מבוא לבינה מלאכותית

תרגיל בית 2

מגישים:

אביב כספי 311136691

יקיר יהודה 205710528

חלק ב'

2. השחקן GreedyAgent עובד בצורה הבאה:

השחקן עובר על כל אפשרויות ההתקדמות שלו (צעד אחד), ומבצע בדיקה עבור כל פעולה. כאשר בהינתן הפעולה אותה בודק השחקן, השחקן בודק את כל האופציות לפעולות של אויביו, כאשר עבור כל אופציה של פעולות, הוא בונה מצב חדש ובודק בעזרת היוריסטיקה (שתוסבר בהמשך) , איזה מהמצבים מחזיר ערך מקסימלי. עבור מצב מקסימלי זה השחקן יבחר לבצע את הפעולה שהביאה אותו לאותו מצב. (עבור יותר משני אוייבים, השחקן יפתח רק מצב אחד עבור פעולות אויביו, בשביל לחסוך בזמן וזיכרון).

היוריסטיקה שהשחקן GreedyAgent משתמש:

אם הוא מת, מחזיר את האורך שלו.

אם הוא חי, בודק כמה פירות נשארו לאכול (פירות במסך + אורך הנחשים האויבים החיים) ומחזיר את אורך הנחש + תוספת עבור מספר הפירות שנשארו ביחס לכמות התורות שנשארו (היוריסטיקה מניחה כי בכל צעד יאכל הנחש תפוח (אם קיים) ונותנת פקטור ודאות עבור כל תפוח).

השחקן נמנע ממוות על ידי כך שנותן ציון יוריסטי נמוך למצב בו הוא מת ביחס למצב בו הוא חי, לדוגמא, אם במצב שנבדק השחקן מת, הציון שיקבל המצב יהיה אורך השחקן, ואם במצב הבא השחקן נשאר חי, בהכרח המצב יקבל ציון גדול שווה לאורך הנחש (אם יש תפוחים או אויב אזי יקבל ציון גבוה יותר).

חלק ג'

1. **יוריסטיקה :** 1- אורך הנחש, 2- מרחק לתפוח הקרוב ביותר, 3- שטח שניתן להגיע אליו לאחר ביצוע הפעולה, 4- מרחק מראש הנחש האויב
הציון למצב הניתן על ידי היוריסטיקה יחושב לפי הנוסחא הבאה :
$$h(state) = snake.length + \frac{1}{apple\ dist} + 2 * 1_{reachable>snake.length+5} + 1_{enemy-dist>2}$$
כלומר היוריסטיקה מחושבת על ידי אורך הנחש + 1 חלקי המרחק לתפוח הקרוב ביותר, + 2 אם השטח שניתן להגיע אליו גדול מאורך הנחש פלוס 5 + 1 אם המרחק מהאויב הקרוב ביותר גדול מ 2.
בנוסף הגדרנו כי אם הנחש מת הערך שיוחזר מהיוריסטיקה יהיה אורכו של הנחש.
2. אורך הנחש – נועד לתת מוטיבציה לנחש להגדיל את אורכו, ככל שהאורך גדל כך גם הציון שיקבל המצב.
1 חלק מרחק לתפוח קרוב – נועד לתת מוטיבציה לנחש לבצע צעד שיקרב אותו לתפוח הקרוב ביותר.
שטח שניתן להגיע אליו – נועד למנוע מהנחש לעשות צעדים לכיוון תפוח הנמצא במקום חסום, אם הנחש יתקדם לכיוון שלא מקיים את התנאי (שטח שניתן להגיע אליו גדול ב5 לפחות מאורך הנחש) הוא יאבד 2 נק' ביוריסטיקה (יעדיף להתקדם לכיוון אחר אפילו אם התקדמות זו תאריך את אורכו).
מרחק מנחש אויב – נועד למנוע מהנחש שלנו להתקרב יותר מידי לראש הנחש האויב, מפני שאנחנו לא יכולים לצפות איזה מהלכים יבצע האויב.
אנחנו צופים כי הגדרות אלה ישפרו את פעילות הנחש הבסיסי, מפני שבכל צעד הנחש ינסה להגדיל את אורכו, להתקרב לתפוח כלשהו, ולהימנע מלחסום את עצמו ולגרום למוות ובנוסף ישמור מרחק מהנחש האויב כדי לא להיפגע ממנו.

בנוסף העובדה שהגדרנו את הערך המוחזר במקרה של מוות להיות קטן מכל ערך אחר שייתכן אם הנחש לא מת, הבטחנו כי אם יש לנחש אפשרות להישאר בחיים הוא יבחר באפשרות זו, ובכך ימנע ממוות.

חלק ד'

2. תחילה אנחנו מצפים שיעבוד יותר טוב מGREEDY בגלל השימוש בהיוריסטיקה שלנו אשר ממפה מצבים בצורה טובה יותר וגם בגלל שכעת אנחנו מסתכלים לעומק עמוק יותר ויכולים לזהות בעיות בצעדים הבאים.

האלגוריתם יעבוד טוב עבור מקרים בהם ייתכן כי נתקל במכשול (סוף הלוח או נחש אחר) במרחק שגדול מצעד יחיד, האלגוריתם יזהה את הבעיה (כל עוד המכשול בעומק בו מחפשים) וימנע מלהתקדם במסלול שיוביל לבעיה.

