תרגיל בית 2 – PACMAN

מגישים: בינת זיסר 206078040
נתניאל ג'וזף 302305396

חלק א' – היכרות עם הקוד והמשחק

שאלה 3 –

נשים לב כי לפי הקוד ה-ReflexAgent מסתכל רק שלב אחד קדימה באופן הבא:
- ראשית נחשב את כל הצעדים החוקיים האפשרים מהנק' שבה אנו נמצאים.
- נעבור על צעדים אלו ונסתכל מה תיהיה התוצאה שלנו במשחק אילולא היינו מתקדמים בצעד זה.
- נבחר צעד בצורה רנדומלית מתוך קבוצת הצעדים המקסימלית.

ההיוריסטיקה שבה הפאקמן הגבר שלכם משתמש היא לשאוף לציון הכי גבוה שהוא יכול.

חלק ב' – בניית סוכן משופר

שאלה 1 –

הפרמטרים שבחרנו להתחשב בהם בהיוריסטיקה (הוספנו מלא פרמטרים כי מימשנו פה כבר את סוכן התחרות):
- המרחק של הפאקמן לפרי הקרוב ביותר.
- המרחק של הפאקמן לפרי הרחוק ביותר.
- מספר הקפסולות על לוח המשחק.
- המרחק של הפאקמן מרוחות.
- הציון של המשחק.

שאלה 2 –

ההיוריסטיקה שלנו מתבססת על כך שלמות לא משתלם – זו גם ההיוריסטיקה שלנו בחיים האמיתיים.
מרחק מנהטן מהפירות: אנו מתחשבים במרחק בין הפאקמן לפרי הקרוב ביותר והרחוק ביותר.
המרחק לפרי הרחוק ביותר זהו בעצם הפרי שהכי קשה לפאקמן שלנו להגיע אליו והקרוב ביותר זהו הפרי המשתלם והזמין ביותר – ושוב, כמו בחיים האמיתיים, כנראה שנגיע לאוכל הכי קרוב אלינו קודם ונתבאס מאוכל שנמצא רחוק מאיתנו.
מרחק מנהטן מהרוחות: מכיוון שהפאקמן שלנו לא רוצה למות, אנחנו לא נרצה להתקרב לרוחות יותר מידי, אך גם לא נרצה לברוח מרוחות שבמילא נמצאות רחוק מאיתנו ולא יכולות להשפיע עלינו, לכן ניסינו למצוא את האיזון בין השניים.
קפסולות: אנחנו לא כל כך יודעים האם משתלם לפאקמן שלנו "לרדוף" או לא אחרי קפסולות ולכן נרצה לתת משקל כלשהו לקפסולות.
אנו לא רוצים שהפאקמן יבחר ללכת לקפסולות ויפסיד אוכל בגלל זה. לכן נתחשב במספר הקפסולות על גבי הלוח כדי שבמקרה והפאקמן כן יהיה קרוב הוא יבחר לאכול את הקפסולה אך אם הוא לא יהיה קרוב פרמטר זה ישפיע בצורה שווה על כל הבחירות.
נשים לב כי ברגע שהפאקמן אוכל את הקפסולה אנו ניתן משקל שונה למרחק מהרוחות מכיוון שכעת הם נותנים לנו נקודות במקום להרוג אותנו.
ציון כולל במשחק: אנו שואפים לקבל את הציון המקסימלי וזה נראה לנו אחלה של בסיס להתחיל ממנו.

