מבוא לבינה מלאכותית - תרגיל בית 2

4 בינואר 2021

שאלה 1

האסטרגיה של השחקן היא לבחור את העמדה בין 4 האפשרויות (לכל היותר) שבו מספר האפשרויות ללכת הוא המינימלי.

היתרון באסטרטגיה זו היא לנצל כמה שיותר את המשבצות שניתן לעבור אליהם בסביבת השחקן ולצמצם משבצות לא מנוצלות. בנוסף, שחקן זה נצמד לשפות הלוח (בגלל שבשפת הלוח יש פחות צעדים לבצע - מה שגורם לניצול הלוח באופן אופטימלי.

החסרון הוא חוסר יכולת הסתכלות לעתיד כיצד השחקן היריב יפעל, כלומר הוא לא תלוי בפעולותיו של היריב.

שאלה 2

0	0	0	0
0	2	x	0
x	x	x	x
0	1	x	0
0	0	0	0

נשים לב שבפעם הראשונה שחקן 1 (נניח כי הוא $SimplePlayer$) יצמד לשפה (צד שמאל) ואז מסלולו ייקבע מראש, לאחר מכן אם שחקן 2 יצמד לשפה השמאלית מסלולו ייקבע מראש ונקבל תיקו. במידה ושחקן 2 יעבור למשבצת העליונה, למעשה שחקן 2 יפסיד כי לשחקן 1 יש יותר צעדים לבצע.

לכן, נסיק שאכן מדובר בלוח שבו שחקן $SimplePlayer$ יפעל בצורה אופטימלית.

שאלה 3

יתרון:

- צבירת נקודות בטווח הקצר.

חסרונות:

- פעולות השחקן לא תלויות בפעולות השחקן יריב, כלומר הוא לא מסתכל על מיקום השחקן היריב.
- חוסר ניצול מקום כתוצאה מהעדפה ללכת לפרס הקרוב.

שאלה 4

נגדיר את היורסטיקה הבאה:

$$H\left(s\right)=sumFruit\left(s\right)+100\cdot\frac{1}{\min_{fruit\in fruits}\left\{md\left(s,fruit\right)\right\}}+enemyDist\left(s\right)+5\cdot availableMoves\left(s\right)+\frac{1}{distToFrame\left(s\right)}$$

הסבר:

$\min_{fruits}\left\{md\left(s,fruit\right)\right\}$ - המטרה במשחק היא לצבור כמה שיותר נקודות. לכן, נעדיף לצבור פירות כמה שיותר מהר.

$enemyDist\left(s\right)$ - מחשב את המרחק בין השחקן ליריב במידה וקיים (כלומר במידה ו-$isPath\left(s\right)=true$). האסטרטגיה היא להתרחק מהיריב כמה שיותר כיוון שהתקרבות אליו יכולה להביא למצב בו הוא יחסום אותנו ונפסיד.

$availableMoves\left(s\right)$ - מספר המשבצות הלבנות (כולל פרי) שניתן ללכת אליהם ממשבצת של $s.pos$. האסטרגיה היא לעבור למצבים עם יותר אפשרויות ובכך להימנע ממבוי סתום.

$distToFrame\left(s\right)$ - המרחק המינימלי לשפת הלוח. האסטרטגיה היא להיצמד לשפת הלוח ולכן חיברנו את המחובר $\frac{1}{distToFrame(s)}$ - בכך נצמצם משבצות לא מנוצלות.

$sumFruit$ - פונקציה אשר סוכמת את סך הנקודות שהרווחנו מהפירות כאשר מחלקים כל פרי שנסכם בערכו של מספר הצעדים שלקח מהצעד הנוכחי בלוח.

שאלה 5

ראשית, אסטרטגית $Minimax$ מתאימה עבור משחק של יותר משני שחקנים.