מצד שני האלגוריתם מצפה לגרוע מכל מיריביו, כלומר נניח כי יש תפוח קרוב לנחש שלנו, אך יש גם נחש יריב קרוב, ייתכן כי כאשר האלגוריתם יפתח את המצבים הבאים, הוא יניח כי הנחש האויב יתקדם לתפוח קודם, או יחסום את הנחש שלנו ולכן האלגוריתם ימנע מהנחש מלהתקרב לתפוח למרות שמצב זה אינו סביר.

3. כאשר כל היריבים בחיים, לכל צומת מינימום יהיה 3^k בנים, כאשר כל בן מסמל בחירה של פעולה עבור כל אחד מהאויבים (לכל אויב יש 3 אפשרויות פעולה).
עבור k=20 מספר הבנים לכל צומת מינימום יהיה $3^{20} = 3{,}486{,}784{,}401$ בנים, כלומר על מנת לחשב את תוצאת האלגוריתם יהיה עלינו לעבור על כל הבנים של כל צומת מינימום בעץ המצבים כלומר אפילו עבור מצב בו נסתכל בעומק 1 החיפוש יקח יותר מידי זמן ומשאבים, לכן בפועל לא ניתן להשתמש באלוגריתם במצב זה.
4. כאשר יש יותר מיריב אחד, אנחנו מניחים כי כל אחד מהיריבים, יבצע פעולה שהכי פחות טובה לשחקן שלנו (בחירת ציון היוריסטי מינימלי – כלומר להביא למינימום את הרווח של השחקן שלנו) ,בנוסף אנחנו מניחים כי כל שחקן משחק בתורו ויחסית לפעולות של השחקנים לפניו. במשחק שלנו ההנחה הזאת אינה נכונה, מפני שאצלנו צומת מינימום יחיד שייך לכל היריבים יחדיו. כלומר במקרה שלנו ההנחה היא כי כל היריבים משחקים יחדיו נגדנו, וכי כל האויבים משחקים בתור יחיד יחסית לפעולה שאנחנו עשינו (ייתכן כי כאשר נפריד את השחקנים לצמתים שונות, הפעולות שהאלגוריתם שלנו מניח שהם יבצעו יהיו שונות מאשר הפעולות שהאלגוריתם הנוכחי מניח).
5. *a.* במקרה בו נרצה לממש שכבה נפרדת לכל אויב, נפרק את המצבים שהיו לנו עכשיו, כאשר לכל אויב תהיה שכבה בעץ המצבים, כעת כאשר נעבור על עץ המצבים, כל צומת יריב הינה צומת מינימום, כלומר נחפש את הערך המינימלי של בני כל צומת כזו.
חסרונות מימוש זה : זיכרון נוסף עבור כל מצב פנימי בין צומת המקסימום לבנים האחרונים של צמתי המינימום. בין כל שני צמתי מקסימום יהיו לנו עכשיו *k* צמתי מינימום עבור כל אחד מהיריבים לעומת צומת מינימום אחת במימוש הקודם, הוספה של $\frac{(3^k-3)*3}{2}$ צמתים.
יתרונות : מבחינת מימוש, עבור כל צומת מינימום נעבור רק על אפשרויות הפעולה של יריב זה (כלומר 3 פעולות אפשריות) לעומת המימוש הנוכחי, שדורש לעבור על כל אפשרויות היריבים במקביל ולהחזיר מצב יחיד.

b. ההנחה במקרה זה היא כי כל יריב פועל בתורו, ביחס לפעולות הקודמות של השחקנים שלפניו. בנוסף בגלל מימוש *MINIMAX*, ההנחה היא כי כל יריב בוחר את הפעולה הרעה ביותר עבור השחקן שלנו. במימוש הקודם, כל היריבים שיחקו במקביל באותו תור, וכולם עבדו יחדיו כנגדנו, כלומר יחסית לכל הצעדים האפשריים לשחקנים, האלגוריתם הניח כי יבחר סידור הפעולות שיביא את הערך שהשחקן שלנו יקבל למינימום.(במקרה זה אין תלות בסידור השחקנים כי כולם משחקים יחדיו, לעומת המצב הקודם בו ייתכן כי אם נחליף את סדר השחקנים האלגוריתם ימצא פעולות שיביאו את הערך שנקבל לנמוך יותר), (בנוסף, נציין רק כי

במשחק שלנו, אין באמת משמעות לתורות, בגלל שכל הצעדים של השחקנים נעשים במקביל בלי תלות).

חלק ה'

2. *a*. מבחינת זמן ריצה, הסוכן אלפא בטא ישתפר ברוב המקרים, בגלל קיצוץ של מסלולים שלא יביאו לשיפור, כמו שראינו בהרצאה יתכנו מצבים בו האלגוריתם יפתח את כל הצמתים כמו אלגוריתם *Minimax*, אך מקרה זה לא נפוץ.

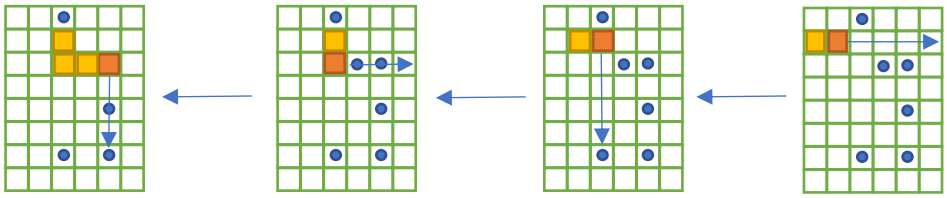
b. הסוכן יפעל בצורה זהה לסוכן *minimax* , מפני שבחירת המהלכים זהה במהלך האלגוריתם וההבדל היחיד הוא ויתור על פיתוח מהלכים שלא ישפרו, כלומר לא ייתכן כי סוכן אלפא בטא יבחר לבצע פעולה שונה מסוכן *minimax*.

