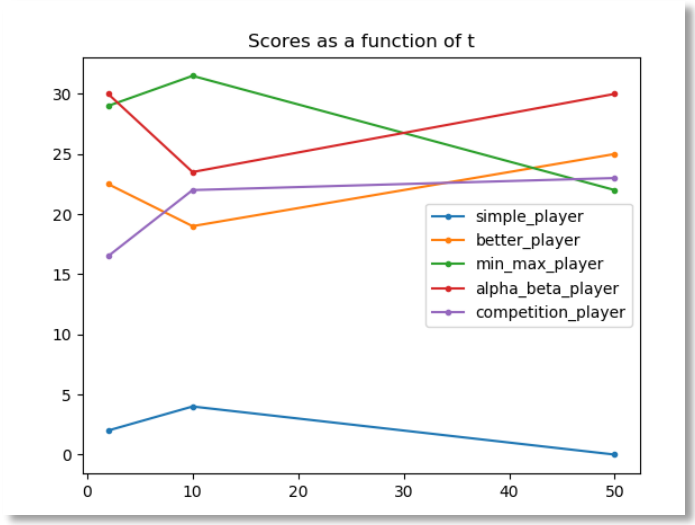
חלק ז'

סעיף 1:

צורף הקובץ experiments.csv לקבצי ההגשה.

סעיף 2:

	A	B	C	D
1	t = 2	t = 10	t = 50	player_name
2	2	4	0	simple_player
3	22.5	19	25	better_player
4	29	31.5	22	min_max_player
5	30	23.5	30	alpha_beta_player
6	16.5	22	23	competition_player



- ניתן להתעלם מ- competition_player

נתחיל בניתוח התוצאות.

- נציין ש- simple, min_max_ and alpha_beta פועלות כולן עם אותה יוריסטיקה בסיסית של simple, אך לעומתם ל- better יש יוריסטיקה חכמה יותר ובנוסף מתחזקת ספר פתיחות.
- התוצאות ברובן תואמות את הצפיות למעט העובדה שנראה ש- min_max משחק טוב יותר מ-alpha_beta, דבר שלכאורה מנוגד לאינסטינקט הראשוני שלנו.

Simple:

ניתן להבחין כי השחקן הנ"ל משחק בצורה הכי פחות טובה מבין השחקנים בכל אחד מהזמנים האפשריים, מה שתואם לחלוטין לצפיות שלנו.

better משתמש ביוריסטיקה מיודעת יותר ו- opening book אשר מקנים לו יתרון ב- 2 רבדים.

בניגוד ל- better, min_max ו- alpha_beta פועלים ללא ספר פתיחות ועם יוריסטיקה זהה אך מה שמקנה להם יתרון הינו אלגוריתם החיפוש החכם שלהם שמאפשר להם לראות מספר צעדים קדימה

בנוסף ניתן להבחין שההצלחה המעוטה של simple אף נהית פחות טובה ככל שמגדילים את הזמן המוקצה לכל מהלך, את הדעיכה הזאת ניתן בעיקר לקשר לעצם העובדה ש- better ו- alpha_betta משתפרים כאשר יש להם יותר זמן לחישוב ולאו דווקא בגלל מגבלת זמן על simple

Better:

ניתן לראות כי better משחק טוב יותר מ- simple אך נמצא בין min_max ל- alpha_beta עבור זמני חישוב גדולים ופחות טוב מהם עבור זמני חישוב קטנים.

כאשר הזמן המוקצה הינו 2 או 10 שניות למהלך נבחין כי min_max ו- alpha_beta גוברים על better וזאת מכיוון ששחקנים אלו צופים מספר מהלכים קדימה לעומת better שרואה צעד יחיד קדימה בכל שלב ולבסוף עבור זמן ארוך לכל מהלך היוריסטיקה החכמה שבחרנו גוברת על min_max אך עדיין פחות טובה מ- alpha_beta.

Min_max & alpha_beta:

הנקודה הקריטית עבור שחקנים אלו מעבר לכל מה שכבר נאמר עליהם הינה ש- min_max משחק טוב יותר מ- alpha_beta עבור 2 ו-10 שניות, דבר שלכאורה לא הגיוני שכן ראינו בהרצאה ש- alpha_beta אינו פוגע באופטימליות של min_max.

הסיבה המרכזית לכך היא ש- alpha_beta נותן פתרון מיודע יותר, כלומר מחפש עמוק יותר בכל איטרציה ולכן במידה והיורסטיקה אינה נותנת אומדן איכותי למצב המשחק, פער זה רק יגדל עם עומק החיפוש.

עבור זמן חיפוש מספיק גדול alpha_beta יתחיל לחשב צעדים קדימה כמעט עד סוף המשחק ולכן עבור היוריסטיקה שלו הסופרת את מספר החיילים מכל צבע נקבל אומדן דיי קרוב לתוצאה האמתית שכן במצב סופי כך נבדקת התוצאה.

לכן עבור זמן חיפוש ארוך הפערים בין alpha_beta ל- min_max מצטמצמים ואפילו ניתן יתרון ל- alpha_beta.

לסיכום, עבור זמן חישוב קצר לא מספיקים לחשב מספר צעדים גדול קדימה ולכן בעקבות אי דיוק היוריסטיקה מחמיר הפער בין alpha_beta ל- min_max עד אשר זמן החישוב מספיק גדול ($t=10$) בכדי שעבור חיפוש עמוק מספיק היוריסטיקה תהיה מדויקת מספיק על מנת לצמצם את הפערים ואף לאפשר לאלפא בטא לגבור על מינימקס