חיפוש רב סוכני - פקמן

מטרות התרגיל

- התנסות באלגוריתמים לחיפוש רב סוכני.
- מימוש אלגוריתמים Minimax, Expectimax ועוד שיפורים והרחבות.
- התנסות בפיתוח היוריסטיקה למשחק פקמן.

מבוא והנחיות

1. במטלה זו תתכננו ותממשו שחקנים למשחק פקמן.
2. את חוקי המשחק ניתן למצוא בוויקיפדיה.

- https://he.wikipedia.org/wiki/%D7%A4%D7%A7-%D7%9E%D7%9F

4. מומלץ לחזור על שקפי ההרצאות והתרגולים בנושא "חיפוש רב סוכני" לפני תחילת העבודה על התרגיל.
5. שימו לב כי ישנם תרגילים רבים בנושא המשחק פקמן באינטרנט אך לא כדאי להתבסס עליהם במהלך פתרון התרגיל כיוון שהתרגיל שאתם מקבלים שונה (גם אם הגרפיקה דומה).
6. במשחק שלנו, פקמן צובר ניקוד לפי השיטה הבאה, כאשר המטרה היא כמובן לסיים את המשחק עם כמה שיותר נקודות:
   - על כל חתיכת אוכל שפקמן אוכל הוא צובר 10 נקודות
   - על ניצחון (ריקון כל האוכל מהלוח) פקמן צובר 500 נקודות - על אכילת רוח (לאחר לקיחת קפסולה) פקמן צובר 300 נקודות
   - על כל מהלך שעובר יורדת לפקמן נקודה
   - על פסילה (הפסד במשחק) יורדות לפקמן 500 נקודות
7. הקוד שאתם מקבלים מכיל את הקבצים הבאים:
   1. submission.py - הקובץ היחיד שעליכם לשנות ולכתוב בו קוד, פה תממשו את השחקנים והאלגוריתמים שלכם.
   2. pacman.py - הקובץ המרכזי המריץ את המשחק. מכיל את הטיפוס GameState בו תעשו שימוש רב.
   3. game.py - הלוגיקה מאחורי חוקי המשחק ולוח המשחק
   4. util.py - מבני נתונים בהם תוכלו להשתמש במימוש אלגוריתמי החיפוש
   5. layout.py - קוד המטפל בפריסות הלוח השונות (הנמצאות בתיקייה layouts)
   6. ghostAgents.py מימוש סוכני הרוחות
   7. קבצי תמיכה שאין צורך להתעמק בהם: graphicsDisplay.py, graphicsUtils.py, textDisplay.py, keyboardAgents.py
8. שימו לב שעליכם לכתוב קוד ולשנות אך ורק את הקובץ submission.py

חלק א' – היכרות עם הקוד והמשחק

1. ראשית, העבירו את התיקייה pacman שסופקה לכם למקום שנוח לכם לעבוד בו במחשב שלכם ופתחו את התיקייה דרך ה ide שאתם עובדים איתו)אנו, כאמור בתרגיל הראשון, ממליצים על pycharm .) אנו נריץ את המשחק דרך ה command line ולכן פתחו במקביל גם את ה command prompt בתיקייה המתאימה.
2. הריצו את המשחק בעזרת הפקודה הבאה: python pacman.py
שימו לב שזוהי הרצה אינטראקטיבית (כלומר, נותנת אפשרות למשתמש להחליט מהו המהלך הבא דרך המקלדת). אתם יכולים להשתמש בה על מנת להבין את המשחק טוב יותר (או סתם לשחק להנאתכם) אך בהמשך התרגיל לא נשתמש בהרצה זו אלה ניתן לסוכן, כלומר לקוד שתכתבו, להיות זה שמחליט על המהלכים הבאים.

- כעת, תנו לשחקן הבסיסי שסופק לכם לשחק. הדגל –p בוחר את הסוכן שישחק את פקמן. הריצו את השורה הבאה:

python pacman.py -p ReflexAgent

- השתמשו בדגל -l על מנת להשתמש בפריסות לוח שונות הנמצאות בתיקייה layouts . אם לא הוגדרה פריסה ברירת המחדל היא הפריסה mediumClassic . הריצו למשל השורה הבאה:

python pacman.py -p ReflexAgent -l testClassic

- דגלים נוספים שכדאי להכיר לקראת המשך התרגיל:

q- : מאפשר כיבוי גרפיקה. טוב להרצה מהירה של מספר משחקים.

n- : דגל המאפשר הרצת מספר משחקים ברצף. מקבל פרמטר של מספר משחקים.

k- : דגל המאפשר להגדיר כמה רוחות יופיעו בלוח. שימו לב שלפריסות שונות יש מגבלה שונה על מספר הרוחות המקסימלי שהן יכולות להכיל. הריצו את הפקודות הבאות:

python pacman.py -p ReflexAgent -k 1

python pacman.py -p ReflexAgent -k 2

python pacman.py -p ReflexAgent -k 12 (הוא מעבר למגבלה לכן המשחק ירוץ עם מספר הרוחות המקסימלי המתאים ללוח זה).