חלק ו'

1. מניחים שהמשחק קבוע ואינו משתנה מפני שהאלגוריתם שלנו מנסה לשפר את המצב הבא יחסית ללוח הנוכחי, כלומר אם כל איטרציה נשנה את הלוח, האלגוריתם ינסה להשתפר עבור הלוח הנוכחי, אך שיפור זה לא יהיה רלוונטי עבור הלוח הבא. (הכוונה בלוח היא לכל הנתונים כולל מיקום הנחשים והתפוחים). לדוגמא נניח כי באיטרציה כלשהי האלגוריתם מצא כי אם הנחש יבצע תנועה ימינה, ישתפר הציון הסופי שלו, אך כעת באיטרציה הבאה, הלוח משתנה וכעת ביצוע תנועה ימינה יגרום להתנגשות בנחש אחר וציון סופי נמוך יותר. אך בגלל הגדרת האלגוריתם, באיטרציות הבאות לא נשנה את הבחירה לבצע צעד ימינה, כלומר אין משמעות לבחירת פעולה באיטרציה כלשהי אם באיטרציה הבאה משתנה מצב הלוח.
2. המשחק אינו סכום אפס לפי הגדרת התרגיל, נשים לב כי פונקציית התועלת במשחק שלנו היא כך שכל שחקן מקבל ניקוד השווה לאורכו + 1 אם השחקן חי, כלומר ניקוד כל שחקן יהיה תמיד חיובי (אורך נחש התחלתי הוא 2) ולכן אם נסכום את התועלת של כל השחקנים בהכרח נקבל ערך חיובי, כלומר משחק זה אינו סכום אפס.
3. מצבים במרחק החיפוש שלנו:
$$\boldsymbol{states} = \{(\boldsymbol{a_1}, \boldsymbol{a_2}, \dots, \boldsymbol{a_N}) | \forall \boldsymbol{i}, \boldsymbol{a_i} \in \boldsymbol{GameAction}, \boldsymbol{N} = \boldsymbol{Number\ Of\ Steps}\}$$
הסבר: כל מצב מוגדר להיות וקטור של *N* פעולות משחק, כאשר פעולה במיקום ה- *i* , הינה הפעולה אותה יבצע הסוכן בתור ה- *i* של המשחק.
4. אופרטור במרחב החיפוש הם:
נגדיר 3 אופרטורים, כאשר כל אחד מהם יכול לפעול על *N* אינדקסים שונים במערך:
$$O^i_{LEFT}\big((a_1, a_2, \dots, a_i, \dots, a_N)\big) = (a_1, a_2, \dots, LEFT, \dots, a_N)$$
$$O^i_{STRAIGHT}\big((a_1, a_2, \dots, a_i, \dots, a_N)\big) = (a_1, a_2, \dots, STRAIGHT, \dots, a_N)$$
$$O^i_{RIGHT}\big((a_1, a_2, \dots, a_i, \dots, a_N)\big) = (a_1, a_2, \dots, RIGHT, \dots, a_N)$$
כאשר $i \in \{1, \dots, N\}$
הסבר: כל אופרטור מקבל אינדקס *i* , ופועל על מצב מסויים, כאשר עבור מצב זה הוא משנה, את הפעולה ה*i* לפעולה המתאימה לאופרטור.
5. מספר האיטרציות שנבצע בחיפוש זה הינו כגודל וקטור הפעולות, במקרה שלנו *N* פעולות.
6. תחילה נציין כי פונק' היוריסטיקה פועלת על מצבים במרחב החיפוש, לכן הפונקציה היוריסטית מצפה לקבל מצב מהצורה $(\boldsymbol{a_1}, \boldsymbol{a_2}, \dots, \boldsymbol{a_N})$ כאשר כל איבר מסמל פעולה שתתבצע בצעד המתאים.
נרצה להגדיר פונקציה שתעריך בצורה הטובה ביותר את איכות המצב הנבדק, כלומר יהיה

עלינו להגדיר מדדים, אשר מגדירים את איכות המצב.
במקרה שלנו מדובר במשחק מסוג סנייק, לכן מדדים אפשרים יהיו אורך הנחש בסיום המשחק כאשר משתמשים במצב כסדרת הפעולות, האם הנחש חי בסוף הריצה, האם הנחש ניצח בסוף הריצה, האם הנחשים האויבים חיים בסוף הריצה ואם כן מה האורכים שלהם ביחס לאורך של הנחש שלנו.
נגדיר פונקציה שתתייחס לכל המדדים הללו, ותיתן ציון למצב יחסית לחשיבות כל אחד מהמדדים.
הפונקציה תעבוד בכך שתריץ משחק של סנייק עם רצף הפעולות הנתון, ובסיום המשחק תסכום את המדדים בצורה ממושקלת (לדוג' נתן לאורך הנחש משקל גבוה יותר מאשר מספר האויבים שחיים בגלל שהעדיפות שלו גבוה יותר מבחינת הגדרות המשחק).