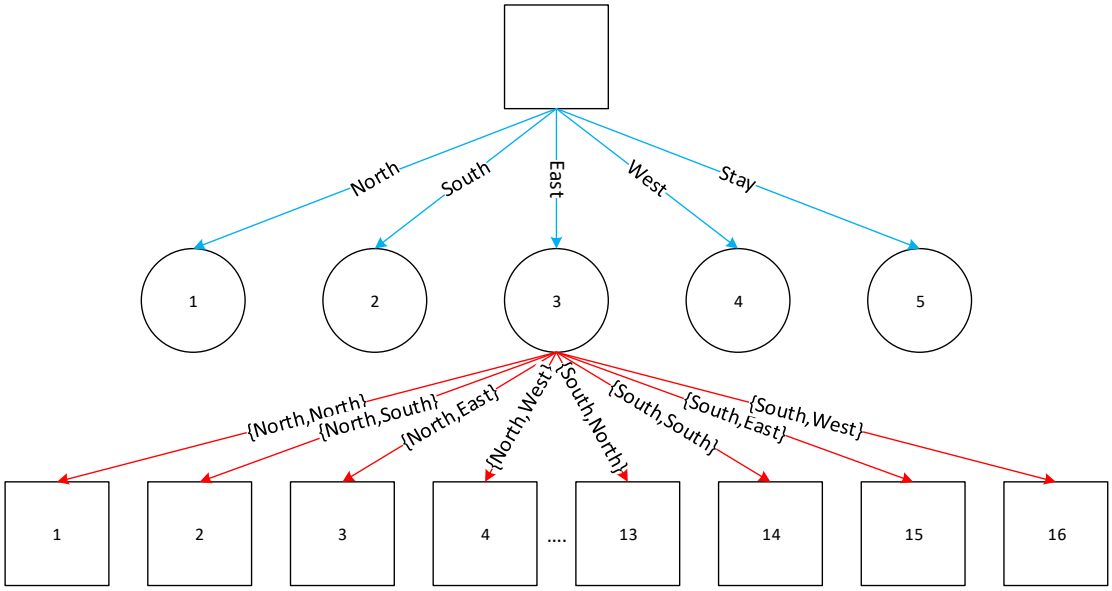
חלק ג' – בניית סוכן Min-Max

שאלה 1 –

ההנחה היא שתמיד הרוחות בוחרות את הצעדים בסדר עולה לפי מספר הסוכן שלהם ולא כולם יחד.
הנחה זו אינה נכונה מכיוון שכל הרוחות פועלות יחד ולא סוכנים נפרדים והפעולות שהרוחות עושות צריכות להיות לטובת הקבוצה ולא לטובת כל רוח בנפרד. כלומר אנו נבחר את הצעד המינימלי הראשון לפי הרוח הראשונה.
לדוגמא במצב שכל הרוחות נמצאות אחת אחרי השניה בטור מצד ימין של הפאקמן, יכול להיות שלפי ההיוריסטיקה שלהן הן כולן ינסו להתקדם לאותו כיוון וישארו בטור. אך אם הן היו מחליטות ביחד הן היו מבינות שזה פחות יעיל ומנסות להקיף את הפאקמן המסכן והחמוד שלנו מכל הכיוונים.
אך נשים לב שזה לא משנה את התוצאה הסופית.

שאלה 3 –

נפתח שכבה אחת משותפת לכל הרוחות שכל אחד מהבנים שלה מורכב מכל אחד מהקומבינציות האפשריות של המהלכים של כל אחת מהרוחות, כלומר אם היו לנו 2 רוחות אז העץ היה נראה כך:



בשכבת המקסימום אנו בוחרים את הצעד הבא של הפאקמן שלנו. כלומר בהנתן 4 כיוונים, עבור כל צומת נפתח 16 (4^{ghosts}) צמתים – כל קומבינציה אפשרית של המהלכים של של הרוחות ונבחר מתוכם את האופציה המינימלית שאנו רוצים.

היתרון של שיטה זו הינו שנקבל מספר מועט יותר של פיתוחי צמתי מינימום (כי אין לנו את הפיצול עבור כל שכבה נוספת של מינימום).
החסרון של שיטה זו הינו שאנו נצטרך להגיע עד לשכבת המקסימום הבאה בשביל לדעת שהפסדנו כאשר בשיטה עם כמה שכבות מינימום נוכל להבין זאת עוד באמצע שכבות המינימום ולהפסיק את הפיתוח כבר משם (כשמתים מחזירים את ההיוריסטיקה).

חלק ד' – בניית סוכן Alpha-Beta

שאלה 1 –

באלגוריתם אלפא-בטא הגיזום מתבצע בין שכבת המינימום למקסימום ולכן במבנה העץ החדש אנחנו נצטרך לבצע יותר פיתוחים מיותרים עד שנגיע לשכבה ממנה יתבצע הגיזום, לעומת שכבת מינימום אחת שנוכל לבצע גיזום בין כל שכבה ושכבה. בנוסף עבור שכבת מינימום אחת, לכל צומת מינימום יש הרבה יותר בנים ולכן עבור פיתוח של תת עץ אחד עבור צומת מינימום כלשהו נוכל לקבל מידע עבור הרבה יותר גיזומים פוטנציאלים.

שאלה 3 –

א. מבחינת זמן ריצה – ברור, זה כל הקטע של האלגוריתם אלפא-בטא. זמן הריצה משתפר משמעותית משום שהאלגוריתם מונע פיתוחים של צמתים שבמילא לא נבחר ובכך חוסך בזמן הריצה.
ב. מבחינת בחירת מהלכים – בחירת המהלכים תיהיה גרועה יותר. זאת משום שהאלגוריתם מניח שהרוחות יבצעו את המהלכים המקסימלים מבחינתם (המינימלים מבחינתנו), אך הרוחות רנדומליות ולכן זהו לא המצב ויכול להיות שנגזום ענף מכיוון שהצעד של הרוח תיהיה לטובתנו אך זה באמת הכיוון שהרוח תבחר ולכן בעצם לא נפתח בכלל את תת העץ הרלוונטי לנו. בזמן ריצה הבנו שבחירת המצבים היא לרוב אותו הדבר.

חלק ה' – בניית סוכן Expectimax לרוח רנדומלית

שאלה 2 –

בניגוד ל-Minimax שמפתח את הצמתים ובוחר מתוכם את התוצאה הטובה ביותר, אלגוריתם Expectimax מתחשב גם בסבירות של הרוח לבחור את המהלך הנתון.
הציפייה שלנו מהתוצאות הן שה-Expectimax יפחד פחות מרוחות ויתקרב אליהם יותר, או לחילופין לא יתרחק ממקומות סתם כי הוא חושב שרוח תגיע לשם.
במקרה בו יש מהלך מאוד גרוע של הרוח בעומק 4.
ב-Minimax אנו לא מתחשבים בכך שהמהלך קורה בעומק 4 ולכן אנו נמנע מהפאקמן להתקדם בעץ לכיוון מהלך זה.
ב-Expectimax אנו מכפילים כל פעם את המהלך בהסתברות לבחירתו. מכיוון שהמהלך נמצא בעומק 4 אנו נוריד את ערכו בכך שנכפיל את ההסתברות לבחירתו בהסתברות לבחירת כל המהלכים שהובילו אליו. ובכך, הוא ישפיע עלינו פחות בבחירת המהלך של הפאקמן בשכבה הראשונה בעץ.

