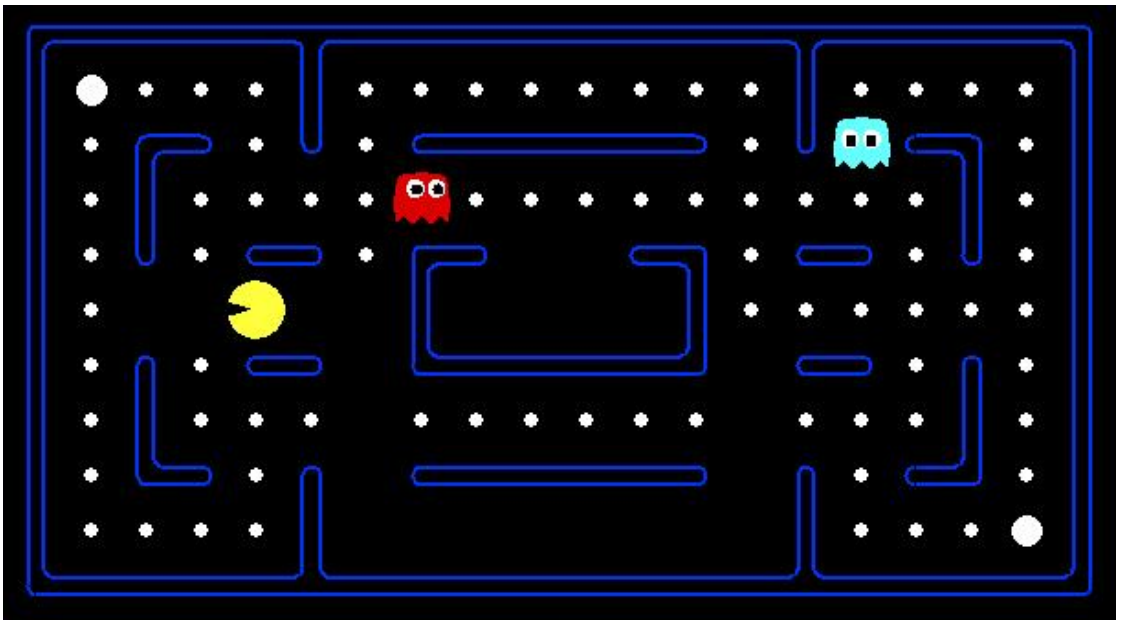
תרגיל בית 2- פתרון בעיות על ידי למידה מחיזוקים

מבוא

מטרת התרגיל היא להתנסות בפתרון גירסה של משחק pacman בעזרת למידה מחיזוקים, למודל ספציפי.



עקרון המשחק הוא ש-pacman (הדמות הצהובה) צריכה לנסות "לאכול" את כל האסימונים (הנקודות הלבנות בתמונה למעלה), מבלי להיאכל על ידי הרוחות רפאים שיש על הלוח. באופן יותר קונקרטי, מטרת המשחק היא להוביל את Pacman על גבי לוח המשחק ולמקסם את כמות הנקודות שנצברו לאחר מספר צעדים נתון.

לוח המשחק מורכב מ – NxM (כאשר N מספר השורות ו-M מספר העמודות) משבצות ומכיל 4 סוגים של אוביקטים:

- רוחות רפאים
- אסימונים
- קיר
- שחקן (ה-pacman)

תרגיל בית 2 אינו תלוי בתרגיל בית 1.

הבעיה

כקלט תקבלו את המצב ההתחלתי של המשחק. זוהי מטריצה (game) בגודל NxM שמייצגת את לוח המשחק, הכניסות במטריצה מסמנות את המצב של האוביקטים בו. השחקן יכול לזוז ימינה, שמאלה, למעלה ולמטה, בתנאי שבמשבצת אליה הוא זז אין קיר.

המערכת אותה תממשו תתקשר בצורה איטרטיבית עם הקוד אותו אנחנו מספקים. הקוד שלנו בעצם מסמלץ את הסביבה של Pacman, כלומר את לוח המשחק. לאחר אתחול של אובייקט המחלקה שתבנו, הקוד שלכם יקבל את המצב הנוכחי של הלוח ויחזיר את הפעולה שעל Pacman לבצע.