7. בחרנו להתחיל ממצב בו כל הפעולות הן פעולות להתקדם ישר.
בחרנו במצב זה לאחר בדיקות של מצבים רבים, ובחירה במצב שהביא לנו תוצאות טובות ביותר.
עפ"י פונקציית התועלת שהוגדרה לתרגיל *get_fitness* , מצב זה נותן תוצאות טובות מפני שבכל איטרציה, הנחש שלנו בודק באיזה כיוון יש מספר מקסימלי של תפוחים במסלול ישר מהנקודה הנוכחית שלו, ובוחר להתקדם לשם וממשיך באותו כיוון עד אשר יגיע לנק' בה מספר התפוחים שיוכל לאכול בצדדים יגדל ממספר התפוחים שנותרו לו במסלול הישר.
נדגים את ההסבר בתמונות הבאות :



ניתן לראות בדוגמא, כי בכל פעם שהנחש מזהה מסלול טוב יותר בקו ישר, הוא ימשיך אליו.
נציין כי בחירת מצב התחלתי אקראי, הביא לפעמים גם לניקוד טוב בסיום האלגוריתם אך בחירת מצב התחלתי כמו שהגדרנו הביאה תוצאה עקבית טובה מאד.

8. כאשר מבצעים שינוי בפעולה ה-*i* משפיעים רק על המצבים שיבדקו לאחר פעולה זו.
מצבים שהיו לפני הפעולה נשארו כמו שהם, מפני שהמעבר מהמצב ההתחלתי עד למצב ה-*i* אינו משתנה.
לעומת זאת כל המצבים שיופיעו אחרי הפעולה ה-*i* יושפעו, מהסיבה שכל מצב תלוי בכל הצעדים הקודמים לו. מספר המצבים שיושפעו בשינוי זה הוא לכל היותר *(N-i)*3* כאשר *N* הוא מספר הפעולות הכולל, מפני שבכל איטרציה נפתח 3 מצבים ולאחר שינוי צעד *i* כל המצבים שיפותחו יהיו חדשים.

.10

```
[<GameAction.STRAIGHT: 1>, <GameAction.RIGHT: 2>,
<GameAction.STRAIGHT: 1>, <GameAction.RIGHT: 2>,
<GameAction.STRAIGHT: 1>, <GameAction.LEFT: 0>,
<GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>,
<GameAction.STRAIGHT: 1>, <GameAction.LEFT: 0>,
<GameAction.LEFT: 0>, <GameAction.STRAIGHT: 1>,
<GameAction.RIGHT: 2>, <GameAction.STRAIGHT: 1>,
<GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>,
<GameAction.LEFT: 0>, <GameAction.RIGHT: 2>,
<GameAction.RIGHT: 2>, <GameAction.STRAIGHT: 1>,
<GameAction.LEFT: 0>, <GameAction.LEFT: 0>,
<GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>,
<GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>,
<GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>,
<GameAction.STRAIGHT: 1>, <GameAction.RIGHT: 2>,
<GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>,
<GameAction.STRAIGHT: 1>, <GameAction.LEFT: 0>,
```

```
<GameAction.RIGHT: 2>, <GameAction.STRAIGHT: 1>,
<GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>,
<GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>,
<GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>,
<GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>,
<GameAction.STRAIGHT: 1>, <GameAction.STRAIGHT: 1>,
<GameAction.STRAIGHT: 1>, <GameAction.RIGHT: 2>,
<GameAction.LEFT: 0>, <GameAction.LEFT: 0>]
```

Winner: WinnerAtTurn(player_index=0, length=13)

הנחש שלנו הגיע לאורך 13, וניצח!

ניתן לראות כי אכן רוב הפעולות נשארו פעולות ישר, ובכל מצב הנחש בחר להתקדם למסלול הישר הטוב ביותר בשבילו.

11. בחרנו לממש עבור סעיף זה, את האלגוריתם הגנטי אשר בכל איטרציה, מריץ דור של נחשים (וקטור באורך N של פעולות). האלגוריתם מדרג את הנחשים לפי פונקציית התועלת שהוגדרה לנו, ויוצר נחשים "ילדים" חדשים. האלגוריתם בוחר בצורה הסתברותית מבין הנחשים, זוג הורים מהדור הנוכחי (זוג הורים יבחרו יחסית ל fitness שלהם), ומבצע הצטלבות בין זוג ההורים (בהסתברות שנגדיר להיות 0.5, אחרת מעתיק את ההורים כמו שהם), לאחר מכן מבצע מוטציה בוקטור הפעולות של כל אחד מהילדים בהסתברות נמוכה (אשר נגדיר להיות 0.05). כאשר נגדיר ביצוע מוטציה לפעולה ה-i , כשינוי אקראי של הפעולה ה-i בוקטור הפעולות של הילדים, לפעולה אקראית אחרת. לסיום אנחנו מוסיפים את כל הנחשים הילדים לדור חדש, ומריצים את האלגוריתם מחדש.

בחרנו באלגוריתם זה, מפני שאנחנו חושבים שכאשר יש לנו זוג נחשים שפועל בצורה טובה, איחוד הפעולות שלהם + מוטציה לחלק מהפעולות, יכול להניב נחש חדש, שיתגבר על המכשולים שכל אחד מהוריו נתקל בריצתו ובכך להשיג תוצאה טובה יותר.

שינויים שנבצע באלגוריתם :

במימוש ההצטלבות, בחרנו לבצע את ההצטלבות מנק' רנדומלית בוקטור הפעולות כאשר כל הפעולות לפני פעולה זו ילקחו מהורה אחד, וכל שאר הפעולות מההורה השני.