חלק ו' – בניית סוכן Expectimax לרוח לא רנדומלית

שאלה 1 –

נחשב עבור כל צעד אפשרי את המרחק מהפאקמן.
אם אנחנו במצב רגיל (לא מפחד) – נבדוק מה המרחק מבין הצעדים שביצענו ונשמור את הערך המינימלי. ניתן לכל הצעדים בעלי ערך מינימלי זה את ההסתברות הנתונה למצב (0.8) חלקי מספר הצעדים בעלי ערך זה. כעת נעבור על כל הצעדים ונוסיף להם את המשלים (0.2) חלקי מספר הצעדים הכולל ולבסוף ננרמל בשביל שסכום הערכים שקיבלנו יהיה שווה ל-1.
כלומר אנו נותנים הסתברות גבוהה וזהה לצעדים שהכי מקרבים אותנו לפאקמן והסתברות נמוכה וזהה לשאר הצעדים – האסטרטגיה במצב זה היא שהרוח תנוע לכיוון הפאקמן (לפי התוחלת).
אם אנחנו במצב מפחד – נבצע את אותו הדבר עם השינוי היחידי שאנו מחפשים את המרחק המקסימלי. האסטרטגיה במצב זה היא שהרוח תתרחק מהפאקמן (לפי התוחלת).

שאלה 3 –

ההבדל במימושים הינו בחישוב ההסתברויות לרוחות. עבור הרוח הרנדומלית ההסתברות לכל צעד הינה $\frac{1}{numOfMoves}$, עבור רוח לא רנדומלית ההתפלגות היא כפי שתוארה בסעיף הקודם.

שאלה 4 –

שיפורים להיוריסטיקה של הרוחות:

- התחשבות במרחק משאר הרוחות: בכך שנתחשב במרחק משאר הרוחות נוכל לגרום לרוחות להיות יותר פזורות במרחב ולהקיף את הפאקמן מכל מיני כיוונים. כמו כן, אפשר לגרום להם להתפרס על אותו ציר ואז להתקדם כמו מסרק לכיוון הפאקמן.
- התחשבות במיקום הפאקמן: נתקוף את הפאקמן כאשר הוא נמצא בפינות המרחב, אלו המצבים בהם לפאקמן הכי קשה לברוח מהרוחות.

חלק ז' – ניסוח השערות במשחק פאקמן

H_0- סוכן ה-Minimax יותר טוב מסוכן ה-Reflex.
H_1- סוכן ה-Reflex יותר טוב מסוכן ה-Minimax.

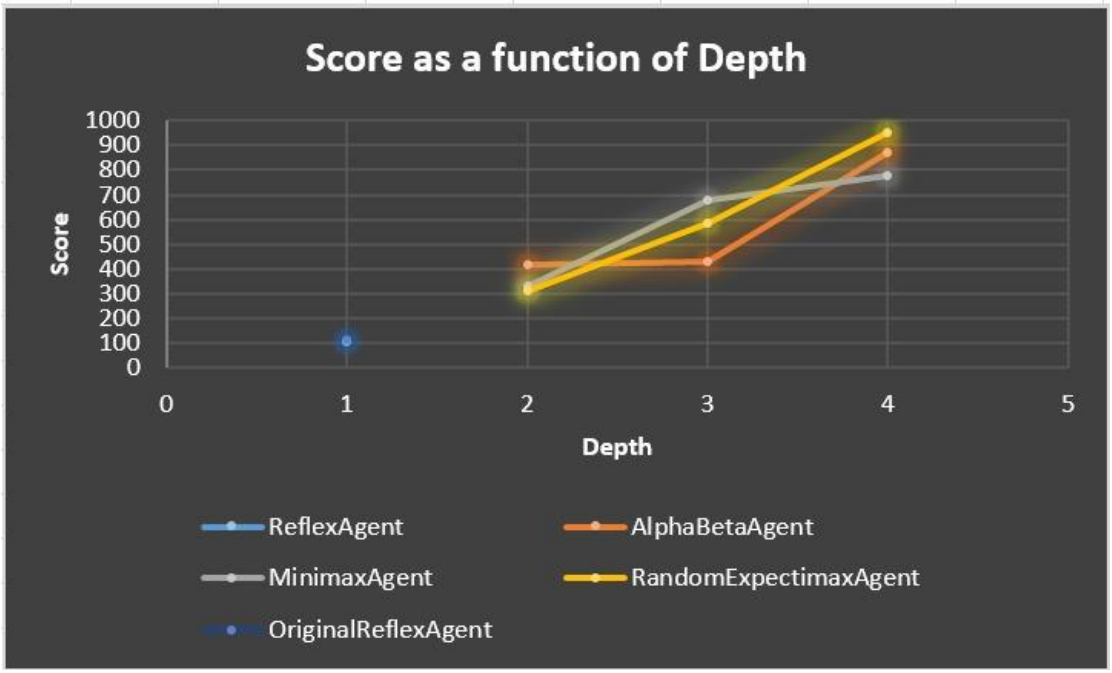
המבחן שנבצע הינו הרצת 1000 משחקים על כל אחד מהסוכנים עבור כל הלוחות, כאשר הריצות מבוצעות עם הדגל $-f$ בשביל שכל הריצות יהיו דומות אחת לשניה עבור הסוכנים השונים.

לא נדחה את H_0 אם נקבל כי סוכן ה-Minimax קיבל תוצאה גבוהה יותר או מת פחות (תלוי מה ההגדרה של "טוב יותר") במעל מ-50% מהמשחקים.

