

# תרגיל בית 2 : חיפוש רב סוכני

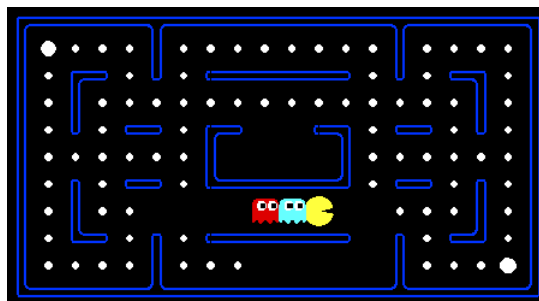
## פקמן

### מטרות התרגיל

- התנסות באלגוריתמים לחיפוש רב סוכני.
- מימוש אלגוריתמים *Minimax, Expectimax* ועוד שיפורים והרחבות.
- התנסות בפיתוח היוריסטיקה למשחק פקמן.
- ביצוע מחקר המשווה ביצועים של שחקנים ואלגוריתמים שונים.


### הערות

- תאריך הגשה : 30.12.18, 23:59.
- את המטלה יש להגיש **בזוגות בלבד!**
- שאלות בנוגע לתרגיל יש לשלוח **אך ורק למייל הקורסי** : [ai.technion@gmail.com](mailto:ai.technion@gmail.com)
- המתרגלים האחראים על התרגיל הם **גיא ק. מיכל ופאר**.
- שאלות הנוגעות לבקשות לדחייה- יש לחפש תחילה תשובה ב**סילבוס** הקורס. במידה ולא קיבלתם מענה שם יש לשלוח מייל **למיכל**.
- קראו היטב את ההסברים וההוראות במסמך זה, מטרתם לסייע לכם בהבנת הדרישות של התרגיל. שימו לב להוראות ההגשה המצורפות בסוף התרגיל.
- התעדכנו ברשימת ה-FAQ באתר הקורס בתדירות גבוהה, לפני פנייה בשאלות דרך המייל ולפני הגשת התרגיל. ההערות שתתפרסמה באתר הקורס מחייבות את כלל הסטודנטים בקורס!
- עקבו בתשומת לב רבה אחר הוראות ההגשה המצורפות במהלך התרגיל ובסופו לפני הגשתו.
- בתרגיל זה, כמו גם בתרגילים הבאים בקורס, הרצת הניסויים עשויה לקחת זמן ולכן מומלץ מאוד להימנע מדחיית העבודה על התרגיל לרגע האחרון. **לא תינתנה דחיות על רקע זה.**



## מבוא והנחיות

1. במטלה זו תתכננו ותממשו שחקנים למשחק פקמן.
2. המטלה מכילה חלק יבש וחלק רטוב.
3. את חוקי המשחק ניתן למצוא בוויקיפדיה.  
<https://he.wikipedia.org/wiki/%D7%A4%D7%A7-%D7%9E%D7%9F%D7%A4%D7%A8%D7%98%D7%99-%D7%94%D7%9E%D7%A9%D7%97%D7%A7>
4. העתקות יטופלו בחומרה בדין משמעתי.
5. מומלץ לחזור על שקפי ההרצאות והתרגולים בנושא "חיפוש רב סוכני" לפני תחילת העבודה על התרגיל.
6. 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).  
העתקות גם מהאינטרנט, יטופלו בחומרה.
7. 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
8. הקוד שאתם מקבלים מכיל את הקבצים הבאים:
  - a. submission.py - **הקובץ היחיד שעליכם לשנות ולכתוב בו קוד**, פה תממשו את השחקנים והאלגוריתמים שלכם.
  - b. pacman.py - הקובץ המרכזי המריץ את המשחק. מכיל את הטיפוס GameState בו תעשו שימוש רב.
  - c. game.py - הלוגיקה מאחורי חוקי המשחק ולוח המשחק
  - d. util.py - מבני נתונים בהם תוכלו להשתמש במימוש אלגוריתמי החיפוש
  - e. layout.py - קוד המטפל בפריסות הלוח השונות (הנמצאות בתיקייה layouts)
  - f. ghostAgents.py - מימוש סוכני הרוחות
  - g. קבצי תמיכה שאין צורך להתעמק בהם:

graphicsDisplay.py, graphicsUtils.py, textDisplay.py, keyboardAgents.py
9. **שימו לב שעליכם לכתוב קוד ולשנות אך ורק את הקובץ submission.py**
10. בסיום העבודה תגישו גם חלק יבש הכולל מענה לשאלות ודיווח תוצאות, עליכם לכתוב תשובה בדו"ח היבש עבור כל סעיף שלצדו תראו את הסימן 

## 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-idea שאתם עובדים איתו (אנו, כאמור בתרגיל הראשון, ממליצים על pycharm). אנו נריץ את המשחק דרך 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 אלא 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 -l testClassic -p ReflexAgent`

- 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` (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 שתרכזו 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 Min-Max המתאים לשני שחקנים (בו רק שכבת מינימום אחת עבור יריב יחיד אחרי כל שכבת מקסימום) אלא 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 שתרכזו 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.depth` ו `self.evalFn`.

`self.evalFn` : שדה זה יחזיק את הפונקציה ההיוריסטית שבה הסוכן ישתמש. הוא מוגדר כברירת 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 `sucessesor` 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.

`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

- 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.
- ממשו את אלגוריתם אלפא בטא בclass המתאים (`AlphaBetaAgent`) בקובץ `submission.py`.
- האם הסוכן `AlphaBetaAgent` שמימשתם בסעיף זה יתנהג שונה מסוכן `MinMaxAgent` שממשתם בסעיף הקודם :
  - מבחינת זמן ריצה? הסבירו מדוע.
  - מבחינת בחירת מהלכים? הסבירו מדוע.



### חלק ה' – בניית סוכן Expectimax לרוח רנדומלית

נרצה לבנות סוכן המשתמש באלגוריתם `Expectimax` בclass בשם `RandomExpectimaxAgent` אשר בקובץ `submission.py`.

שימו לב שבברירת 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 המותרים 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.



### חלק ז' – ניסוח השערות במשחק פקמן

כפי שהוסבר בתרגול על ניסויים סטטיסטיים, בשביל לערוך מבחן סטטיסטי, עלינו לנסח את השערת האפס- $H_0$ , ההשערה החלופית,  $H_1$ , וכן לנסח מבחן סטטיסטי שבאמצעותו נדע האם עלינו לקבל את  $H_0$ , או לדחות אותה.



בסעיף זה אתם נדרשים לנסח השערת אפס והשערה חלופית, וכן לנסח מבחן שתוכלו לעשות. אין צורך לבצע את המבחן עצמו או להגיע לתוצאות חישוב, עליכם לפרט ברמת התיאוריה בלבד. תוכלו להשתמש במה שנלמד בתרגול, או במבחנים חדשים שתמצאו באינטרנט (מומלץ להסתכל על מבחן ה- t-test). המבחן שלכם יכול להתייחס לתוצאות המשחקים ולערוך השוואה בין השחקנים עצמם.

## חלק ח' – ניסויים, תוצאות ומסקנות.

נרצה להשוות בין השחקנים השונים שראינו עד כה, תחת מגבלות עומק שונות ולוחות משחק שונים. בפועל, ההשוואה תיעשה בין 5 שחקנים:

(1) *ReflexAgent* המסופק

(2) *ReflexAgent* עם היוריסטיקה משופרת (נקרא לו *BetterAgent* בדיווח התוצאות)

(3) *MinMaxAgent*

(4) *AlphaBetaAgent*

(5) *RandomExpectimaxAgent*

\* שימו לב שכאשר אתם מריצים את (1) ו(2) אתם צריכים להחליף בקוד בין ההיוריסטיקות בהן השחקן משתמש.