ביצענו שינוי זה, מפני שלדעתנו יש לשמור כמה שיותר על סדר הפעולות שביצע ההורה מבלי לערבב פעולות מההורה השני באמצע (פעולות שיופיעו באמצע הרצף יכולות לפגוע בהתקדמות הנכונה שהגיעה מההורה).

12. מצבים במרחב החיפוש זהים למצבים שהוגדרו בסעיף 3, כאשר כל מצב מגדיר וקטור של N פעולות שהנחש מבצע.

אופרטורים במרחב זה :

1. אופרטור ביצוע Crossover :

$$O^i_{cross}\big((a_1,\dots,a_N),(b_1,\dots,b_N),P_c\big) = \begin{cases} ((a_1,\dots,a_i,b_{i+1},\dots,b_N),(b_1,\dots,b_i,a_{i+1},\dots,a_N)), & with\ P_c\ probability \\ ((a_1,\dots,a_N),(b_1,\dots,b_N)), & with\ 1-P_c\ probability \end{cases}$$

הסבר : אופרטור זה פועל על 2 נחשים, כאשר בהסתברות P_c מבצע הצטלבות בין ההורים ויוצר שני ילדים חדשים כמתואר, אחרת יוצר 2 ילדים הזהים להורים.

2. אופרטור ביצוע *mutate* :

$$O^i_{mutate}\big((a_1,\dots,a_N),P_m\big) = \begin{cases} (a_1,\dots,b,\dots,a_N), & with\ P_m\ probability, b \in GameAction \\ (a_1,\dots,a_N), & with\ 1-P_m\ probability \end{cases}$$

הסבר: אופטור זה פועל על נחש יחיד כאשר בהתסברות P_m מבצע מוטציה לפעולה ה-i בוקטור הפעולות של הנחש, אחרת משאיר את הפעולה כמו שהי.

14. [<GameAction.LEFT: 0> <GameAction.LEFT: 0> <GameAction.RIGHT: 2>

<GameAction.RIGHT: 2> <GameAction.STRAIGHT: 1> <GameAction.STRAIGHT: 1>

<GameAction.RIGHT: 2> <GameAction.STRAIGHT: 1> <GameAction.LEFT: 0>

<GameAction.LEFT: 0> <GameAction.LEFT: 0> <GameAction.STRAIGHT: 1>

<GameAction.RIGHT: 2> <GameAction.LEFT: 0> <GameAction.STRAIGHT: 1>

<GameAction.RIGHT: 2> <GameAction.STRAIGHT: 1> <GameAction.STRAIGHT: 1>

<GameAction.RIGHT: 2> <GameAction.RIGHT: 2> <GameAction.STRAIGHT: 1>

<GameAction.LEFT: 0> <GameAction.RIGHT: 2> <GameAction.LEFT: 0>

<GameAction.RIGHT: 2> <GameAction.STRAIGHT: 1> <GameAction.STRAIGHT: 1>

<GameAction.STRAIGHT: 1> <GameAction.RIGHT: 2> <GameAction.RIGHT: 2>

<GameAction.RIGHT: 2> <GameAction.LEFT: 0> <GameAction.STRAIGHT: 1>

<GameAction.STRAIGHT: 1> <GameAction.STRAIGHT: 1> <GameAction.RIGHT: 2>

<GameAction.LEFT: 0> <GameAction.STRAIGHT: 1> <GameAction.RIGHT: 2>

<GameAction.STRAIGHT: 1> <GameAction.RIGHT: 2> <GameAction.RIGHT: 2>

<GameAction.RIGHT: 2> <GameAction.RIGHT: 2> <GameAction.STRAIGHT: 1>

<GameAction.RIGHT: 2> <GameAction.RIGHT: 2> <GameAction.LEFT: 0>

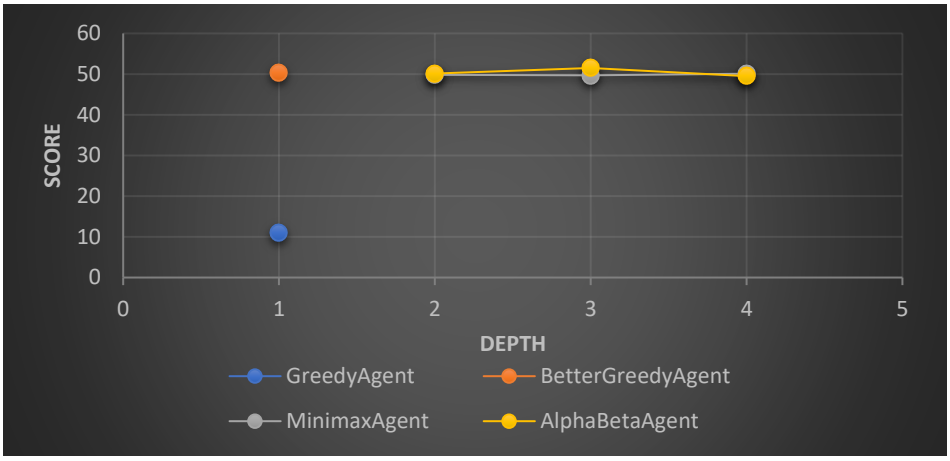
<GameAction.STRAIGHT: 1> <GameAction.RIGHT: 2>]

הסוכן שלנו הגיע לאורך 4, ולא ניצח. ביצענו 30 איטרציות על אוכלוסייה של 100 נחשים וראינו כי אין שיפור בפעולת הנחש שלנו.

