***חלק ב'***

1. השחקן GreedtAgent בודק עבור כל מהלך אפשרי שלו את כל המהלכים האפשריים של יריביו, ומבין כל השילובים האלו השחקן מכניס לרשימה את כל המהלכים בעלי הערך היוריסטי הגבוה ביותר. לאחר מכן בוחר רנדומלית מהלך מתוך רשימה זו.

הפונקציה היוריסטית בה השחקן משתמש מניחה כי בכל מהלך עתידי של השחקן יאכל פרי עד שנגמר מספר התורות במשחק או עד שייגמרו כל הפירות במשחק (סוכמת גם את אורכי הנחשים מכיוון שייתכן כי יהפכו לפירות אם ימותו). בנוסף, עבור כל צעד עתידי שיותר מתקדם מהצעד הנוכחי נקטין את המשקל של צעד זה בחישוב הפונקציה, מכיוון שהוא פחות וודאי. הפונקציה מחושבת באופן הבא – עבור מקדם הנחה נתון (a < 1) ועבור מספר צעדים אפשרי (k) הפונקציה מחשבת :

הפונקציה מחזירה את תוצאת החישוב הנ"ל + אורך הנחש לאחר הצעד.

הפונקציה היוריסטית מאפשרת להימנע ממוות מכיוון שעבור צעד בו הנחש מת, הפונקציה מחזירה רק את אורך הנחש ללא התוספת העתידית. מכיוון שתוספת זו תמיד גדולה מ-0, במקרה של מוות תוצאת היוריסטיקה תהיה פחות עדיפה על מהלכים אחרים בהם הנחש לא מת, לכן לא תיבחר.

***חלק ג'***

1. נגדיר יוריסטיקה המתחשבת בערכים הבאים:

* snake\_length = the length of the snake in the next state (considering eating fruits or dying)
* distance\_from\_closest\_walls = calculates the distance from the head position of the snake to the closest horizonal wall and vertical wall. This value holds the sum distance from both walls.
* distance\_from\_closest\_fruit = calculates the "Manhattan" distance from the head position of the snake to the closest fruit.
* closest\_rival\_snake = calculates the "Manhattan" distance from the head position of the snake to the position of each rival snake and chooses the minimal distance. Meaning, the distance from the closest point of the rival snake body to the head of the player snake. Then, if this distance is greater than the length from this closest point of the rival snake body to its tail, then by the time the player will potentially reach the rival snake, it wont be at this point anymore, so we don’t take this rival snake into consideration.

הפונקציה היוריסטית מחזירה את המשוואה הבאה:

(w\_of\_len \* snake\_length) + (w\_of\_walls\*distance\_from\_closest\_walls) +  
 (w\_of\_rival\*closest\_rival\_snake) – (w\_of\_fruits\*distance\_from\_closest\_fruit)

כאשר הפרמטרים מהצורה W\_\* מציינים משקל מסוים אותו אנחנו מכפילים בערך המתאים עבור כל פרמטר על מנת לקבל את התוצאה הרצויה.

1. נתייחס עבור הערכים שתוארו לעיל ונסביר את המוטיבציה מאחוריהם:

* אורך הנחש (snake\_length) – כאשר אנחנו מחשבים במהלך הבא של הנחש את אורכו אנחנו בפועל מייחסים חשיבות האם הנחש אכל פרי או לא. לכן אם במהלך הבא יש אפשרות לאכול פרי, הפונקציה היוריסטית נותנת למהלך זה משקל גדול יותר.
* מרחק מהקירות (distance\_from\_closest\_walls) – המוטיבציה מאחורי ערך זה היא שככל שהנחש קרוב יותר לקיר יש פוטנציאל גדול יותר להיתקע ופחות מהלכים טובים אפשריים. לכן הפונקציה היוריסטית תתעדף מהלכים שמרחיקים את הנחש מהקיר.
* מרחק מהפרי הקרוב ביותר (distance\_from\_closest\_fruit) - המוטיבציה מאחורי ערך זה היא שככל שהנחש קרוב יותר לפרי יש פוטנציאל גדול יותר לאכול אותו. לכן הפונקציה היוריסטית תתעדף מהלכים שמקרבים את הנחש לפירות.
* מרחק מהיריב הקרוב ביותר (closest\_rival\_snake) – המוטיבציה מאחורי ערך זה היא שכאשר הנחש נתקל ביריב הנחש מת, וככל שהוא קרוב יותר ליריב סיכויו להיתקל בו גבוהים יותר. הפונקציה היוריסטית לוקחת גם בחשבון את המרחק מהנקודה הקרובה ביותר של היריב אל ראש הנחש ביחס למרחק של אותה נקודה מהזנב של אותו נחש, מכיוון שכאשר הנחש יגיע לנקודה זו היריב כבר לא יהיה שם. לכן הפונקציה היוריסטית תתעדף מהלכים שמרחיקים את הנחש מהיריב.

כל הערכים (פרט לאורך הנחש) שציינו אינם נלקחים בחשבון ביורסטיקה של הנחש הפשוט, לכן לפי המוטיבציה שהסברנו אנחנו צופים כי הביצועים של השחקן החדש ישתפרו.