\*\* בריצות המתוארות כעת נשתמש ברוח הרנדומלית בלבד.

נסמן  $D = \{2, 3, 4\}$ . עבור כל שחקן, מגבלת עומק  $d \in D$ , ובחירת לוח הריצו סדרת של 7 משחקים. עבור כל 7 משחקים כאלה, התוצאה הסופית היא ממוצע ה score של שבעת המשחקים שהרצתם. שימו לב שעבור השחקנים 1 ו 2 אין משמעות לפרמטר העומק כי השחקנים אינם מסתכלים לעומק הגדול מ 1, ולכן עבורם הריצו רק עבור  $d = 1$ . אנחנו נתעניין במצב שבו ישנן 2 רוחות בלבד (עבור לוחות המאפשרים זאת, עבור כאלה המאפשרים רק רוח אחת נריץ עם רוח אחת). נבצע את הניסוי עבור כל הלוחות שנמצאים בתקיה *layouts*. לסיכום, כל תת ניסוי מוגדר ע"י:

בחירת שחקן, בחירת לוח, בחירת מגבלת עומק. כל תת ניסוי כזה נריץ 7 פעמים ונחשב ממוצע על ה score. סה"כ עליהם להריץ  $3_{d \in D} \cdot 10_{boards} \cdot 7_{games} + 3_{players} \cdot 10_{boards} \cdot 7_{games}$  משחקים. בנוסף לנתונים אלה, נרצה להוסיף את הזמן הממוצע שלוקח לשחקן לשחק תור, לשם כך שמרו את ההפרש בין הזמן בתחילת הפונקציה *getAction* ובסופה בכל תור, ודווחו את הממוצע בסוף המשחק.

1. יש להגיש קובץ בשם *experiments.csv* שיכיל את תוצאות המשחקים שהרצתם. כל שורה בקובץ תייצג 7 משחקים שהרצתם ותכיל 5 עמודות: שם שחקן, מגבלת העומק, שם הלוח, ניקוד השחקן והזמן הממוצע שלקח תור, לדוגמה:

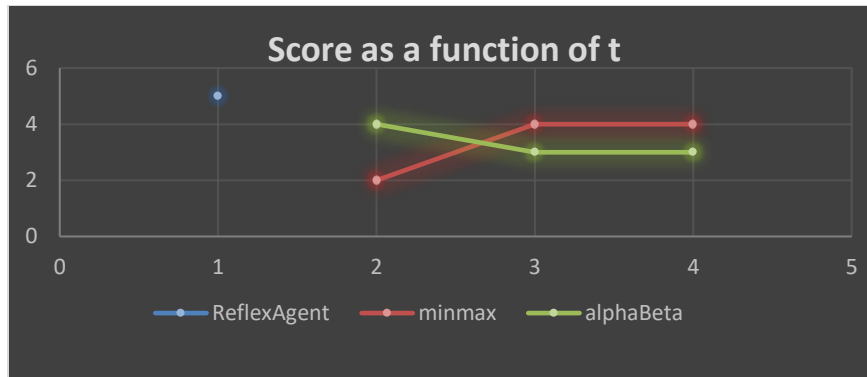
*BetterAgent, 1, testClassic, avgScore, avgAvgTrunTime*

תודפס שורה אחת עבור כל שבעת המשחקים הזוהים שהרצתם שתכיל את שלושת הנתונים הקבועים עבור שבעת המשחקים (שם השחקן, העומק והלוח), את ממוצע ה score עבור שבעת המשחקים, ואת ממוצע הזמן הממוצע שלקח כל תור בשבעת המשחקים. אין לכלול כותרות בקובץ, אלא רק את תוצאות הניסויים.

2. הצגת התוצאות:

I. הציגו גרף המתאר עבור כל אחת ממגבלות העומקים את הניקוד הממוצע הכולל של כל שחקן, כלומר סכום הממוצעים על כל הלוחות בעומק זה. הקפידו להשתמש בצבעים או בסוגי קו שונים עבור שחקנים שונים. לנוחיותכם מצורף גרף לדוגמה (\*).