לאחר בדיקה, הגענו למסקנה כי האלגוריתם הגנטי אינו מתאים לבעיה זו, מפני שלקיחת חלק מהפעולות של ההורה, מוציאה את הפעולות מהקשר. ייתכן כי אותן פעולות היו טובות לנחש ההורה לאחר שעשה פעולות שקדמו לפעולות שלקחנו, אך לאחר השינוי, פעולות אלה התחילו ממצב התחלתי שונה, ולכן לא הובילו לביצועים טובים כמו אצל ההורה. גם לאחר כמות גדולה של איטרציות לא היה שום שיפור בפעולת הנחש או עקביות כלשהי בנוגע לתוצאות.

חלק ז'

.2

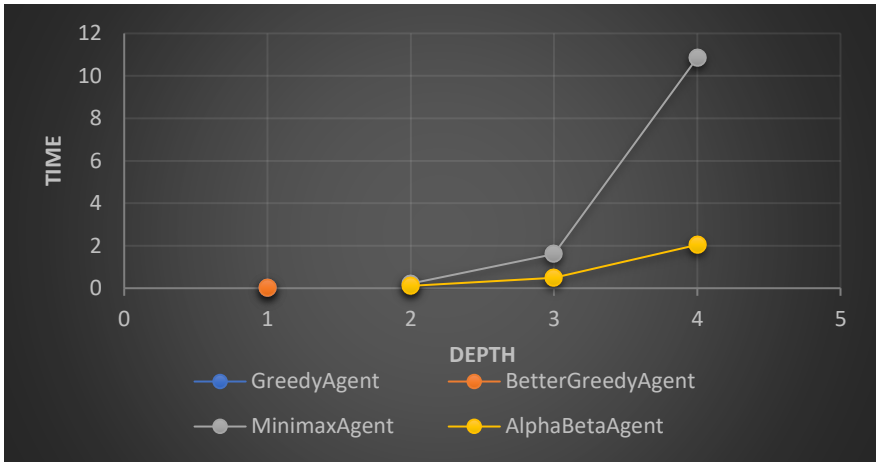


d=1	d=2	d=3	d=4	
11				**GreedyAgent**
50.3				**BetterGreedyAgent**
	49.8	49.7	50.1	**MinimaxAgent**
	50.1	51.5	49.5	**AlphaBetaAgent**

3. תחילה ניתן לראות כי, אכן כל האלגוריתם שמימשנו פועלים בצורה טובה יותר מאשר האלגוריתם הבסיסי GreedyAgent, הסיבה העיקרית לכך היא שימוש בהיוריסטיקה טובה יותר עבור המצבים. במקרה של GreedyAgent, תפקיד היוריסטיקה הוא בעיקר למנוע מהנחש למות (בשאר המקרים הוא נע די באקראיות). בכל שאר המקרים, תפקיד היוריסטיקה החדשה הוא לגרום לנחש לאכול כמה שיותר תפוחים ולהימנע ממוות ולכן ניתן בהחלט לראות כי הניקוד הסופי של הסוכנים המשופרים גבוה בהרבה מהניקוד של השחקן הבסיסי.

בנוסף, בניגוד לציפיות שלנו, ניתן לראות כי עומק החיפוש באלגוריתם minimax ו alpha beta , לא תמיד משפיע בצורה טובה על הניקוד הסופי של השחקן. תחילה הנחנו כי ככל שהשחקן שלנו יסתכל עמוק יותר כך יבטיח ניקוד גבוה יותר מפני שיוכל להימנע ממכשולים או כניסה למצבים שיגרמו לו למות בעתיד, אך לפי התוצאות ניתן לראות כי התוצאה אינה מתאימה לציפיות ברוב המקרים (נציין כי ביצענו רק 10 ניסוים לכל עומק, לכן ייתכן כי הסידור האקראי של הלוח בכל משחק הוא זה שגרם לתוצאה פחות טובה ולאו דווקא עומק החיפוש). אנחנו מניחים כי הירידה\חוסר השיפור בביצועים ככל שמעמיקים נובעת מההנחות שמבצעים באלגוריתם minimax , בהן מניחים כי הנחש האויב משחק על מנת להביא את התוצאה שלנו למינימום. ייתכן כי הנחש שלנו מוותר על ביצוע צעדים טובים (אכילת תפוחים או התקרבות לתפוחים) , מפני שמזהה כי ייתכן שבעתיד (יחסית לעומק החיפוש) הצעד יהיה לא לטובת הנחש שלנו (מזהה התנגשות בנחש אויב או מצב גרוע אחר). ולכן בסופו של דבר מגיע לניקוד נמוך יותר מאשר היה מגיע אם לא היה מסתכל עמוק.

4.



d=1	d=2	d=3	d=4	
0.006192				**GreedyAgent**
0.028454				**BetterGreedyAgent**
	0.2183	1.607826	10.85283	**MinimaxAgent**
	0.109592	0.484477	2.041039	**AlphaBetaAgent**

5. ניתן לראות בגרפים ובטבלה בסעיף 4, כי ככל שמעמיקים בחיפוש , כך גדל זמן החיפוש בצורה אקספוננציאלית.
בדומה לציפיות שלנו, ככל שעומק החיפוש גדול יותר, כך מספר המצבים אותם מפתחים משפיע על זמן החישוב. בנוסף ניתן לראות כי עבור עומק 4 בסוכן minimax , זמן החישוב הממוצע הינו 10 שניות, כלומר על מנת להשלים משחק יחיד באורך מלא (500 צעדים) , יהיה עלינו להמתין 5000 שניות או 83 דקות. (לקח לנו כמעט יומיים מלאים להריץ את כל הטסטים)

בנוסף, ניתן לראות כי אלגוריתם alpha beta מוריד בצורה משמעותית את זמן החישוב על ידי קיצוץ של מסלולים לא משפרים מעץ המצבים. השיפור בביצועים הינו משמעותי מאד ככל שמעמיקים את החיפוש , עבור עומק 4 ישנו שיפור של כמעט 9 שניות מאשר אלגוריתם minimax .