סעיף א'

חסרונות:

- נשים לב כי כאשר יש 3 שחקנים ומעלה, השחקן מניח כי השחקן היריב מבצע את הצעד הגרוע ביותר עבורנו (שזהו למעשה הצעד הטוב ביותר עבור היריב). אך, הצעד הטוב ביותר עבור היריב תלוי בשאר השחקנים, ולכן אסטרטגיה זו לא בהכרח תניב את בחירת הצעד הגרוע עבור היריב.

- ברגע שיש יותר משני שחקנים, עלינו לייצג את הבחירות של כל שחקן. אנו יודעים מההרצאה שהחיפוש הוא סוג של חיפוש לעומק ולכן דרישות הזיכרון הוא לינארי כעומק עץ המשחק ומכיוון שהוספנו עוד שחקנים עומק העץ גדל. בפרט, סיבוכיות המקום גדלה.
- מכיוון שחישוב ערך כל צומת נקבע לפי חישוב ה-$Minimax$ של הילדים, אז החישוב גדל משמעותית כשמוסיפים שחקנים נוספים (כי עומק העץ גדל).

סעיף ב'

ראשית, נשים לב כי לא נוכל להשתמש באלגוריתם $minimax$ מכיוון שלא בהכרח נכון שהצעד הטוב ביותר עבור היריב הוא הצעד הגרוע ביותר עבורנו. כפי שהוסבר בסעיף א', הדבר תלוי בפרמטרים נוספים. לכן, נציע אלגוריתם כאשר בכל תור של השחקן ה-i, נחפש מהו הצעד הטוב ביותר עבור שחקן זה. נשמור וקטור בגודל מספר השחקנים, שיכיל בכניסה ה-i את ערך עבור השחקן ה-i בתור זה.

נניח וקיימים k שחקנים ומקדם הסיעוף הוא B, ושחקן 1 מתחיל. נסמן את גובה עץ הפיתוח ב-n. בגובה 1 בעץ הפיתוח עבור כל צומת נסתכל על הכניסה ה-$n\%k$ ונסמן את הבנים של הצומת הנוכחית להיות $c_1, \ldots c_B \in \mathbb{R}^k$ ונקח את הוקטור c_{i_0} כך
ש-$c_{i_0}\left[n\%k\right] = \max_{1\le i\le B}\left\{c_i\left[n\%k\right]\right\}$ - כלומר נסתכל הווקטורים המוחזרים מכל הבנים בקואורדינטה $n\%k$ שלהם ונעלה לצומת האב את הוקטור שבקואורדינטה זו הוא מקסימלי.

נבצע זאת בכל צומת עד שנגיע לשורש ונחזיר את הקואורדינטה הראשונה.

שאלה 6

סעיף א'

זמן הריצה של $Alpha-beta$ יהיה קצר מאשר סוכן ה-$Minimax$ מכיוון שב-$Alpha-beta$ יתבצע בחלק מהפעמים גיזום של מסלולים באלגוריתם החיפוש ולכן זמן הריצה יתקצר.

סעיף ב'

ערך ה-$Minimax$ בשני האלגוריתמים אינם משתנים מכיוון שאנו עוברים על המסלולים הטובים ביותר. ההבדל העיקרי הוא ביעילות חיפוש המסלול המתאים. מכיוון שעומק החיפוש זהה בשני האלגוריתמים נקבל את אותו ערך $Minimax$ ולכן במקרה זה סוכן $Alpha-beta$ יתנהג באופן זהה לסוכן $Minimax$.

שאלה 7

סעיף א'

זמן הריצה של $Alphabeta$ עם סידור ילדים יהיה מהיר יותר כי כאשר אם אנו בצומת max וערך ה-$minimax$ של אחד הבנים גדול מ-$beta$, נדע זאת לאחר פיתוח הבן הראשון ונגזום, ואם אנו בצומת min וערך ה-$minimax$ של אחד הבנים קטן מ-$alpha$, נדע זאת לאחר פיתוח הבן הראשון ונגזום.

סעיף ב'

במידה ומגבלת העומק זהה, סוכן $Alpha - beta$ יתנהג בהכרח אותו דבר כמו סוכן $Alpha - beta$ עם סידור ילדים.