כמו כן, תוכלו להשתמש ב - help- על מנת לקבל את כל האפשרויות לדגלים נוספים. אין צורך לעבור על כל האפשרויות כי חלקן לא רלוונטיות עבור התרגיל שלנו. אם יהיה צורך בדגל מסוים, נציין זאת ספציפית בסעיף שבו צריך להשתמש בו.

3. הסבירו כיצד השחקן הבסיסי ReflexPlayer (הממומש בקובץ submission.py) עובד ומהי ההיוריסטיקה בה הוא משתמש.

חלק ב' – בניית סוכן משופר

אתם נדרשים לממש ההיוריסטיקה יותר מתוחכמת מזו הפשוטה שסופקה לכם עבור השחקן ReflexAgent . על ההיוריסטיקה להיות ממומשת בפונקציה betterEvaluationFunction בקובץ submission.py .

1. הגדירו באופן מפורש היוריסטיקה משלכם להערכת מצבי המשחק. על ההיוריסטיקה להכיל לפחות 3 פרמטרים שונים (אפשר ורצוי למצוא יותר). שימו לב כי ההיוריסטיקה צריכה לעבוד לכל layout של לוח המשחק (כולל מספר רוחות משתנה).
2. הסבירו את המוטיבציה להגדרה מהסעיף הקודם (ולכל פרמטר המופיע בחישובים שלכם). למה אתם צופים שהיא תשפר את ביצועי השחקן ביחס להיוריסטיקה scoreEvaluationFunction בה השתמש השחקן הפשוט עד כה?
3. ממשו את ההיוריסטיקה שהגדרתם ללא תוספות נוספות.

שם הפונקציה שעליכם לממש הוא כאמור betterEvaluationFunction וחתימתה נתונה לכם. היא נמצאת בקובץ submission.py . כמו כן, עליכם לוודא שהשחקן יריץ את ההיוריסטיקה שלכם ולא את scoreEvaluationFunction , עשו זאת ע"י שינוי השורה הקוראת ל scoreEvaluationFunction בתוך ה class של ReflexAgent בפונקציה evaluationFunction כך שבמקומה תקרא הפונקציה betterEvaluationFunction שכתבתם.

טיפים לפני מימוש: עברו על המחלקה GameState בקובץ pacman.py כדי להבין באילו פונקציות אתם יכולים להשתמש כדי לקבל מידע על מצב המשחק. שימו לב- אין לשנות את הקוד בקובץ pacman.py, אך ניתן (ואף צריך) להשתמש בפונקציות שכבר קיימות בו. דרכן תוכלו להשיג את כל המידע שתרצו על מצב הלוח הנוכחי באופן פשוט יחסית. כמו כן שימו לב שקיימות פונקציות שניתן להשתמש בהן גם בקובץ util.py , דוגמה לאחת כזו שיכולה אולי להועיל כבר בסעיף זה היא manhattanDistance

חלק ג' – בניית סוכן Min-Max

נרצה לממש סוכן Min-Max ב class בשם MinMaxAgent אשר בקובץ submission.py .
שימו לב כי לפקמן יכולים להיות מספר יריבים)רוחות רפאים(, כתלות במשחק הספציפי שרץ.
לכן, לא נרצה להשתמש באלגוריתם ה Min-Max המתאים לשני שחקנים (בו רק שכבת מינימום אחת עבור יריב יחיד אחרי כל שכבת מקסימום) אלה בזה הכללי יותר המתאים ליריבים מרובים. בפרט, בעץ ה Min-Max שניצור יהיו מספר שכבות min)אחת לכל רוח רפאים(עבור כל שכבת max (המייצגת את פקמן). שימו לב שהסבר זה מתייחס לכל שארית התרגיל, כלומר גם עבור מימוש הסוכנים המשתמשים באלגוריתמים אחרים (Alpha-Beta, Expectimax) נשתמש בכמה שכבות min עבור כל שכבת max לפי מספר הרוחות.
הבהרה: בסעיף זה ובסעיף הבא, במימוש האלגוריתמים Min-Max ו Alpha-Beta , אנו מניחים שסוכן הפקמן שתממשו אינו יודע מול איזה יריבים (רוחות) הוא מתמודד. מאוחר יותר בתרגיל נראה שלמימוש שקיבלתם צורפו רוחות משני סוגים, אך בשלב זה אנחנו מניחים שפקמן אינו מכיר אותן ויתכן שבבדיקת סעיפים אלה נשתמש ברוח שלישית שאינה מצורפת למימוש שקיבלתם, לכן האלגוריתמים שתממשו צריכים להתאים לכל רוח ובסעיפים אלה לא תוכלו לבצע הנחות כלשהן על התפלגות התנועה של הרוחות.