***חלק ד'***

1. אנחנו צופים כי אלגוריתם ה-MinMax שכתבנו יעבוד טוב בתרחישים בהם יש חשיבות להסתכלות מספר צעדים קדימה. לדוגמא, במצב בו ייתכן כי הסוכן ייכנס ללולאה סביב עצמו שבסופה ייחסם ויאלץ להרוג את עצמו, צפייה של מספר צעדים קדימה יכולה למנוע מהסוכן להכנס לבור זה. כמו כן ייתכן מצב בו אחד היריבים סוגר על הסוכן ויוכל להרוג אותו בעוד מספר מהלכים, ולכן גם כאן צפייה של מספר צעדים קדימה יכול לעזור לסוכן. לעומת זאת, הסתכלות קדימה גם יכולה לפגוע בפעולת הסוכן, למשל כאשר בצעד הבא קיימת אפשרות לסוכן לאכול פרי, אבל במספר צעדים קדימה תתכן אפשרות מנוונת לסוכן להיפגע, אפשרות שממנה הסוכן יוכל להימנע בכל מקרה, לכן במצב כזה פעולת הסוכן פחות טובה.
2. בהנחה שמספר היריבים הוא k לכל צומת מינימום יהיו בנים, זאת מכיוון שניתן להגדיר כל צומת של בן מינימום כווקטור מהצורה - כך ש - כאשר k הוא מספר היריבים. כלומר עבור כל רמה בעץ ה-MinMax יפותחו צמתים לכן סיבוכיות זמן הריצה תהיה כאשר d הוא עומק העץ (העומק שניתן לאלגוריתם).עבור מספר רב (20+) של יריבים לא ניתן בפועל להשתמש באלגוריתם מכיוון שזמן החישוב של כל צעד גדול מאוד.
3. עבור המשחק שלנו כאשר יש יותר מיריב אחד, אנחנו מניחים כי פעולות כל היריבים יחד יזיקו הכי הרבה לסוכן שלנו. אך בפועל ייתכן כי כאשר כל יריב יבחר את המהלך הטוב ביותר בשבילו, סך הבחירות של כל היריבים של הסוכן שלנו יהיו יותר טובות מהמקרה הגרוע ביותר ואולי אף יעזרו לסוכן שלנו. לכן בבחירת המהלך הבא אנחנו מבצעים הנחות שאינן בהכרח נכונות.

***חלק ה'***

1. סוכן האלפא-בטא מבחינת זמן הריצה צפוי לרוץ יותר מהר, מכיוון שלפי עקרון הגיזום שנלמד בשיעור האלגוריתם אינו מפתח תתי עצים שלא יילקחו בחישוב הכולל. לכן סך הפיתוחים בסוכן אלפא-בטא יותר קטן ולכן זמן הריצה הכולל קצר יותר ביחס ל-MinMax הרגיל.
2. סוכן האלפא-בטא מבחינת בחירת מהלכים צפוי להיות זהה לסוכן ה-MinMax הרגיל. זאת מכיוון שבריצת האלפא-בטא נעשה גיזום של מהלכים שלא ייבחרו בכל מקרה, כלומר התוצאה הסופית ששני הסוכנים השונים יחזירו למעשה תהיה זהה.

***חלק ו'***

1. במצב בו אנחנו צריכים לבחור מראש את כל N המהלכים של הסוכן שלנו, נצטרך כי מצב הלוח ההתחלתי ופעולות היריבים לכל אורך המשחק יהיו דטרמינסטים. זאת מכיוון שלצורך חישוב הפעולה ה-i אנחנו נצטרך לדעת את מצב הלוח והיריבים בתור זה. כלומר, על מנת שנוכל לבחור את סדרת N הפעולות הטובות ביותר נצטרך כי הסביבה של הסוכן תהיה ידועה. במידה וסביבה זו אכן הייתה משתנה היינו נתקלים בבעיה שבתכנון סדרת המהלכים היה חסר לנו מידע על מנת לבחור את המהלך הטוב ביותר לכל צעד.
2. המשחק אינו משחק סכום אפס. פונקציית התועלת של כל שחקן מחזירה את אורך הנחש + נקודה נוספת במידה והוא סיים את המשחק חי. כלומר הנקודה שנחש מקבל על היותו חי לא באה על חשבון נחש אחר, ובנוסף כל אכילת פרי שמגדילה את אורכו של הנחש לא באה על חשבונו של נחש אחר. כלומר סכימת הניקוד של כל השחקנים בסוף המשחק אינה אפס ולכן המשחק אינו סכום אפס.
3. כל מצב יהיה וקטור באורך N כאשר כל איבר בווקטור ייצג פנייה ימינה, שמאלה או המשך ישר: כך ש - . סך הכל הווקטור ייצג את בחירת הצעדים של הסוכן לאורך כל המשחק.
4. האופרטורים במרחב החיפוש יהיו בכל איטרציה החלפת איבר מסוים בווקטור ל-. בכל איטרציה אנחנו נשתמש ב-3 אופרטורים עבור שלושת הצעדים האפשריים.
5. לפי האמור לעיל, כמות האיטרציות המבוצעות בחישוב זה הוא N. לכל איבר במערך נבצע איטרצית חישוב לבדוק מה הצעד העדיף מבחינת החיפוש המקומי, נבחר את הצעד המועדף ונעבור לאיטרציה הבאה. כלומר לאחר i איטרציות חושבו i הצעדים הראשונים.
6. נגדיר את הפונקציה היוריסטית להערכת טיב הפתרון כהפרש בין ניקוד הסופי של הסוכן לבין הניקוד הסופי של היריב בעל הניקוד הגבוה ביותר, כאשר נגדיר את הניקוד הסופי להיות אורך הנחש + נקודה אם הנחש חי בסוף המשחק. בעזרת פונקציה זו נוכל למדוד שיפור של מצב הסוכן בסוף המשחק ביחס ליריביו לאחר כל שינוי בכל איטרציה, ונשפר את רצף הפעולות בהתאם.
7. --------