במידה וקיימת הגבלת סוכן $Alpha - beta$ עם סידור ילדים יתנהג יותר טוב (או באופן זהה) מאשר סוכן $Alpha - beta$ בלבד.

שאלה 8

אלגוריתם $Anytime\ contract$ של $Minimax$ מחזיר את הצעד הטוב ביותר שיוכל למצוא תוך זמן כלשהו.

נפעיל את אלגוריתם החיפוש עם עומקים הולכים וגדלים כאשר בכל חישוב נשמור את הצעד האחרון שעשינו שעמד במגבלת הזמן וכאשר הזמן נגמר מפסיקים את החיפוש ומחזירים את הצעד האחרון שנשמר.

שאלה 9

הבעיה המרכזית בהעמקה הדרגתית היא שאנו משתמשים במשאבים רבים לחישוב הצעד האחרון שלא בהכרח יסתיים ויחזיר תשובה.

פתרון אפשרי הוא שבכל איטרציה נשמור את ערך המינימקס של כל אחד מהבנים ברמה העליונה. בנוסף, באיטרציה האחרונה (נניח בעומק 10) החצי הראשון של הבנים ישקף חיפוש עמוק יותר לעומק 10 והחצי השני לעומק 9. כך לפחות עבור חצי מהבנים ניצלנו את המשאבים כדי לראות לעומק גדול יותר.

שאלה 10

ראשית, נריץ את אלגוריתם BFS על 2 הקודקודים שבהם השחקנים נמצאים על מנת למצוא את מספר הצעדים הזמינים של כל שחקן. נשמור משתנה בוליאני $isTogether$ שבהתחלה יהיה מאותחל ל-$False$. שערכו יהיה $True$ אמ"ם קיים מסלול בין שחקן 1 לשחקן 2.

בנוסף, נגדיר משתנים $counter1$ ו-$counter2$ להיות מאותחלים אשר יהוו את מספר המשבצות הזמינות במפה

במהלך האלגוריתם BFS במהלך כל פיתוח של צומת (שערכה 0 או גדול מ-2, כלומר פרי), נבדוק האם הגענו לשכן אשר מהווה את הקודקוד של השחקן היריב. במידה והגענו לצומת כזאת, נגדיר את $isTogether$ להיות $True$.

בנוסף, במהלך האלגוריתם BFS על כל צומת שפיתחנו נוסיף 1 ל-$counter$ המתאים לשחקן. בסוף האלגוריתם, נדע כמה משבצות זמינות במפה יש לכל אחד מהשחקנים.

- במידה ו-$isTogether = False$ אזי נגדיר $counter1 = num\ turn1$. ונגדיר $time\ frame1 = \frac{global\ time}{num\ turn1}$. בנוסף, נגדיר $num\ turn2 = counter2$. ונגדיר $time\ frame2 = \frac{global\ time}{num\ turn2}$.
- במידה ו-$isTogether = True$ נבצע את הפעולות הבאות:
  - נפעיל את האלגוריתם BFS פעמיים נוספות כאשר בפעם הראשונה נפעיל את האלגוריתם על הקודקוד של השחקן הראשון כאשר נחליף את ערך המשבצת של השחקן השני לערך -1 (לצורך ריצה זו בלבד) ונגלה את מספר הצעדים ששחקן 1 יכול לבצע ונסמנו $counter1$. בפעם השנייה נפעיל את האלגוריתם על הקודקוד של השחקן השני כאשר נחליף את ערך המשבצת של השחקן הראשון לערך -1 (לצורך ריצה זו בלבד) ונגלה את מספר הצעדים ששחקן 2 יכול לבצע ונסמנו $counter2$.
  - לבסוף, נגדיר $counter1 = num\ turn1$. ונגדיר $time\ frame1 = \frac{global\ time}{num\ turn1}$. בנוסף, נגדיר $counter2 = num\ turn2$. ונגדיר $time\ frame2 = \frac{global\ time}{num\ turn2}$

שאלה 11

אפקט האופק זוהי תופעה בה אלגוריתם מוגבל משאבים בוחר צעדים ''סתמיים'' כדי לדחות ''צרות'' מעבר לאופק החיפוש.