חלק ח' – ניסויים, תוצאות ומסקנות

שאלה 2 –

גרף:



טבלה:

	1	2	3	4
OriginalReflexAgent	113.328			
ReflexAgent	109.372			
AlphaBetaAgent		415.742	427.957	871.871
MinimaxAgent		331.828	680.429	780.471
RandomExpectimaxAgent		314.173	585.615	951.83

שאלה 3 –

המסקנות שלנו מהגרף הינן שדירוג הסוכנים לפי תוצאות הינו כדלהלן:

$$Reflex < Original \ll Minimax < AlphaBeta < RandomExpectimax$$

כמו כן ניתן לראות כי התוצאות משתנות עבור העומקים השונים וכי העומק הטוב ביותר הוא עומק 4. ניתן לראות שככל שהעומקים גדלים התוצאות גדלות.

הציפיות שלנו היו שהגרפים לא יצאו ככה בכלל. ציפינו שהסידור יהיה:

$$Original \ll Reflex \ll AlphaBeta \leq Minimax \leq RandomExpectimax$$

כמו כן, כאשר בנינו את ההיוריסטיקה בנינו מערכת לומדת שבודקת את המשקלים של ההיוריסטיקה ומשנה אותם בשביל להגיע לאחוזי ההצלחה הכי טובים – אך הטעות שעשינו הייתה שהרצנו את המערכת רק על ה-ReflexAgent ורק על המפה mediumClassic ולכן המשקלים שנמצאו הותאמו במיוחד למגרש הספציפי ולהתמודדות עם 2 רוחות (0.92 אחוז הצלחה עבור 1000 ריצות עם ה-ReflexAgent), וכאשר הרצנו את שאר השחקנים או ניסינו להתמודד עם מגרשים אחרים הבנו את הטעות החמורה שעשינו בכך שהתאמנו אותו יותר מידי והוא לא היה מספיק להתמודד עם תנאי מגרש שונים.
כמו כן, שאר האלגוריתמים שמסתכלים מעבר לשלב אחד קדימה גם נתקלו בבעיות שההיוריסטיקה המותאמת ל-Reflex גרמה להם. ככל שהגדלנו את העומק אז באמת השחקנים היו חכמים יותר (המינימקס והאלפא פחדנים יותר, האקספקטימקס היה אגדה אבל לקח לו זמן לרוץ) ובמפה mediumClassic הם באמת שיפרו את התוצאות כפי שציפינו, אך בשאר המפות הם תפקדו פשוט גרוע וסירבו לאכול את האוכל האחרון על המגרש ופשוט נתקעו לידו בלולאה אין סופית עד שרוח באה לסביבתו.

תוצאות הריצה על המפה $mediumClassic$:

Player \ Depth	1	2	3	4
OriginalReflexAgent	74.71			
ReflexAgent	963.71			
AlphaBetaAgent		885	1727.43	1679.71
MinimaxAgent		1727.14	1732.14	1742.86
RandomExpectimaxAgent		1188.57	1883.43	1982.14

לא הצלחנו לתקן את הבעיה בלי לפגוע דרמטית בביצועים, אך הוספנו תנאי שהפאקמן פשוט יתאבד אם התוצאה שלילית מידי (כלומר נתקע להרבה זמן) ובכך הצלחנו למנוע לולאות אינסופיות ולמנוע סטייה גדולה מידי של התוצאות – נבע מכך שקיבלנו באחת הריצות תוצאה של 4,000,000-.

קיצר באסה שלא חשבנו להריץ ישר על סוכן אחר ועל כל המפות... אבל איזה תותח הריפלקס אההה

שאלה 4 –

גרף:

טבלה:

	1	2	3	4
OriginalReflexAgent	0.00016			
ReflexAgent	0.000541			
AlphaBetaAgent		0.004215	0.016098	0.063522
MinimaxAgent		0.007345	0.048866	0.410902
RandomExpectimaxAgent		0.007935	0.051102	0.38681

שאלה 5 –

המסקנות מהתוצאות שקיבלנו הינן כי דירוג זמני הריצה של הסוכנים הינו כדלהלן:

$$Original \ll Reflex \ll AlphaBeta < Minimax \approx RandomExpectimax$$

ניתן להסיק כי אלגוריתם האלפא בטא באמת עובד כמו שצריך ובאמת גוזם ענפים. נשים לב כי ככל שהעומק גדול יותר, כל הגזימה יותר משמעותית ומשפרת את זמני הריצה היחסיים.
כמו כן, ניתן לראות כי זמני הריצה של ה-Minimax וה-RandomExpectimax דיי זהים, זאת משום שהם מבוססים על אותו אלגוריתם ומפתחים עץ שלם כל תור.

ציפינו שאלגוריתם ה-RandomExpectimax יהיה מהיר יותר מאלגוריתם ה-Minimax מכיוון שהוא פחות פחדן ולכן יעז לאכול אוכל מהר יותר או לחילופין ימות כי הרוח תבחר במהלך עם ההסתברות הנמוכה), אך ניתן לראות כי ברוב הפעמים זמני הריצה שלהם יחסית זהים. אנו חושבים (וגם ראינו את זה קורה) שהדבר נובע מכך שההיוריסטיקה שוב מותאמת יותר מידי ל-Reflex והאלגוריתמים נתקעים ליד האוכל האחרון, כשהדבר קורה ב-Minimax והרוח מתקרבת אליו מספיק והפאקמן יאכל את האוכל כי הוא יפחד שהרוח תצליח לאכול אותו במהלכים הבאים. ה-Expectimax יודע להעריך את הסיכוי שהרוח תגיע אליו ולכן הוא לא ימהר לאכול את האוכל האחרון והרוח תצטרך להתקרב אליו יותר בשביל שזה יקרה – מה שגורר זמני ריצה ארוכים יותר למרות השיפור.