1. מהי ההנחה שלנו (אחת או יותר) ביצירת העץ כמתואר מעלה, בנוגע לסדר קבלת ההחלטות של הסוכנים במשחק? האם היא בהכרח נכונה? הרחיבו.
2. ממשו את האלגוריתם כפי שהוצג ביחד עם ההיוריסטיקה שיצרתם בחלק ב' ב class שהוגדר.
   וודאו שהשחקן החדש שיצרתם רץ ללא תקלות על הלוחות השונים.
   שימו לב לפרטים הבאים הנוגעים למימוש (רלוונטי גם לסעיפים הבאים):
   - ודאו כי בכל שכבת min אתם מתייחסים לרוח שונה.
   - ודאו כי המימוש שלכם עובד עבור מספר רוחות משתנה.
   - אתם רשאים לממש כל פונקציית עזר שתרצו בתוך ה class של הסוכן, אך אתם חייבים לממש את הפונקציה getAction ואין לשנות את חתימתה (הסבר לגבי מה בדיוק היא צריכה לקבל ולהחזיר תוכלו למצוא בהערות בקוד).
   - שימו לב כי כל הסוכנים המתוחכמים (שתממשו מסעיף זה והלאה) יורשים מהמחלקה MultiAgentSearchAgent - אם תרצו לכתוב פונקציות שרלוונטיות לכל הסוכנים הללו תוכלו לכתוב אותן פעם אחת בלבד במחלקה זו. כמו כן תהיה לכם גישה לשדות המוגדרים במחלקה זו ובפרט לשדות self.evalFn ו- self.depth
   - self.evalFn : שדה זה יחזיק את הפונקציה ההיוריסטית שבה הסוכן ישתמש. הוא מוגדר כברירית מחדל לפונקציה betterEvaluationFunction , תוכלו לשנות זאת אם תרצו.
   - self.depth : שדה זה יחזיק את מגבלת העומק עבור הסוכן- כלומר העומק המקסימלי שהסוכן צריך ויכול לחפש בו בכל תור. במהלך כל התרגיל נגדיר את העומק להיות מספר השכבות בעץ חלקי מספר

הסוכנים. למשל, נניח שיש 3 רוחות במשחק, עץ בעומק 2 יכיל 8 שכבות- 2 שכבות המייצגות כל אחד מארבעת הסוכנים)פקמן ו 3 הרוחות(. הגדרת העומק כך נובעת מהעובדה שמספר הרוחות משתנה ולכן נרצה דרך להגדיר את העומק כסבב שבו כל אחד מהסוכנים)פקמן וכל הרוחות(משחק פעם אחת. ערך ברירת המחדל של הגבלת העומק הוא 2. השאירו את הערך הזה כערך ברירת המחדל. כשתירצו בכל זאת להריץ עם עומק שונה תוכלו לעשות זאת בעזרת הדגל –a depth=num . שימו לב שבסעיפים הבאים לא נגביל אתכם לעומק מסוים (על מנת לזרז את הריצה כדאי לכם רוב הזמן לרוץ עם עומק= 2 , כלומר לא להגדיר עומק אחר ולכן ערך זה יבחר כברירת המחדל), אך בשלב הניסויים ובשלב התחרות נבקש מהסוכנים שלכם לעבוד על עומקים משתנים ולכן כדאי לוודא כבר בשלב זה שהסוכנים רצים (ובזמן סביר) על עומקים 2,3,4 .

-לפני שתתחילו לממש, מומלץ לעשות רשימה של כל הפונקציות שאתם צריכים לצורך המימוש, למשל פונקציה הבודקת אם מצב הוא מצב סופי, פונקציית sucssesor וכו'. לאחר מכן בדקו האם פונקציות כאלה כבר קיימות, בפרט במימוש של GameState . תגלו שאת רובן אין צורך שתממשו, לדוגמה:

- gameState.getLegalActions : מחזירה עבור מצב במשחק את הפעולות האפשריות
- gameState.generatePacmanSuccessor(action) מחזירה עבור מצב נוכחי ופעולה את המצב הבא.