המשחק רץ מספר מוגדר של צעדים ומחולק לתתי-משחקים כאשר כל תת-משחק מתחיל במצב ההתחלתי של הלוח, ויכול להסתיים בשני אופנים:

- הצלחה – Pacman "אכל" את כל האסימונים – מזכה בעשר נקודות.
- כישלון – אחת המפלצות "אוכלת" את Pacman (יכול להיגרם על ידי כניסה של Pacman למשבצת בה יש מפלצת) – גורע עשר נקודות מהניקוד הכולל.

אם תת-משחק מסתיים, הלוח חוזר למצב ההתחלתי והכל חוזר חלילה עד ש- Pacman יסיים את צעדיו.

ייצוג קלט

Game - מטריצה NxM (כאשר כאמור N מספר השורות ו-M מספר העמודות) שמייצגת את לוח המשחק.
הערכים האפשריים במטריצה והמשמעויות שלהם מפורטים להלן:

- 99 – קיר
- 70 – שחקן (פאקמן)
- 21 – רוח רפאים **אדומה** במשבצת עם **אסימון**
- 20 – רוח רפאים **אדומה** במשבצת ללא **אסימון**
- 31 - רוח רפאים **כחולה** במשבצת עם **אסימון**
- 30 - רוח רפאים **כחולה** במשבצת ללא **אסימון**
- 41 - רוח רפאים **צהובה** במשבצת עם **אסימון**
- 40 - רוח רפאים **צהובה** במשבצת ללא **אסימון**
- 51 - רוח רפאים **ירוקה** במשבצת עם **אסימון**
- 50 - רוח רפאים **ירוקה** במשבצת ללא **אסימון**
- 11 – משבצת רגילה עם **אסימון**
- 10 – משבצת ריקה

הקלט שמתאר את הבעיה מועבר למחלקה Controller לאחר שינויים קלים, כך שהוא מקבל שני מילונים:

- locations – מילון המכיל ייצוג של המיקום הנוכחי של Pacman והמפלצות תחת המפתחות הבאים:
  - 7 – Pacman.
  - 5 – מפלצת **ירוקה**.
  - 4 – מפלצת **צהובה**.
  - 3 – מפלצת **כחולה**.
  - 2 – מפלצת **אדומה**.
- pellets – קבוצה של האסימונים הנמצאים על הלוח.

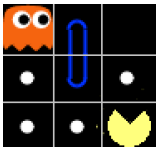
לדוגמא לוח המשחק הבא:

((21,99,10),

(11,99,11),

(11,11,70))

מתאר את המשחק הבא:



כלומר גריד בגודל 3 על 3, ונמצאים בו 2 קירות, 4 משבצות אסימונים, משבצת ריקה, רוח רפאים אחת, והשחקן (ה-pacman).

הקלט שתקבלו ייוצג באופן הבא:

{(locations = {2: (1, 1), 3: None, 4: None, 5: None, 7: (3, 3

([(pellets = set([(1,1),(2, 1), (3, 1), (3, 2), (2, 3

אופן וחוקיות התזוזה

לרשותכם 4 פעולות של השחקן:

- תזוזה ימינה – מיוצגת ע"י מחרוזת "R"
- תזוזה למטה – מיוצגת ע"י מחרוזת "D"
- תזוזה שמאלה – מיוצגת ע"י מחרוזת "L"
- תזוזה למעלה – מיוצגת ע"י מחרוזת "U"

כל פעולה כזאת גורמת לפקמן לזוז משבצת אחת בכיוון הרצוי, חוץ מתזוזה לכיוון הקיר.

הפעולות של Pacman אינן דטרמיניסטיות. ישנה הסתברות מסוימת, p, ש- Pacman יבצע את הפעולה שהוא בחר לבצע, ובהסתברות 1-p הוא יבצע את אחת הפעולות שלא נבחרו, כאשר ההתפלגות ביניהן היא התפלגות אחידה. ההסתברות p אינה ידועה לסוכן!

לדוגמא: אם Pacman בוחר לזוז למעלה, אזי בהסתברות של p הוא אכן יזוז למעלה, ובהסתברות 1-p יזוז באופן רנדומלי לאחד מ – 3 הכיוונים הנותרים (ימינה, למטה ושמאלה).