שאלה 6 –

	DirectionExpectimaxAgent		RandomExpectimaxAgent	
	RandomGhost	DirectionalGhost	RandomGhost	DirectionalGhost
1	2099	-30	1297	1949
2	1932	573	2743	725
3	2783	154	1244	2228
4	2315	943	2366	-389
5	-415	861	2225	1180
avg score	1742.8	500.2	1975	1138.6

עבור הרוחות הרנדומליות – אכן קיבלנו תוצאות התואמות את המצופה. הפאקמן שיותר מותאם לרוחות הללו ניבא טוב יותר את כיווני הרוחות ואכן השיג תוצאה טובה יותר.
עבור הרוחות הלא רנדומליות – קיבלנו תוצאות הפוכות ממה שציפינו, מהתבוננות במשחקים (וואו זה שנים כל תזוזה), גלינו שה-Directional אכן מנבא את כיווני הרוחות כמו שצריך אז מכיוון שיש 4 רוחות הם התקדמו אליו והוא ברח מהם ופחות התעסק בלהשיג תוצאה גבוהה לעומת ה-Random שבאמת פחות ברח מהרוחות אך יכל להתעסק בהשגת אוכל ואפילו הצליח לנצח (הרצנו עם f– אז המשחקים אותם משחקים)

שאלה 7 –

	1	2	3	4
OriginalReflexAgent	223.86			
ReflexAgent	-62			
AlphaBetaAgent		219	511.29	367.71
MinimaxAgent		359.29	365.29	512.71
RandomExpectimaxAgent		365.43	367.29	371.86

נתייחס רק לעמודה עם עומק 4.
מה שמאפיין את הלוח הינו שיש קצת מאוד אוכל והרבה רוחות ובעצם רק כיוון אחד אפשרי ללכת אליו בכל פעם.
נשים לב שההיוריסטיקה שלנו מותאמת בכלל לריפלקס ולכן הינה פועלת היטב בעומק 4 – הרצנו עוד 5 ריצות וכל השחקנים קיבלו בדיוק את אותה תוצאה וכשהרצנו את הריפלקס 5 פעמים הוא ניצח יותר מהאחרים.
בעיקרון מה שאמור לקרות זה שעבור ה-Minimax וה-AlphaBeta בעומק 4 תיהיה דרך שהוא ימות ולכן בחירת הכיוונים שלו תיהיה רנדומלית, לעומת ה-Expectimax שיודע להתחשב בהסתברויות לבחירת הכיוונים ולכן יצליח להעדיף כיוון אחד על השני – ניתן לראות זאת קורה בצורה טובה יותר ככל שמקטינים את העומקים.

שאלה 8 –

	1	2	3	4
OriginalReflexAgent	-501.43			
ReflexAgent	88.86			
AlphaBetaAgent		-501	-501	-501
MinimaxAgent		-501	-501	-501
RandomExpectimaxAgent		88.86	-206.57	88.86

מה שמאפיין את הלוח הינו שיש 2 רוחות מכל כיוון של הפאקמן והפאקמן בעצם צריך לבחור כיוון כך שהוא לא ימות בזמן שהם מצמצמים את המרחק אליו ויצליח לברוח אם זה אפשרי.

כפי שהוסבר בסעיף הקודם, ה-Minimax וה-AlphaBeta יבחרו כיוון רנדומלי ובעצם יתאבדו. כאן ניתן לראות בצורה הטובה ביותר את התנהגות ה-Expectimax שאכן חוזה בצורה טובה יותר את הכיוון הנכון ללכת אליו.

<u>שאלה 9 –</u>

לפי התוצאות, ניתן לראות שהיוריסטיקה שאנו מימשנו טובה יותר מהיוריסטיקה הבסיסית.
היוריסטיקה שלנו מנסה לא להתקרב לרוחות אשר קרובות מדי (ניסינו גם שהשחקן שלנו לא יהיה סתם פחדן שבורח מכל רוח – לכן אמרנו לו שרק ממרחק מנהטן קטן מ2 יתחיל לברוח מהרוח), לאכול כמה שיותר פירות וקפסולות (כשצריך) וכמובן מנסה להביא את הפאקמן שלנו לנצח בציון מרבי כמה שיותר.

כעת, נעבור שחקן שחקן וננתח באיזה מפה הוא הצטיין ובאיזה מפה הוא התרסק:

<u>– OriginalReflex</u>
זהו השחקן שקיבלנו. זהו השחקן "הטיפש" ביותר שלנו. הוא רואה רק צעד אחד קדימה ובנוסף היוריסטיקה שלו היא נטו מצב הנקודות.

המפה שבה הוא הצטיין: openClassic
המפה שבה הוא התרסק: trappedClassic

מגבלת עומק: אין מגבלת עומק מכיוון שאנחנו רואים רק צעד אחד קדימה – כלומר העומק שווה ל1.
מגבלת זמן: אם היתה מגבלת זמן, היא לא היתה משנה כנראה את התוצאה כי השחקן הזה די "טיפש" ולכן הוא גם יחסית מהיר ולא נתקע.