הפתרון המוצע בהרצאה לבעיה זו הוא העמקה סלקטיבית. במקרה זה, פורשים כמו קודם עץ מלא עד עומק מסוים. אבל, כאשר המצב בעלה אינו ''שקט'', כלומר לא מיטבי לטובתנו נרצה להעמיק עוד טיפה על-מנת להגיע למצב יותר טוב עבורינו.

לדוגמא נקח לוח שהעומק ההתחלתי בהרצת האלגוריתם הוא 2.

0	X	1
0	X	0
0	X	0
2	X	0
X	X	0

כעת, כאשר הגענו לעומק $d = 2$ (למגבלת העומק) נגיע ללוח הבא:

0	X	X
2	X	1
X	X	0
X	X	0
X	X	0

ניתן לראות כי בעוד מספר צעדים קטן שחקן 1 הולך לנצח ולכן נרצה להמשיך להעמיק עוד טיפה על-מנת לקבל את הניצחון המובטח במקום שנקבל ערך יוריסטי שלא בהכרח מציג את המצב לטובתנו.

דוגמא נוספת:

לדוגמא נקח לוח שהעומק הנוכחי בהרצאת האלגוריתם הוא 2.

0	0	X	1
0	0	X	0
0	0	X	0
2	0	X	160
X	0	X	150

כעת, כאשר הגענו לעומק $d = 2$ נגיע ללוח הזה

2	0	X	X
X	0	X	X
X	0	X	1
X	0	X	160
X	0	X	150

נשים לב שאם נעמיק עוד מספר צעדים קטן שחקן 1 יטה את הכף לטובתו (כלומר ייתכן שהנקודות שיצבור בצעדים הבאים יגרמו לנו לנצח), ולכן במקרה זה נרצה להמשיך עוד טיפה במקום שנקבל ערך יוריסטי שלא בהכרח מציג את המצב לטובתנו.

שאלה 12

תהי h פונקציה היוריסטיקה - נגדיר אותה להיות קבילה. כעת, נסמן את הערך של העץ הנתון ב-$value$.

כעת, נפריד למקרים :

- נשים לב כי בצמתי max נפעיל את הפונקציה היוריסטית על כל אחד מהבנים ונבדוק האם $h\left(child\right) > value$. במידה והתנאי מתקיים נגזום מכיוון שלא מתקיים התנאי ש-$h \leq h^{*}$ ולכן איננו חלק מהמסלול עבור הצעד המומלץ ונמשיך לבן הבא. אחרת, נפתח צומת זאת.
- נשים לב כי בצמתי min נפעיל את הפונקציה היוריסטית על כל אחד מהבנים ונבדוק האם $h\left(child\right) < value$. במידה והתנאי מתקיים נגזום מכיוון שלא מתקיים התנאי ש-$h \leq h^{*}$ ולכן איננו חלק מהמסלול עבור הצעד המומלץ ונמשיך לבן הבא. אחרת, נפתח צומת זאת.

<u>במקרה הטוב ביותר -</u> בתחילת האלגוריתם נגיע לצומת ה-min/max ונגזום את כל הילדים חוץ מילד אחד - אשר יהווה חלק מהמסלול לפתרון הטוב ביותר, בזמן הטוב ביותר.

<u>במקרה הרע ביותר -</u> בכל צומת min/max לא נבצע אף גיזום (מכיוון ש-$h\left(child\right) \leq value$ בצומת מקסימום או $h\left(child\right) \geq value$ בצומת מינימום) עבור כל הבנים של צמתי המקסימום והמינימום - דבר אשר יעכב את מציאת הפתרון הטוב ביותר ויגרום לפיתוח כל הצמתים פעם נוספת.

<u>במקרה הכללי -</u> עבור כל צומת min/max נבדוק האם התנאי מתקיים - במידה ומתקיים נגזום, אחרת נמשיך לבן הבא. דהיינו, נקבל כי ייתכן שחלק מתתי העץ של צמתי ה-min/max ייגזמו וחלקם לא - דבר אשר יגרום לצמצום בזמן הריצה.