\* שימו לב כי הגרף חלקי. בגרף שלכם דווחו על כל חמשת הסוכנים.

\*\* שימו לב כי עבור ReflexAgent רק ערך D אחד רלוונטי.

II. צרפו את טבלת הנתונים שיצרו את הגרף. למשל (\*):

$d = 1$	$d = 2$	$d = 3$	$d = 4$	
1				<i>ReflexAgent</i>
4				<i>BetterAgent</i>
	3	3	4	<i>MinMaxAgent</i>
	4	2.5	1	<i>AlphaBetaAgent</i>
	4	5	2	<i>RandomExpectimaxAgent</i>

(\*) שימו לב כי הגרפים והטבלאות שהצגנו כאן נוצרו לצורך המחשה בלבד.

3. מהן המסקנות שניתן להסיק מהגרף והטבלה מסעיף 2? האם התוצאות שקיבלתם תואמות את ציפיותיכם?



4. צרפו את אותו גרף ואותה טבלה כמו בסעיף 2, רק שהפעם נסתכל על הזמן הממוצע שלקח כל תור כפונקציה של  $d$ . כלומר, עבור כל שחקן וכל אחת ממגבלות העומקים הציגו את הזמן הממוצע שלקח תור (ממוצע על כל הלוחות).



5. מהן המסקנות שניתן להסיק מהגרף והטבלה מסעיף 4? האם התוצאות שקיבלתם תואמות את ציפיותיכם?



6. נרצה להשוואת בין *RandomExpectimaxAgent* ו-*DirectionalExpectimaxAgent*. לצורך כך הריצו חמישה משחקים של כל אחד מהסוכנים, עם עומק 4, על הלוח trickyClassic. עשו זאת פעם אחת עם הרוח הרנדומלית ופעם אחת עם הרוח הכיוונית (כך שכל סוכן ירוץ 5 משחקים עם הרוח לה הוא מיועד 50 משחקים עם הרוח השנייה. בסך הכל 20 משחקים). דווחו את התוצאות שקיבלתם (בטבלה בדוח היבש, אין צורך להכליל אותן בקובץ csv). האם התוצאות תואמות את ציפיותיכם?





7. בצעו השוואה עבור הלוח: *minimaxClassic* בהגבלת עומק  $d = 4$ . אתם יכולים להתייחס לתוצאות הקודמות או לבצע מספר הרצות חדשות ולתאר את התוצאות שקיבלתם. כיצד שיחק כל אחד מהשחקנים (שימו דגש על שחקן minimax ושחקן RandomExpectimax). מה מאפיין את הלוח?
8. באופן דומה לסעיף הקודם, הפעם בצעו השוואה עבור הלוח: *trappedClassic* בהגבלת זמן  $d = 4$ . כיצד שיחק כל אחד מהשחקנים (הדגש על שני השחקנים שהוזכרו בסעיף קודם)? מה מאפיין את הלוח?
9. השוו את ביצועי השחקנים ביחס למגבלת העומק, לזמן הממוצע לתור, הלוחות, ושאר הפרמטרים שראיתם בחלק הניסויים. תוכלו לבצע ניתוח של מגמות השיפור והדעיכה, לבחון את הניקוד הכולל של כל אחד מהשחקנים. שאלות שתוכלו לבחון: באילו מקרים ההיוריסטיקה שלכם מתגברת על ההיוריסטיקה הפשוטה אם בכלל? מי השחקן הטוב ביותר בכל לוח? איך הלוחות משפיעים על השחקנים השונים? האם מגבלות העומק השפיעו על התוצאות? מה היה קורה אילו היו מגבלות זמן במקום מגבלות עומק? הסבירו בהרחבה את כל המאפיינים שראיתם בתרגיל ובצעו ניתוח מעמיק בין זוגות השחקנים. שימו לב, זהו סעיף סיכום התרגיל ומשקלו בציון משמעותי.