במסגרת התרגיל אתם יכולים לנסות להזיז את Pacman לתוך קיר: זה לא יזיז את השחקן. בהינתן ו-Pacman נכנס למשבצת בה קיימים מפלצת ואסימון בו-זמנית אם Pacman "מת" הוא אינו "אוכל" את האסימון, כלומר ה"פרס" עבור מהלך זה הוא עדיין מינוס 10. אם Pacman "אוכל" את האסימון אז הוא גם "אוכל" את המפלצת (כלומר היא נעלמת מהלוח), ואז ה"פרס" עבור מהלך זה הוא 1.

<u>חוקים לתזוזת המפלצות:</u>

1. **המפלצות במשחק אינן זזות.**
2. לכל מפלצת מצבע מסוים יש הסתברות q "לאכול" את Pacman, והסתברות 1-q ש- Pacman "יאכל" את המפלצת.
3. ההסתברות q תלויה בצבע של המפלצת.
4. הסתברויות q האלה לא ידועות לסוכן!

<u>אסימונים</u>

האסימונים במשחק "נאכלים" דטרמיניסטית.

<u>ניקוד</u>

הניקוד המצטבר במשחק יחולק באופן הבא:

- "אכילה" של אסימון – נקודה אחת.
- סיום המשחק בהצלחה – עשר נקודות.
- כישלון במשחק – מפחית עשר נקודות.

<u>ארכיטקטורה כללית של המערכת:</u>

בכל מהלך, ה- Controller שבניתם בוחר פעולה לביצוע (כיוון התזוזה של Pacman), מעביר אותה לקוד הסביבה (שאנו מספקים). לאחר שהסביבה מעדכנת את לוח המשחק בהתאם למהלך שקיבלה היא מחזירה את המצב המעודכן של הלוח ל-Controller. **לתשומת לבכם, אם Pacman מת או כל האסימונים "נאכלו", המערכת תחזיר את המצב ההתחלתי של הלוח**. כזכור, מצב מיוצג ע"י מטריצה (טאפל של טאפלים) עם הקידוד שהוגדר לעיל, ופעולה תיוצג ע"י מחרוזת בעלת תו אחד מבין התווים: 'L', 'R', 'D', 'U'.

תיאור המשימה: מימוש

עליכם לממש מחלקה בשם Controller בתוך קובץ בשם ex2.py מחלקה זו צריכה להכיל את שתי הפונקציות הבאות (לפחות):

- __init__(self, N, M, init_locations, init_pellets, steps)

פונקצית אתחול ה-Controller. מקבלת כפרמטרים:

1. NxM – גודל של המטריצה.
2. init_locations – מילון של מיקומים התחלתיים בהן נמצאים Pacman והמפלצות.
3. init_pellets – קבוצה של מיקומים התחלתיים של האסימונים הנמצאים על הלוח.
4. steps – מספר הצעדים שהמערכת תרוץ על המשחק. פונקציה זו צריכה לחשב מדיניות (policy) לבחירת התזוזה של Pacman בהתחשב במצב הנוכחי של הלוח. חישוב המדיניות צריך להתבצע לפי אלגוריתם Q-Learning שלמדתם בקורס.

שימו לב: פונקציה זו תורץ עם הגבלת זמן של 5 שניות, והיא **חייבת** לסיים עד אז.

קבלת השעה הנוכחית בפייתון (מיוצגת כמספר שניות מאז 1.1.1970) מתבצעת ע"י קריאה ל time.time().
מומלץ בחום לקרוא לפונקציה זו בתחילת הקריאה ל__init__, ובמקומות אסטרטגיים בזמן הריצה, לקרוא לה שוב, ולבדוק כמה זמן עבר.

- choose_next_move(self, locations, pellets)

פונקציה זו משמשת לעדכון המצב הנוכחי ובחירת פעולה. תכנית הבדיקה תקרא לפונקציה זו, כאשר:

1. locations – מילון של מקומות בהן נמצאים Pacman והמפלצות:
2. pellets – קבוצה של האסימונים הנמצאים על הלוח.

פונקציה זו צריכה להחזיר את הפעולה לביצוע במצב הנתון. הפעולה מיוצגת ע"י מחרוזת בעלת תו אחד ('U', 'D', 'R', 'L').

בנוסף, הקובץ ex2.py צריך להכיל משתנה בשם id, שמכיל את מספר הזהות שלכם.

קבצים נוספים

הקובץ *pacman.py* מכיל את המחלקה **Game**.