שאלה 13

תחילה, נגדיר $e_{max}=-\infty$, $e_{min}=\infty$. נסמן את מקדם הסיעוף עבור צומת הסתברותי להיות k.

כעת, נפצל למקרים:

במידה ואנו בצומת הסתברותית כאשר הצומת הקודם הוא צומת max נחשב את סדרת ערכים $a_1,a_2,\ldots,a_k$ (עבור D העומק הנוכחי) באופן הבא:

$$a_i=\sum_{m=1}^{i}p_m\cdot RB-Expectimax\left(Child_m,D-1\right)+\sum_{m=i+1}^{k}p_m\cdot\underbrace{H_{max}}_{=5}$$

כעת, עבור כל $1\le i\le k$, במידה ו-$a_i<e_{max}$ אז מבצעים גיזום עבור הצומת הנוכחית. במידה והתנאי לא מתקיים ממשיכים לבן הבא.

בנוסף, מעדכנים את e_{max} להיות $e_{max}=\max\left\{e_{max},a_i\right\}$.

במידה ואנו בצומת הסתברותית כאשר הצומת הקודם הוא צומת min נחשב את סדרת ערכים $a_1,a_2,\ldots,a_k$ (עבור D העומק הנוכחי) באופן הבא:

$$a_i=\sum_{m=1}^{i}p_m\cdot RB-Expectimax\left(Child_m,D-1\right)+\sum_{m=i+1}^{k}p_m\cdot\underbrace{H_{min}}_{=-5}$$

כעת, עבור כל $1\le i\le k$, במידה ו-$a_i>e_{min}$ אז מבצעים גיזום עבור הצומת הנוכחית. במידה והתנאי לא מתקיים ממשיכים לבן הבא.

בנוסף, מעדכנים את e_{min} להיות $e_{min}=\min\left\{e_{min},a_i\right\}$.

הרעיון מאחורי הגיזום: בכל צומת הסתברותית (כאשר הצומת לפניה היא צומת מקסימום/מינימום) נעדכן את החסם העליון/תחתון (כלומר את הערכים e_{max} ו-e_{min} בהתאמה), על מנת לדעת האם בתורות הבאים יהיה ניתן לבצע גיזום או לא.

שאלה 14

סעיף א'

מכיוון שבמהלך הראשון, השחקן מחליט לחפש לעומק $D+1$, זמן החיפוש הוא מעשה פונקציה של מספר הקריאות לפונקציה היוריסטית. נשים לב כי בחיפוש מלא (בעומק $D+1$) אנו מגיעים לרמה התחתונה ביותר (עומק 0) ולכן בהכרח קוראים לפונקציה היוריסטית כמספר העלים של עץ החיפוש - שהוא למעשה B^{D+1} ולכן זמן החיפוש של המהלך הראשון הוא $B^{D+1} \cdot h_{time}$ (עבור זמן של פונקצית היריסטיקה - h_{time}).

נתון כי ב-$\frac{M}{B}$ דקות ניתן לחפש בעומק D. במילים אחרות בעץ בעומק D יש B^D עלים ולכן זמן החיפוש $B^D \cdot h_{time}$. לכן, נסיק מהנתון כי $\frac{M}{B} = B^D \cdot h_{time}$. אבל, בנוסף אנו יודעים כי ולכן $M = B \cdot \frac{M}{B} = B \cdot B^D \cdot h_{time} = B^{D+1} \cdot h_{time}$.

כלומר זמן החיפוש של המהלך הראשון לקח M דקות. לכן, לא יישאר זמן למהלכים אחרים והשחקן יפעל ב-$B-1$ המהלכים ללא כל אסטרטגיה.