השחקן הזה הצטיין במפה כי אין בה קירות ויש המון אוכל. אך הוא קיבל ציון שלילי במפה השניה מכיוון שמפה זו מלאה בקירות אשר הוא לא יודע להתמודד איתם.

<u>– Reflex</u>
זהו שחקן אשר רואה צעד אחד קדימה אך בעל היוריסטיקה אשר אנו מימשנו לכן הוא מוצלח יותר מאשר OriginalReflex.

המפה שבה הוא הצטיין: openClassic
המפה שבה הוא התרסק: smallClassic

מגבלת עומק: אין מגבלת עומק מכיוון שאנחנו רואים רק צעד אחד קדימה – כלומר העומק שווה ל1.
מגבלת זמן: אם היתה מגבלת זמן, היא היתה משנה מאד את התוצאות. הבעיה העיקרית עם השחקן הזה שלא הצלחנו להתגבר עליה עם היוריסטיקה שלנו זה המצב שנשאר אוכל אחד והוא נתקע לנו הולך ימינה שמאלה ליד האוכל הזה. אם היתה מגבלת זמן אשר היתה עוצרת אותו מלהיתקע בלופ אינסופי במצב זה, היה יכול להעלות לנו את הציון מאד.

<u>– Minimax</u>
זהו שחקן שיודע לראות לעומק רב יותר מהקודמים. כלומר שחקן זה הוא שיפור מהקודמים.

נסתכל על השחקן במכלול לכל העומקים ונבחר את המפות שהוא היה טוב בהן:
המפה שבה הוא הצטיין: mediumClassic בעומק 4
המפה שבה הוא התרסק: openClassic בעומק 2

מגבלת עומק: ככל שנעלה את העומק כך השחקן אמור להפוך להיות יותר חכם ולדעת להתחשב יותר בסכנות ובאכילת פירות.
מגבלת זמן: אם היתה מגבלת זמן במקום עומק, זה היה משפיע על השחקן הזה כי ככל שאנחנו נכנסים לעומק יותר גדול – הוא מתחיל לחשב יותר דברים וכל תור שלו לוקח יותר זמן. לכן, צריך לעשות בשחקן זה שיווי משקל בין כמה עומק צריך להיכנס לעומת הזמן שהוא יבזבז.

– Alpha-beta
שחקן זה הוא שחקן משופר של הקודם (מינימקס). הוא גוזם ענפים שהוא מבין שהוא לא צריך לפתח כדי לזוז יותר מהר. התוצאות שלו ושל המינימאקס אמורות להיות אותו הדבר רק המהירות של אלפא-בטא אמור להיות יותר גדולה.

נסתכל על השחקן במכלול לכל העומקים ונבחר את המפות שהוא היה טוב בהן:
המפה שבה הוא הצטיין: originalClassic בעומק 4
המפה שבה הוא התרסק: openClassic בעומק 3

* ישנה אי התאמה בין הציפייה שלנו לכך ששני השחקנים יהיו זהים מבחינת מפות מכיוון שכמו שהסברנו בסעיפים קודמים האלגוריתם מניח שהרוחות יבצעו את המהלכים המקסימלים מבחינתם (המינימלים מבחינתנו), אך הרוחות רנדומליות ולכן זהו לא המצב ויכול להיות שנגזום ענף מכיוון שהצעד של הרוח תיהיה לטובתנו אך זה באמת הכיוון שהרוח תבחר ולכן בעצם לא נפתח בכלל את תת העץ הרלוונטי לנו.

מבחינת הגבלת זמן ועומק אותו הדבר כמו minimax.

– RandomExpectimax
זהו שחקן אשר משתמש באלגוריתם אשר נלמד בתרגול ובמשחק עם רוח רנדומלית.

נסתכל על השחקן במכלול לכל העומקים ונבחר את המפות שהוא היה טוב בהן:
המפה שבה הוא הצטיין: mediumClassic בעומק 4
המפה שבה הוא התרסק: openClassic בעומק 2

מגבלת עומק: כמו בקודמים, ככל שהעומק שלו גדל כך הוא מצליח יותר "לראות קדימה" ולהבטיח לעצמו ניקוד גבוה יותר.
מגבלת זמן: חישוב ההסתברויות לכל צעד לוקח לנו יחסית הרבה זמן חישוב, ולכן אם יגבילו לנו את הזמן יכול להיות שהשחקן שלנו יצבור פחות ניקוד כי יחתכו אותו באמצע משחק.

כעת, נסתכל על הלוחות: (נדגיש שהרצת המשחקים היתה עם דגל אשר מבטל רנדומליות ולכן התוצאות הם על אותם משחקים רק עם סוכנים שונים.)

capsuleClassic –
השחקן המצטיין: Alpha-beta בעומק 4.
לפי מה שניתן לראות מהתוצאות השחקנים שלא צלחו את הלוח הזה הם אלה שרואים צעד אחד קדימה. והשחקנים שכן הצליחו הם אלה עם העומק הגדול ביותר.
ניתן להסביר זאת ע"י כך שזהו לוח קשה ויחסית בלתי פתיר במבט אחד קדימה.

contestClassic –
השחקן המצטיין: RandomEcpectimax בעומק 4.
בלוח זה שוב, העמקה נתנה עדיפות לשחקן. אין דברים מיוחדים בלוח זה.

mediumClassic –
השחקן המצטיין: RandomEcpectimax בעומק 4.
בלוח זה שוב, העמקה נתנה עדיפות לשחקן. אין דברים מיוחדים בלוח זה.

minimaxClassic –
השחקן המצטיין: Minimax בעומק 4.
מה שמאפיין את הלוח הינו שיש קצת מאוד אוכל והרבה רוחות ובעצם רק כיוון אחד אפשרי ללכת אליו בכל פעם. בסעיף 7 דיברנו עליו בהרחבה.

openClassic –
השחקן המצטיין: OriginalReflex בעומק 1.