מחלקה **Game** מייצגת משחק בלוח הנתון, ומכילה את בשדות הבאים:

- *steps* – מספר הצעדים.
- *init* – המצב ההתחלתי של הלוח.
- *init_locations* – המיקום התחלתי של המפלצות ו-Pacman.
- *init_pellets*– המיקום התחלתי של האסימונים.
- *board* – המצב הנוכחי של הלוח.
- *locations* – המיקום הנוכחי של המפלצות ו-Pacman.
- *pellets* – אילו מהאסימונים עדין על הלוח.
- *done* – האם תת-המשחק הנוכחי הסתיים.

המתודה:

- *__init__(self, steps, board)*

מאתחלת את המחלקה.

- *set_locations(self)*

מאתחלת את השדות *init_locations, init_pellets* ו-*gates*.

- *reset(self)*

מאתחלת את השדות *pellets* ו-*locations*.

- *(there_is_cell(self, move*

מחזירה *True* אם ורק אם המהלך אינו חורג מהלוח.

- *(move_packman(self, move*

מזיזה את Pacman *(אם אפשר)* ומחזירה את התגמול על כך.

- *(update_board(self, move*

מבצעת את המהלך הנתון. מזיזה את Pacman ומחזירה את התגמול.

- *(play_game(self, policy, p, visualize = True*

מריצה את המשחק מספר צעדים נתון.

- *(evaluate_policy(self, policy, p, times, visualize = True*

מבצעת הערכה של ה-*Controller* שבניתם ע"י הרצה של *play_game 30* פעם, ומחזירה את הממוצע החשבוני.

בדיקת התרגיל

התרגיל המוגש ייבדק באופן אוטומטי, ולכן חשוב להקפיד על שמות מדויקים של קבצים, מחלקות ופעולות.
הרצת הקובץ באמצעות הפקודה:

```
python3 check.py
```

אנחנו נשתמש בקוד שלכם, ונריץ את הסוכן הלומד על מספר בעיות שונות. על כל משחק נריץ את הבקר שבניתם steps צעדים, ונחשב את הפרס המצטבר. (כל זה מתבצעת בקובץ check.py). אם ברצונכם לראות את כל המצבים בהם השחקן עובר, עליכם להחליף את הערך של visualize ל-True בקריאה ל-evaluate_controller(...).

שימו לב: הפונקציה evaluate_ controller(...) תורץ עם הגבלת זמן של 30 שניות, והיא חייבת להסתיים בזמן הנתון.

<u>ציון יינתן על פי נכונות הקוד ועמידה בדרישות בלבד, לא על פי ביצועים. בדיקת הביצועים מתבצעת לצורכי וידוא תקינות (והתכנסות הפתרון) בלבד.</u>

הסבר קצר על הקוד המצורף

הקוד מורכב משלושה קבצי פייתון:

1. ex2.py – קובץ שבו נעשית העבודה העיקרית שלכם, והקובץ היחיד שעליכם לשנות. אמור להכיל את כל מה שנדרתם לממש ב- "תיאור המשימה: מימוש". **התרגיל יבדק עם קובץ ex2.py שלכם, ושאר הקבצים כפי שהם מופיעים במצבם המקורי ולכן אין טעם לשנות קבצים אחרים** (למעט הבעיות השונות שנבדוק עליהם את הקוד).
2. check.py – קובץ המכיל פונקציות מעטפת המנסות לפתור את הבעיה, ומכיל בעיה קטנה לדוגמא שאפשר לפתור. זה הקובץ שעליכם להריץ לבדיקת הפתרון שלכם.
3. pacman.py – מכיל את המחלקה Game.

הגשה

- תאריך ההגשה עד ה 12.3.24, כולל (עד 23:59)
- הגשה ביחידים בלבד
- את הת.ז. של המגיש יש לרשום במשתנה id בקובץ ex2.py וכן לצרף קובץ details.txt המכיל את שם ות.ז של המגיש.
- שאלות על התרגיל יש לשאול בפורום שאלות ותשובות יעודי ב"למדה". שאלות יענו אחת ליומיים - שלושה במרוכז.
- הגשת התרגיל תהיה דרך מערכת ה submit. יש להגיש רק את הקובץ ex2.py וקובץ details.txt. אין להגיש קבצי עזר המצורפים לתרגיל. אין להגיש קובץ בפורמט אחר (לדוגמא zip או rar)