**תרגיל מספר 2 - פלג בן ברק (207233560), עומרי כהן (203246939)**

**למידה מבוססת חיזוקים**

שאלה 1:

הוספנו 3 פונקציות חדשות לאלגוריתם הקודם מעבודה 1.

def approximate\_policy\_evaluation\_montecarlo(policy, num\_of\_iterations=1000):

הפונקציה מקבלת מדיניות ומספר איטרציות רצוי לטובת הערכת המדיניות ומבצעת הערכת מדיניות באמצעות Montecarlo every visit.

אנחנו בחרנו לממש את האלגוריתם באמצעות ריצה של איטרציה שלמה בכל פעם תוך שמירה של הState שהיינו בהם ברשימה ביחד עם כמות הפעמים שראינו כל state ובסוף עברנו על כל המצבים שראינו ועדכנו את הערך שלהם ביחס לפרס האחרון שראינו באיטרציה (מכיוון שאין מחיר לפעולה וכן הפרס מגיע רק בסיום איטרציה).

def approximate\_policy\_evaluation\_montecarlo\_firstvisit(policy, num\_of\_iterations=1000):

באופן דומה מימשנו את הערכת המדיניות אך במהלך איטרציה אם כבר ראינו מצב לא הכנסנו אותו פעם נוספת לרשימת המצבים אותם ראינו באיטרציה (בבלאק ג'ק לא ניתן להגיע חזרה למצב בו היית ולכן פרקטית אין הבדל בתוצאה הצפויה).

def approximate\_policy\_evaluation\_sarsa(policy, num\_of\_iterations=1000):

הפונקציה מקבל מדיניות ומספר איטרציות רצוי לטובת הערכת המדיניות ומבצעת הערכת מדיניות בטכניקת sarsa.

בכל צעד במהלך איטרציה ביצענו עדכון לstate הנוכחי באמצעות שינוי ערך ה-value ביחס של 0.1\*(v(next\_state)-v(state) – אופן העדכון הוא כך מכיוון שכלל הפרסים במהלך ריצה הם 0 למעט הפרס האחרון וכן אנחנו שואפים לעדכן את מטריצת ה V ולא ה Q.

את הקוד של עבודה 1 ביססנו על עבודה 1 של פלג עם שינויים קלים.  
להלן הסבר על הקוד של עבודה 1 (נלקח גם כן מההסבר על עבודה 1):

The code consists of functions applying the policy iteration algorithm for finding the optimal strategy.

We begin by applying a reward of -1, 0, or 1 for the result of an ‘episode’ (one blackjack game), which is given to the agent upon completing a game.

Our first step is to calculate the transition probability matrix in the function:

def transition\_matrix(env, num\_of\_samples=num\_samples):

which returns a dictionary with the probability of passing between each 2 states based on the action taken. The dictionary keys are state, action, next\_state. We chose to represent using a dictionary and not a matrix for code clarity and readability. The code converges rather quickly to optimal values and therefore the runtime isn’t heavily affected by the possible added time complexity.  
  
We then continue to answer 2B to find the best policy using

def policy\_iteration():

which activates the algorithm taught in class – we iterate using greedy policy improvement, where for each policy we calculate the expected value using the formula.  
  
Our code includes a function

def apply\_policy(policy):

which received a policy dictionary and applies it to a single episode.

* ההבדל המהותי בקוד הוא שעכשיו הפונקציה policy\_iteration מקבלת כארגומנט את שיטת ההערכה (לפי שלושת האופציות – SARSA, first or everyvisit MC)), וקוראת לשיטת הערכת המדיניות לפי הארגומנט.  
    
  לבסוף אנחנו מציגים בגרף את הערכים של 3 המדיניות.  
  שימו לב שכדי לקבל התכנסות של ערך המדיניות עבור הדרישה *"כדי לחשב את הערך של מדיניות, השתמשו בערך הממוצע של כל הידיים שלשחקן יש קלפים בסכום 13-16 ואין לו אס, ולדילר יש 7 או 8."* יש להריץ מספר גדול מאוד של הרצות (בערך 10,000 כפול 20 עבור כל אחת מהשיטות, מה שלוקח כמה דקות. אם מחשבים ערך מדיניות כאשר סוכמים את כל המצבים ניתן לקבל התכנסות מהירה תוך כ1,000 כפול 20 הרצות עבור כל אחת מהשיטות.  
    
  להלן הגרף לאחר הרצה   
  A graph of a line graph

  Description automatically generated with medium confidence