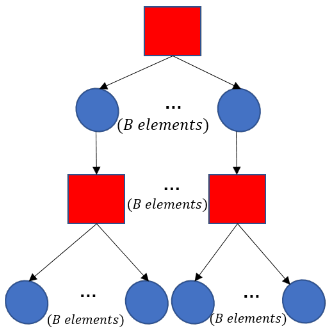
סעיף 1

ראשית, נשים לב שהשחקןשומר את האסטרגיה בצעד הראשון שמתבצע לעומק $D+1$.

כמו כן, אין צורך לשמור את כל הקשתות אשר יוצאות מצומת מקסימום אלא רק הקשת בעלת הערך הגדול ביותר (בה מתקבל ערך מקסימלי), מכיוון שהשחקן תמיד יעדיף לקבל $score$ גבוה ככל הניתן. לעומת זאת, מצומת מינימום נצטרך לשמור את כל הקשתות כי אנו לא יודעים כיצד היריב פועל ובמה הוא יבחר.

לכן, ניתן להסיק כי מספר הקשתות של השחקן שעליו לשמור הוא $\sum_{i=1}^{\lfloor \frac{D}{2} \rfloor} B^i$ ומספר הקשתות שיש לשמור של סוכן המינימקס הוא $\sum_{i=1}^{\lceil \frac{D}{2} \rceil} B^i$.

להלן דוגמא של העץ אותו אנו שומרים:



סעיף 2

נשים לב כי המצב השתפר מעצם העובדה שאנו שומרים את העץ מהסעיף הקודם שהושקע זמן לפיתוחו ובכך לשחקן יהיה מידע להשתמש בו, לעומת סעיף א' בו השחקן שיחק ללא כל אסטרטגיה ב-$B-1$ המהלכים הנותרים.

כמו כן, בתור ה-i ביכולתו להסתכל עד עומק $D-1-i$ ולקבל מידע נוסף אשר ישפיע על אופן פעולותו.

סעיף 3

נשים לב כי השחקן המשופר יהיה יותר טוב משחקן $Minimax$ רגיל כאשר קיים מידע בעומק $D+1$ שלא ניתן היה לגלות אותו בעומק D.

לדוגמא, במשחק (נניח כי שחקן 1 מתחיל) בו המטרה היא לקחת כמה שיותר מטבעות נשים לב כי אנו מעמיקים לעומק $D=2$ שחקן 1 לא יגלה את ה-$coin$ השמאלי, לעומת זאת, אם נעמיק לעומק 3 אכן נגלה את המטבע השמאלי ביותר, דבר אשר יכול להוביל לניצחונו.

coin		1			coin		2

לעומת זאת, השחקן המשופר יהיה פחות טוב משחקן $Minimax$ רגיל במקרה הבא (נניח כי שחקן 1 מתחיל):

Coin		1			Coin			2
x	x	x	x	x	Coin	x	x	x
x	x	x	x	x	Coin	x	x	x
x	x	x	x	x	Coin	x	x	x

מכיוון שבמהלך הראשון מעמיקים לעומק 3 השחקן יגלה את המטבע השמאלי ביותר ויתקדם לכיוונו. בזמן זה, שחקן 2 יתקדם למכרה המטבעות באמצע הטבלה וינצח במשחק. אם שחקן 1 היה מעמיק לעומק D - דהיינו 2, אז במידה ולא היה מגלה את המטבע השמאלי ביותר והיה הולך ימינה, ומגיע לפני שחקן 2 למכרה המטבעות ומנצח.

שאלה 15

נגדיר את היוריסטיקה הבאה:

$$H(s) = sumFruit(s) + 100 \cdot \frac{1}{\min_{fruit \in fruits}\{md(s, fruit)\}} + enemyDist(s) + 5 \cdot availableMoves(s) + \frac{1}{distToFrame(s)}$$

$\min_{fruits}\{md(s, fruit)\}$ - המטרה במשחק היא לצבור כמה שיותר נקודות. לכן, נעדיף לצבור פירות כמה שיותר מהר.