6. כעת נשווה בין כלל הסוכנים שמימשנו, כאשר נתייחס לנתונים המצויינים בגרפים, ולניסויים שביצענו בכל הסעיפים הקודמים:
תחילה נתייחס לסוכן GreedyAgent – כמו שראינו בסעיפים הראשונים, סוכן זה ממומש בצורה חמדנית. הוא מניח כי יכול לאכול בכל תור תפוח אחד ובוחר לבצע מהלך שייתכן כי ייתן לו תוצאה מקסימלית בהמשך(למרות שכמו שראינו, ייתכן גם כי הנחש ימות אם יבצע צעד זה).
בנוסף לכך היוריסטיקה המוגדרת לסוכן זה, הינה היוריסטיקה פשטנית מאד, אשר לא מבדילה בצורה טובה בין מצבים. לדוג' בהרבה מהמקרים תיתן היוריסטיקה ערך זהה לכל הפעולות האפשריות ולכן פעולת הנחש תהיה אקראית ובמקרים רבים יתקע הנחש בלולאה למשך זמן.
לעומתה אם נסתכל על הסוכן המשופר שמימשנו BetterGreedyAgent – היוריסטיקה החדשה שלו, מעניקה לו יתרון רציני מאד ביחס לסוכן הישן. כעת הסוכן בוחר במצב הבא בצורה כזו שתקדם אותו במשחק, ותעניק לו ניקוד גבוה יותר. ניתן לראות בניסויים שביצענו כי הסוכן הישן הצליח להשיג ניקוד של 11, לעומת השחקן המשופר, אשר לא מת אפילו פעם אחת בביצוע הניסויים, והשיג ניקוד של 50 למשחק בממוצע.
החיסרון הבולט ביותר שני הסוכנים הללו, הוא העובדה כי שניהם מסתכלים צעד אחד קדימה, כלומר אם קיים מכשול במרחק גדול יותר מצעד אחד, הסוכנים עיוורים לכך (למעט מכשולים כמו חסימה עצמית שהצלחנו למנוע על ידי היוריסטיקה שלנו).
היתרון של שני הסוכנים האלה, הוא זמן תגובה קצר מאד. נשים לב כי עבור כל אחד מהם זמן התגובה הממוצע קטן מ 30ms.

כעת נסתכל על סוכן הminimax – תחילה נציין כי סוכן זה השתמש באותה היוריסטיקה שהשתמש הסוכן BetterGreedy , כלומר הביצועים שלו יהיו יותר טובים מהסוכן הבסיסי בעיקר בגלל היוריסטיקה, לכן כאשר נבצע השוואה נתייחס בעיקר לסוכן המשופר שלנו ולא לסוכן הנתון בתרגיל.
כאשר נסתכל על תוצאות הניסויים שביצענו, נזהה כי ניקוד סוכן המינימקס כמעט וזהה לניקוד הסוכן החמדן המשופר מבלי שינוי משמעותי בעומק החיפוש.
לדעתנו, מקרה זה קורה, בגלל שימוש באותה היוריסטיקה, וגודל משחק קטן מידי (אויב יחיד , לא חכם, לוח קטן ומעט תפוחים). לדעתנו ייתכן כי אם נגדיל את המשחק, נוכל לזהות הבדלים יותר משמעותיים בין הסוכנים.
אחד המאפיינים ששמנו לב אליו, הוא ירידה קלה בביצועים של סוכן המינימקס ככל שמגדילים את עומק החיפוש, וכמו שציינו בסעיפים קודמים, לדעתנו זה נובע מכך שהסוכן נהיה יותר "פרנואידי" ביחס למשחק, ולפעמים מזהה כי פעולה מסויימת יכולה לגרום לו להפסיד ניקוד ולכן מסרב לבצע אותה למרות שברוב המקרים ייתכן כי יגדיל את הניקוד שלו. בגלל בחירות אלה, ייתכן כי הסוכן "מבזבז" זמן בלהימנע מצרות, ובכך מאפשר לסוכן האויב לאכול פירות ובכך הסוכן שלנו מאבד ניקוד אפשרי.
יתרון עיקרי בשימוש בסוכן זה, הינה הבדיקה לעומק, הסוכן יכול להימנע ממקרים רעים, ובנוסף "הפרנויה" של הסוכן, תורמת לו בכך שמונעת ממנו למות ברוב המקרים (אם מזהה כי ימות במסלול מסויים, לא יבצע את הפעולה שתגרום למותו אם יש פעולה אחרת בה הוא חי). כלומר אם המשחק היה לשרוד כמה שיותר, לדעתנו הסוכן מינימקס היה מתעלה על הרבה מיריביו.
חיסרון עיקרי בסוכן זה ביחס לסוכנים הקודמים, הוא זמן התגובה שלו. כמו שראינו בניסויים זמן התגובה של הסוכן, גדלה בצורה אקספוננציאלית, כאשר עבור עומק 4 קיבלנו זמן תגובה ממוצע של 10 שניות, ואם נרצה להתסכל יותר עמוק נוכל להגיע לכמה דק'\שעות לביצוע פעולה יחידה. כלומר, סוכן זה אינו ישים בעולם האמיתי מפני שניתן להפעילו ברוב המקרים בעומקים רדודים מאד, שבדר"כ תורמים פחות לסוכן שלנו מבחינת ביצועים.
נתייחס כעת לסוכן Alpha Beta – כמו שהסברנו בסעיפים הקודמים, בחירת הפעולות של סוכן זה תהיה זהה לחלוטין לסוכן מינימקס, וההבדל היחיד בין הסוכנים הינו קיצור זמן התגובה לכל פעולה. כלומר, מבחינת ניקוד השחקן, נצפה לראות מגמה זהה למגמה אצל סוכן המינימקס.
ואכן גם אצל סוכן זה אנחנו מזהים בניסויים ירידה קלה בביצועים כאשר מגדילים את עומק החיפוש, אשר נוסעת מאותן סיבות שציינו עבור סוכן המינימקס.
יתרון עיקרי של סוכן זה יחסית לסוכן מינימקס, הינו קיצור משמעותי בזמן התגובה.
ניתן לראות בניסויים כי בממוצע הצלחנו לחסוך פי 5 זמן בשימוש בסוכן אלפא בטא בעומק 4 מאשר בסוכן המינימקס. ואנחנו מאמינים שככל שנעמיק את החיפוש כך השיפור בזמן במקרה הממוצע יהיה גדול יותר ויותר.
בגלל יתרון משמעותי זה, הצלחנו לגרום לסוכן להיות ישים בעולם האמיתי עבור עומקים יותר עמוקים מאשר הסוכן מינימקס, אך גם במקרה זה השיפור לא מספיק טוב כדי לתת לנו ביצועים טובים מספיק. כלומר כעת נוכל לפעול עד עומק 3 במקום 2 אם נרצה תגובה מהירה.
לסיכום, ראינו כי מגבלת העומק לא תמיד מביאה לביצועים יותר טובים בסוכנים שבדקנו, וכי אם הייתה מגבלת זמן, לא היינו יכולים להריץ את הסוכנים המורכבים יותר אלא רק את הסוכנים הפשוטים.