## חלק ט' – תחרות בקורס

לסיים, עליכם להגיש שחקן **פקמן** לתחרות הקורסית.

- שימו לב, סעיף זה ייתן נקודות בונוס לציון **הסופי של הקורס** באופן הבא: הזוג שיזכה במקום הראשון בתחרות יקבל 5 נקודות בונוס לציון הסופי, הזוג שיזכה במקום השני בתחרות יקבל 3 נקודות בונוס לציון הסופי והזוג שיזכה במקום השלישי בתחרות יקבל 2 נקודות בונוס לציון הסופי.
- ההשתתפות בתחרות היא **חובה**. לא יורדו נקודות על שחקן שביצעו לא טובים אך יורדו נקודות אם השחקן שתגישו לא ירוץ/ לא תגישו את ההסבר המילולי הנדרש.
- הגישו הסבר קצר בדו"ח שאתם מגישים המתאר את שחקן התחרות שלכם. כתבו איך הוא עובד ומדוע בחרתם בשיטת פעולה זו.
- התחרות תערך כך:

אנו נריץ את הסוכן שלכם על 3 לוחות שונים (מתוך הלוחות המסופקים בתיקיית layouts) עם 2 סוגי הרוחות, כך שבסך הכל הסוכן שלכם ישחק 6 משחקים. התוצאה שכל צוות יקבל תהיה הממוצע של הרצות אלו. אתם נדרשים להגביל את עומק הפיתוח של כל שלב במשחק **לעומק של 4 בלבד!** אנו נבדוק את כמות הפיתוחים ולכן אין לרמות בעומק הפיתוח! כמו כן, תהיה מגבלת זמן של 30 שניות על משך המשחק כולו (כאשר המשחק מורץ

עם הדגל q-, כלומר ללא גרפיקה), ולכן זמן החישוב מוגבל. הקפידו לבדוק את ההרצה עם פרמטר זה לפני ההגשה.

אתם יכולים להשתמש בכל אלגוריתם בו השתמשתם במהלך התרגיל ו/או לפתח אלגוריתמים חדשים כרצונכם, כל עוד הם לא ניגשים לאינטרנט וירוצו בסביבה מבודדת עליה אנחנו נבדוק.

## הוראות הגשה

- הגשת התרגיל תתבצע **אלקטרונית בלבד** דרך אתר הקורס.
- אתם מתבקשים ליצור קובץ zip בשם  $AI2\_id1\_id2$  (ללא הסוגריים המשולשים), שבתוכו הקבצים הבאים:
  - קובץ הקוד submission.py שימו לב, זה קובץ הקוד היחיד שאתם מגישים ולכן מומלץ לוודא שהוא רץ עם הקבצים המקוריים שקיבלתם מאתנו, כלומר ללא שינויים שאולי עשיתם בטעות בקבצים האחרים.
  - קובץ בשם *AI\_HW2.pdf*, המכיל את התשובות לחלק היבש
  - קובץ בשם *experiments.csv* כפי שמתואר בסעיף ח חלק 1.
- **בעיות הרצה יגורו הורדה בציון התרגיל.**
- אין להעתיק את הקבצים המסופקים לכם אל תוך תיקיית ההגשה. הניחו כי קבצים אלו יהיו זמינים בעת בדיקת התרגיל.
- שימו לב שכל הפנייה למיקום קובץ/תיקייה כלשהם בקוד תהיה רלטיבית (*relative path*) ולא אבסולוטית, כך שהקוד יעבוד כפי שהוא על כל מחשב בכל מיקום שנבחר לתיקיית הפרויקט. הקפידו לבדוק זאת לפני ההגשה!
- הקפידו על קוד **ברור, קריא ומתועד!** עליכם לתעד כל חלק שאינו טריוויאלי בקוד שלכם.
- "המצאת" נתונים לצורך בניית הגרפים **אסורה** ותוביל לדיון בבית הדין המשמעתי של הטכניון.

**בהצלחה!**