$enemyDist(s)$ - מחשב את המרחק בין השחקן ליריב במידה וקיים (כלומר במידה ו-$isPath(s) = true$. האסטרטגיה היא להתרחק מהיריב כמה שיותר כיוון שהתקרבות אליו יכולה להביא למצב בו הוא יחסום אותנו ונפסיד.

$availableMoves(s)$ - מספר המשבצות הלבנות (כולל פרי) שניתן ללכת אליהם ממשבצת של $s.pos$. האסטרטגיה היא לעבור למצבים עם יותר אפשרויות ובכך להימנע ממבוי סתום.

$distToFrame(s)$ - המרחק המינימלי לשפת הלוח. האסטרטגיה היא להיצמד לשפת הלוח כדי לנצל את המרחב כמה שיותר.

$sumFruit$ - פונקציה אשר סוכמת את סך הנקודות שהרווחנו מהפירות כאשר מחלקים כל פרי שנסכם בערכו של מספר הצעדים שלקח מהצעד הנוכחי בלוח.

שאלה 16

נגדיר את היוריסטיקה הבאה:

$$H(s) = sumFruit(s) + sumOfWeightedFruits(s) + enemyDist(s) + 5 \cdot availableMoves(s) + \frac{1}{distToFrame(s)}$$

$sumOfWeightedFruits(s)$ - עבור כל משבצת ב-$k \in succ(s)$ נחשב את: $\sum_{i=0}^{|Fruits|} \frac{1}{dist(k, fruit(i))} \cdot weight(fruit(i))$.

כאשר $dist(k, fruit(i))$ - הוא מרחק מנהטן מ-k לפרי ה-i ו-$weight(fruit(i))$ הוא ערכו של הפרי ה-i. האסטרטגיה היא לצבור פירות תוך התחשבות במשקלן ובמרחקן.

$distToFrame(s)$ - המרחק המינימלי לשפת הלוח. האסטרטגיה היא להיצמד לשפת הלוח כדי לנצל את המרחב כמה שיותר.

השחקן יסתכל על כל הפירות בלוח המשחק ויסכום עבור כל פרי את $\frac{1}{dist(k, fruit(i))} \cdot weight(fruit(i))$.

נשים לב כי אנו לא מתחשבים רק בפרי הקרוב ביותר, אלא בסך הפירות שעל הלוח תוך התחשבות במרחק.

ייתכן כי יש כמה פירות שקרובים לשחקן אך ערכם הכולל נמוך, בעוד ייתכן וקיים פרי אחד שערכו גבוה מאוד אשר רחוק יותר - במקרה זה ייתכן שנעדיף ללכת לפרי הרחוק. בנוסף, נתחשב בפרמטרים הנוספים אשר הזכרנו בסעיפים קודמים.

כאשר אין פירות במפה, אנו משתמשים בפרמטרים האחרים שהשתמשנו שהוגדרו לעיל.

$sumFruit$ - פונקציה אשר סוכמת את סך הנקודות שהרווחנו מהפירות כאשר מחלקים כל פרי שנסכם בערכו של מספר הצעדים שלקח מהצעד הנוכחי בלוח.

שאלה 17

נתאר עבור שני מקרי הגבלת הזמן:

- **זמן מוגבל לתור** - בלולאה הראשית ב-$make\_move$ הפעלנו מונה זמן לפני ולאחר מכן פונקציית $serach$. חישבנו את הפרש הזמנים והוספנו אותו לזמן הכולל שסכמנו עד כה ולפני ביצוע איטרציה נוספת שאלנו האם $totalTime{+}3{*}currTime < timeLimit$, כאשר $currTime$ היא זמן האיטרציה האחרונה. הסיבה היא שהכפלנו ב-3 היא כי אם לקח לנו עבור איטרציה כלשהי t זמן אז באיטרציה הבאה מכיוון שהעומק גדול ב-1 ומכיוון שמקדם הסיעוף הוא 4 (כי בכל משבצת יש 4 אפשרויות ללכת אליהן) נסיק כי לכל היותר באיטרציה הבאה הזמן שייקח הוא פי 3 - דהיינו $3 \cdot t$. לבסוף, נצא כאשר התנאי ונחזיר את הפתרון עבורו העומק מקסימלי תחת הגבלת הזמן.