חלק ח'

שחקן הטורניר שלנו:

עבור שחקן הטורניר בחרנו להשתמש בגירסא דומה לשחקן BetterGreedyAgent עם כמה שינויים.

תחילה, ביצענו שינוי בצורת בחירת הפעולה. השחקן שלנו, מפתח עבור כל פעולה אפשרית שלו, את כל המצבים העוקבים האפשריים עבור פעולות האויב (כמו האלגוריתם החמדן הרגיל), אך לעומת האלגוריתם החמדן הרגיל, אנחנו בוחרים לבצע את הפעולה שהביא לממוצע ניקוד גבוה יותר.
נזכיר כי האלגוריתם החמדן, עובר על כל אפשרויות הפעולה של האויב, ובוחר מסך כל המצבים

שבדק, את הפעולה שהביאה אותו לניקוד הגבוה ביותר (כלומר אם פנייה ימינה פיתחה מצב עם ניקוד גבוה ביותר, השחקן יבחר בה, גם אם באחד המצבים שפותחו עם פעולה זו, השחקן שלנו מת או הוציא ציון נמוך יותר – כלומר האלגוריתם החמדן מניח שנוכל להשיג את הניקוד הגבוה ביותר שנמצא), לעומת זאת האלגוריתם שלנו עובר על כל המצבים האפשריים לאחר פעולה מסויימת ומחשב ממוצע על הניקוד שהתקבל (כלומר האלגוריתם שלנו מבצע הנחה יותר מקלה, אשר מקיימת כי במקרה הסביר נצליח לקבל את הניקוד הממוצע אם נפעל לפי הפעולה המתאימה).

שינוי שני שביצענו, הינו שינוי בהיוריסטיקה בה נשתמש. היוריסטיקה בה נשתמש זהה להיוריסטיקה שבנינו בתחילת התרגיל, אך כעת נוסיף לה שינוי בתחילת הריצה על מנת לחסוך בזמן חישוב.
השינוי שהוספנו הוא בפונקציה המחשבת את השטח שניתן להגיע אליו בכל רגע. פונקציה זו נועדה בשביל למנוע מהשחקן לחסום את עצמו ולגרום למוות, ההנחה שלנו היא, שכאשר השחקן שלנו קצר, הסבירות שיחסום את עצמו קטנה מאד, כלומר אין צורך לבצע חישובים כבדים (ביצוע DFS לעומק מסויים), על מנת לוודא שהשחקן לא יחסום את עצמו, מפני שהשחקן קצר מאד.
לכן הוספנו כי החלק הזה בהיוריסטיקה יתחיל לפעול רק לאחר שהנחש יגדיל את אורכו להיות גדול מ-6.

מלבד שני השינויים האלה, השארנו את האלגוריתם והיוריסטיקה כמו שהם.

בחרנו בשיטת פעולה זו, מפני שאנחנו מאמינים כי היוריסטיקה שלנו הינה היוריסטיקה טובה מאד, אשר מונעת מהשחקן שלנו למות, ובנוסף גורמת לו להשיג ניקוד גבוה מאד (בכל המשחקים שהרצנו עם היוריסטיקה הזאת, השחקן שלנו לא מת אפילו פעם אחת).
בנוסף השינוי הראשון שביצענו, נועד למנוע מהשחקן שלנו לבצע צעדים מסוכנים מידי ובכך להימנע ממוות על ידי השחקן האויב.