-נזכיר שוב שגם בקובץ util.py ישנם מבני נתונים ופונקציות שיכולות לשמש אתכם במימוש.

3. נרצה לחשוב על דרך נוספת לממש את אלגוריתם Min-Max , כך שלא ניצור שכבה נוספת לכל רוח. הסבירו (במילים, אין צורך לממש) איך הייתם עושים זאת. מה החסרונות והיתרונות של כל אחת מהשיטות?

חלק ד' – בניית סוכן alpha-beta

1. האם מבנה העץ החדש שהגדרנו (כמה שכבות Min עבור כל שכבת max) משפיע על אלגוריתם אלפא-בטא? אם כן-איך? אם לא- מדוע? הסבירו בפירוט.
2. ממשו את אלגוריתם אלפא בטא ב class המתאים (AlphaBetaAgent) בקובץ submission.py .
3. האם הסוכן AlphaBetaAgent שמימשתם בסעיף זה יתנהג שונה מסוכן ה MinMaxAgent שממשתם בסעיף הקודם:

a . מבחינת זמן ריצה? הסבירו מדוע.

b . מבחינת בחירת מהלכים? הסבירו מדוע.

חלק ה' – בניית סוכן Expectimax לרוח רנדומלית

נרצה לבנות סוכן המשתמש באלגוריתם Expectimax ב class בשם RandomExpectimaxAgent אשר בקובץ submission.py .

שימו לב שברירת המחדל במשחק (כאשר מריצים את המשחק עם סוכן ולא באופן ידני) היא שימוש ברוח RandomGhost וזו הרוח איתה נעבוד בסעיף זה. הרוח, כפי שניתן להבין משמה, בוחרת מהלך בהתפלגות רנדומלית מבין כל המהלכים המוחזרים לה מהפונקציה getLegalActions ברגע נתון. סוכני הרוח ממומשים בקובץ ghostAgents.py , הסתכלו במימוש הרוח בקובץ זה על מנת להבין טוב יותר את דרך פעולתה

1. בהינתן התפלגות הרוח הרנדומלית, ממשו את האלגוריתם Expectimax במחלקה RandomExpectimaxAgent .
2. מהו השינוי ביחס לשני הסוכנים הקודמים המשתמשים באסטרטגיית MinMax ?

מהי הציפייה שלכם מהתוצאות שתקבלו בהרצה? תנו דוגמאות של מקרים שונים שיתמכו בציפיות

שלכם.

חלק ו' – בניית סוכן Expectimax לרוח לא רנדומלית

נרצה לבנות סוכן המשתמש באלגוריתם Expectimax ב class בשם DirectionalExpectimaxAgent אשר בקובץ submission.py .

ההבדל בין הסוכן החדש לסוכן שמימשתם בסעיף הקודם הוא שסוכן זה מותאם לסוכן הרוח DirectionalGhost הממומשת גם היא בקובץ ghostAgents.py .

על מנת להריץ את המשחק עם רוח שאינה RandomGhost נשתמש בדגל –g ובשם הרוח.

1. הסתכלו במימוש הרוח וענו- מהי התפלגות התנועה של הרוח? מהי האסטרטגיה המלאה שלה? הדפיסו את המילון dist בסוף הפונקציה getDistribution של הרוח וודאו שאתם מבינים למה הוא נראה כפי שהוא נראה ושהוא מתאים לאסטרטגיה שתיארתם)לאחר שתוודאו זאת מחקו את שורת ההדפסה שהוספתם(.
2. בהינתן האסטרטגיה שמצאתם בסעיף 1, ממשו את האלגוריתם Expectimax במחלקה DirectionalExpectimaxAgent . שימו לב שמומלץ מאוד להבין את המימוש של הרוח DirectionalGhost לפני שתיגשו למימוש סעיף זה, תוכלו להשתמש במימוש שלכם בשיטות דומות (ואף בפונקציות זהות) לאלה שמופיעות במימוש הרוח. נזכיר שאין לשנות את הקוד של הרוח.
3. תארו את ההבדלים בין המימוש של DirectionalExpectimaxAgent וזה של RandomExpectimaxAgent .
4. הציעו רעיונות לשיפור אסטרטגיית הרוחות כך שהרוחות יעשו את עבודתן טוב יותר משתי הרוחות שראינו בסעיפים הקודמים. כתבו לפחות 2 רעיונות לשיפור שחשבתם עליהם והסבירו מדוע לדעתם הם ישפרו את הרוחות. אין צורך לממש