- **זמן מוגבל גלובלי** - בתחילת הריצה (בהפעלת הפונקציה set_game_params) הפעלנו BFS על המשבצת שערכה 1 בלוח המשחק (שחקן 1) על-מנת לגלות את מספר המשבצות שאליהן ניתן ללכת (כולל פירות). לאחר מכן חילקנו את התוצאה ל-2 על מנת לגלות את מספר המשבצות המקסימלי שכל שחקן יכול לעבור במהלך התוכנית (שמרנו מספר זה ב-$self.time\_to\_turn$). לאחר מכן, חילקנו את הזמן הכולל ב-$self.time\_to\_turn$ - והגדרנו את התוצאה להיות הזמן המקסימלי עבור תור בודד.

שאלה 18

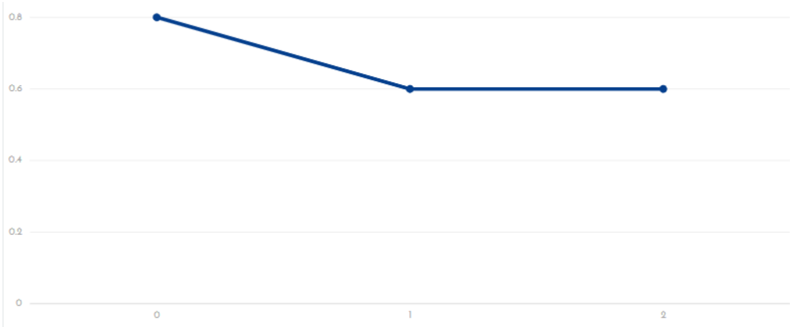
התוצאות שקיבלנו הן שסוכן ה-$AlphaBeta$ ניצח ביותר משחקים את סוכן ה-$Minimax$.

בנוסף, נשים לב כי התוצאות אכן מתאימות לציפיותינו מכיוון שתחת מגבלת זמן לתור הסוכן $AlphaBeta$ מספיק לפתח יותר אפשרויות הודות לתכונת הגיזום שמתבצע באלגוריתם ולכן הוא מגיע לצעד הבא שיותר טוב עבורו מאשר הצעד הבא המחושב ע"י אלגוריתם ה-$minimax$.

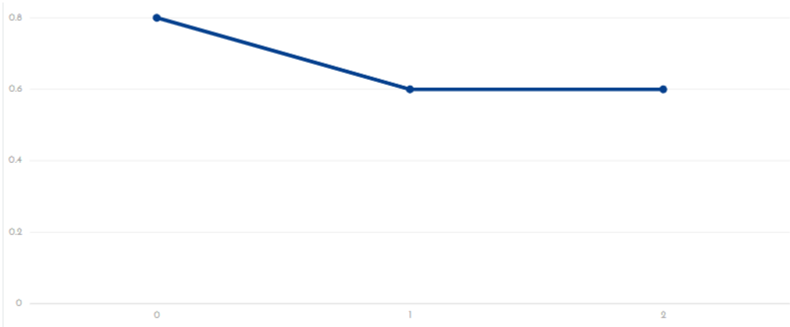
שאלה 19

להלן תוצאות הניסוי:

עבור עומק 3 של $HeavyPlayer$:



עבור עומק 2 של $HeavyPlayer$:



נשים לב ששחקן ה-$HeavyPlayer$ ניצח ביותר משחקים משחקן ה-$LightPlayer$ על אף ששחקן ה-$LightPlayer$ מחפש לעומקים גדולים יותר אך כאשר $LightPlayer$ חיפש לעומקים גדולים יותר הוא ניצח ביותר משחקים מאשר בעומקים נמוכים יותר, זאת מכיוון שחיפוש לעומקים יותר גבוהים מקרב אותנו לתוצאה האמיתית שהיא ערך החזרה מפונקציית ה-$utility$ (ולא פונקציה יוריסטית המנבאת את התוצאה).