מפה זו מהווה את העקב אכילס של היוריסטיקה שלנו. בשל הבעיה שתיארנו בסעיפים קודמים, שלפעמים נתקע לנו הפאקמן ליד האוכל האחרון עד שהוא לא מתאבד על איזה רוח או שרוח דוחפת אותו לפרי זה. מצב זה יכול להיגרר המון זמן ובמפה זאת שהכל פתוח ואין שום סיבוכים, דווקא שם נופלת לנו היוריסטיקה והפאקמן שלנו מתחרפן. לכן, הפאקמן המצטיין הוא דווקא שלכם !☹

– *originalClassic*
השחקן המצטיין: Alpha-beta בעומק 4.
בלוח זה שוב, העמקה נתנה עדיפות לשחקן. אין דברים מיוחדים בלוח זה.

– *smallClassic*
השחקן המצטיין: RandomEcpectimax בעומק 4.
מפה קטנה אך היוריסטיקה שלנו עובדת איתה טוב ולכן אנו שוב רואים שהעומק נותן עדיפות לשחקן.

– *testClassic*
השחקן המצטיין: Reflex בעומק 1.
כל השחקנים שלנו קיבלו כמעט את אותו הציון בלוח זה. אין כ"כ עדיפות להעמקה בלוח זה כי ישנה רק רוח אחת והלוח יחסית קטן.

– *trappedClassic*
השחקן המצטיין: Reflex בעומק 1.
הרחבנו עליה בסעיף 8.

– *trickyClassic*
השחקן המצטיין: Alpha-beta בעומק 4.
בלוח זה שוב, העמקה נתנה עדיפות לשחקן. אין דברים מיוחדים בלוח זה.

לסיכום, התוצאות שקיבלנו קצת אכזבו אותנו. אך הם נגרמו בשל הטעות הקריטית ביותר שלנו בתכנון האסטרטגיה שלנו לניצחון כאשר ייעלנו את השחקן Reflex על סוג לוח ספציפי וכך הוא הפך להיות מצטיין בלוח זה אך גרוע בלוחות אחרים.
בנוסף כפי שצפינו באמת ככל שהעומק יותר גבוה כך השחקנים יודעים לשחק טוב יותר כי הם צופים את הסכנות שלהם. אם היתה מגבלת זמן במקום עומק זה היה משפיע עלינו בכך שהיינו צריכים למצוא עומק לכל שחקן שבו הוא מצליח לסיים את המשחק בזמן המוקצב כי ככל שהעומק גבוה יותר כך ישנם יותר חישובים וחישובים עולים לנו בזמן יקר.

חלק ט' – תחרות בקורס

השחקן שרצינו להגיש לתחרות הוא ה-ReflexAgent עם ההיוריסטיקה שבנינו לו. רצינו להגיש אותו בגלל מגבלת הזמן של 30 השניות וכי לדעתנו ב-30 שניות היתרונות של האלגוריתמים הקיימים נופלים מהחסרון של זמן הריצה שלהם, בנוסף כפי שהוסבר בשאלות קודמות, ביצענו לו אופטימיזציות משוגעות ואדירות.

את המשקלים לפרמטרים שיפורטו בהמשך בחרנו בעזרת לימוד של וקטור המשקולות ומציאת הערכים האופטימליים (במפה mediumClassic) בצורה הבאה:
שינינו את קובץ pacman.py בשביל לתמוך בלימוד הוקטור, שמנו את כל הערכים ההתחלתיים (שנמצאו על ידי הגיון וניסוי וטעייה) של המשקלים השונים בוקטור ורצנו בלולאה אינסופית על כל משקל בוקטור הזה, כאשר כל פעם שאנו עוברים על משקל מסויים אנו מייצרים 2 וקטורי משקלים חדשים. באחד מהם אנו מוסיפים 10% ובשני מורידים 10% מאותו משקל. לאחר מכן, מריצים 200 משחקים על כל אחד מהוקטורים. אם חל שיפור באחד מהוקטורים באחוזי ההצלחה (איפשרנו גם ירידה קטנה למקרה שהגענו למקסימום מקומי) אז נחליף את הוקטור המקורי בוקטור זה. רצנו עם דגל f– בשביל שכל הריצות שלנו יהיו זהות ובכך הורדנו את אפקט הרנדומליות ובאמת מצאנו את הוקטור הטוב ביותר. אם ביצענו סיבוב שלם על הוקטור ולא נמצא שיפור אז העלנו את מספר הפרמטרים שמשנים ב-1. בחרנו את הפרמטרים שמשנים בצורה רנדומלית ועשינו עוד סיבוב, וכך בצורה זהה עד שהגענו לשינוי של 4 פרמטרים (היה נראה לנו ערך מספיק טוב). נשארנו בלולאה אינסופית שכל פעם מנסה לשנות 4 פרמטרים רנדומלים עד שהוא ימצא את המשקלים שיגרמו לפאקמן להפוך לסופרמן ☺.
נציין שאם לא היינו לוקחים NLP והוא היה כל כך עמוס אז היינו מבינים מתישהו את הטעות שעשינו (ריצה רק על מפה אחת) והיינו מוצאים משקלים עבור כל המפות וכמובן שמנצחים בקלות ☺ אך מה לעשות שהחיים לא עובדים ככה ושתמייייד כשיש הגרלות ודברים כאלו אין מצב שנזכה (כמו הטיסה לברלין מאינטל).

אך – בגלל שביקשתם סוכן מסוג multiAgentSearch **החלטנו להגיש את RandomEcpectimax בעומק 3** מכיוון שהוא הסוכן מהסוג שביקשתם עם האלגוריתם הכי טוב. לא שלחנו אותו עם עומק 4 כי אז לא היינו עומדים במגבלת זמן הריצה שדרשתם כי לוקח לו הרבה זמן לחשב ואותו עם עומק 2 לא שלחנו כי הוא היה צובר פחות ניקוד.

מקווים שנהנתם מהעבודה שלנו כמו שנהינו לכתוב פאקמן שלא ינצח בתחרות ☺
(הכל כי לא אפשרתם לנו לשלוח את הreflex שהיה מנצח את כולם אבל אנחנו נאהב אותו לעד)

הפרמטרים ששיחקנו איתם:

```
MIN_DIST_FROM_GHOST = 1.4                          # This is the
minimal distance from ghosts we allow the pacman to be.
# We wanted the highest score - so its only logical to use the
score in the heuristics.
SCORE_MULTI = 4                                              #
Multiplies the game score.
GAME_LOSE_BONUS = -200                             # Additional
bonus given when we lose.
# These parameters only work when there isn't a lot of food on
the board (SMALL_AMOUNT_ON_FOOD_*)
# We calculate small amount of food as: foodGrid.width *
foodGrid.height / 10
```

We added these weights because we wanted the pacman to be more Food oriented when the food was low, and because the small food amount made it
harder for the pacman to choose the right direction to the food.
SMALL_AMOUNT_OF_FOOD_MUL = 1 # Multiplier for the amount of food regarded as small amount of food.
SMALL_AMOUNT_OF_FOOD_MANHATTAN_MUL = 1 # Multiplies the average menhatten distance from the food.
SMALL_AMOUNT_OF_FOOD_CLOSE_FOOD_MUL = 0.93 # Multiplies the weight for the closest food.
SMALL_AMOUNT_OF_FOOD_SAME_DIR_MUL = 1.6 # Multiplies the weight for pacman moving in the same direction
SMALL_AMOUNT_OF_FOOD_CLOSE_GHOST_MUL = 0.8 # Multiplies the weight for the ghosts that are close to us.
SMALL_AMOUNT_OF_FOOD_MEDIUM_GHOST_MUL = 0.4 # Multiplies the weight for the ghosts that are in medium distance from us.
SMALL_AMOUNT_OF_FOOD_FAR_FOOD_MUL = 0.525 # Multiplies the weight for the furthest food.
SMALL_AMOUNT_OF_FOOD_FAR_GHOST_MUL = 0.2 # Multiplies the weight for the ghosts that are far from us.
SMALL_AMOUNT_OF_FOOD_SCARED_DIST_MUL = 0.25 # Multiplies the weight for the ghosts that are scared.
SMALL_AMOUNT_OF_FOOD_CAPSULE_MUL = 0.9 # Multiplies the weight for taking a capsule.
In some cases we will want to stop pacman from eating capsules - like in the case that there are scared ghosts and eating the capsule wont change anything.
SCARED_CAPSULE_MUL = -0.5 # Multiplies the bonus we get for eating a capsule while there are still scared ghosts.
CAPSULE_BONUS = -260 # A bonus we get for every capsule in the layout (so eating a capsule will result in a higher score).
Bonuses for ghosts, we also considered far ghosts to add another level to the heuristics that will help pacman when he is stuck.
We wanted to differ between the distances of the ghosts and give them weights according to how dangerous they are to us.
GHOST_SCARED_DIST_MUL = -3.6 # Multiplies the distance regarded when trying to ass
GHOST_EAT_BONUS = 150 # Bonus for eating a ghost.
GHOST_DIST_BONUS = 0.4 # Multiplies the distance from the ghosts.
GHOST_FAR_BONUS = -1 # Multiplies the number of far ghosts.
GHOST_MEDIUM_BONUS = 0.9 # Multiplies the number of medium ghosts.
GHOST_BAD_BONUS = -480 # Multiplies the number of ghosts closer than MIN_DIST_FROM_GHOST

MIN_DIST_MULTIPLIER = 2.7 # Multiplier for MIN_DIST_FROM_GHOST - distance for saying whats a medium ghost is
#
Notice: all other ghosts will be considered far ghosts.

DIST_IF_SCARED_MUL = 1.38 # Multiplier for MIN_DIST_FROM_GHOST considering what in an eatable ghost.
These params are to make pacman be more food oriented and to make him prefer closer food than far food but still not try and find the best weights.
FOOD_FAR_BONUS = -0.5 # Multiplies the distance of the furthests food.
FOOD_CLOSE_BONUS = -5 # Multiplies the distance of the closest food.
NUM_FOOD_BONUS = -10 # Multiplies the total number of food - to make pacman be more food oriented.
NO_FOOD_BONUS = 0 # To help pacman not leave 1 food left alone, considers close food around him - WE FAILED TO MAKE THIS WORK @$#%@%$!#
MANHATTAN_ENABLE = -1.1 # Multiplies the average menhatten distance from the food.

SAME_DIR_BONUS = 2.25 # A bonus given if pacman is continueing in the same vector as last turn.