ה א ו נ י ב ר ס י ט ה  ה פ ת ו ח ה

20551

# מבוא לבינה מלאכותית

חוברת הקורס אביב 2023ב

כתב: עירן בהט-פטל

מרס 2023 – סמסטר אביב – תשפ"ג

**פנימי – לא להפצה.**

# תוכן העניינים

## אל הסטודנטים,

אנו מקדמים את פניכם בברכה עם הצטרפותכם אל הלומדים בקורס "מבוא לבינה מלאכותית".

בחוברת זו תמצאו את לוח הזמנים של הקורס, תנאים לקבלת נקודות זכות וחלק מהמטלות.
קראו אותה היטב כדי לחסוך בעיות בהמשך הדרך.

לקורס קיים אתר אינטרנט בו תמצאו חומרי למידה נוספים, אותם מפרסם צוות ההוראה.
בנוסף, האתר מהווה עבורכם ערוץ תקשורת עם צוות ההוראה ועם סטודנטים אחרים בקורס.
מידע על שירותי ספרייה ומקורות מידע שהאוניברסיטה מעמידה לרשותכם, תמצאו באתר הספרייה באינטרנט www.openu.ac.il/Library.

צוות הקורס ישמח לעמוד לרשותכם בכל שאלה שתתעורר.
ניתן לפנות למנחים בשעות ההנחיה הטלפוניות שלהם או אלי בכל יום ד' בשעות 21:30-22:30
בטלפון 054-7211962. כמו כן ניתן לפנות אלי ב- e-mail כתובתי : eranba@openu.ac.il

**לתשומת לב הסטודנטים הלומדים בחו"ל :**

למרות הריחוק הפיסי הגדול, נשתדל לשמור אתכם על קשרים הדוקים ולעמוד לרשותכם ככל האפשר.
הפרטים החיוניים על הקורס נכללים בחוברת הקורס וכן באתר הקורס.
מומלץ מאד להשתמש באתר הקורס ובכל אמצעי העזר שבו וכמובן לפנות אלינו במידת הצורך.

אנו מאחלים לכם לימוד פורה ומהנה.

ב ב ר כ ה,

עירן בהט-פטל
מרכז ההוראה בקורס

## 1.לוח זמנים ופעילויות (מס' קורס 20551 /ב2023)

| שבוע הלימוד | תאריכי שבוע הלימוד | יחידת הלימוד המומלצת | מפגשי ההנחיה* | תאריך אחרון למשלוח הממ"ן (למנחה) |
|---|---|---|---|---|
| 1 | 10.03.2023-5.03.2023 | פרקים 1,2 | | |
| 2 | 17.03.2023-12.03.2023 | פרק 3 | מפגש 1 | |
| 3 | 24.03.2023-19.03.2023 | פרק 4 | | ממ"ן 11 (להרצה) 24.3.23 |
| 4 | 31.03.2023-26.03.2023 | פרק 5 | מפגש 2 | |
| 5 | 07.04.2023-02.04.2023 (ד-ו פסח) | פרק 6 | | ממ"ן 12 (תיאורטי) 7.4.23 |
| 6 | 14.04.2023-09.04.2023 (א-ד פסח) | פרק 7 | מפגש 3 | |
| 7 | 21.04.2023-16.04.2023 ( ג יום הזכרון לשואה) | פרק 8 | | ממ"ן 13 (להרצה) 21.4.23 |
| 8 | 28.04.2023-23.04.2023 (ג יום הזיכרון, ד יום העצמאות | פרק 9 | מפגש 4 | ממ"ן 14 (תיאורטי) 28.4.23 |

** התאריכים המדויקים של המפגשים הקבוצתיים מופיעים ב"לוח מפגשים ומנחים".*

לוח זמנים ופעילויות - המשך

| שבוע הלימוד | תאריכי שבוע הלימוד | יחידת הלימוד המומלצת | מפגשי ההנחיה* | תאריך אחרון למשלוח הממ"ן (למנחה) |
|---|---|---|---|---|
| 9 | 05.05.2023-30.04.2023 | פרק 11 | | |
| 10 | 12.05.2023-07.05.2023 (ג ל"ג בעומר) | פרק 12 | מפגש 5 | ממ"ן 15 (להרצה) 12.5.23 |
| 11 | 19.05.2023-14.05.2023 | פרק 13 | | |
| 12 | 26.05.2023-21.05.2023 (ו שבועות) | פרק 16 | מפגש 6 | ממ"ן 16 (תיאורטי) 26.5.23 |
| 13 | 02.06.2023-28.05.2023 | פרק 19 | | |
| 14 | 09.06.2023-04.06.2023 | | מפגש 7 | ממ"ן 17 (תיאורטי) 9.6.23 |
| 15 | 16.06.2023-11.06.2023 | חזרה | | |

**תאריך אחרון למשלוח ממ"ן 18 (להרצה) : 23.6.23**

מועדי בחינות הגמר יפורסמו בנפרד

** התאריכים המדויקים של המפגשים הקבוצתיים מופיעים ב"לוח מפגשים ומנחים".*

## 2. תיאור המטלות

| |
|---|
| קרא היטב עמודים אלו לפני שתתחיל לענות על השאלות |

בקורס זה 8 מטלות, 4 מטלות תיאורטיות ו-4 מטלות להרצה.
פתרון המטלות הוא חלק בלתי נפרד מלימוד הקורס, שכן הבנה מעמיקה של חומר הלימוד דורשת תרגול רב.
יש להגיש לפחות 2 מטלות מבין המטלות התיאורטיות (12,14,16,17) (במשקל כולל של 5 נק׳ לפחות) ו-2 מטלות לפחות מבין מטלות הרצה (11,13,15,18) במשקל כולל של 10 נק׳ לפחות .
במטלות ההרצה חובה לקבל ציון עובר (60 ומעלה) כדי לעבור את הקורס בהצלחה.
אם שאלה מסוימת בממ״ן אינה ברורה לכם, אל תהססו להתקשר אל המנחה שלכם (בשעות הייעוץ הטלפוני שלהם) או להיעזר בקבוצת הדיון של הקורס.
להלן תמצאו הסבר על אופן הפתרון הנדרש וכיצד לשלוח את המטלה למנחה.

### 2.1 מבנה המטלות וצורת הגשתן

בקורס ישנן כאמור מטלות משני סוגים :

**מטלות רגילות** :

מטלה כזו מורכבת מכמה שאלות. בראש כל שאלה מצוין משקלה היחסי בקביעת ציון המטלה.
פתרון השאלות במטלה כזו אינו דורש הרצת תוכניות במחשב. הן נועדו לבדוק את הבנתכם בחומר הלימוד.
את הפתרונות למטלה כזו יש לכתוב בצורה ברורה ומסודרת.

**מטלות הרצה:**

במטלות אלה עליכם לכתוב תוכניות ולהריץ אותן במחשב.

את התוכניות במטלות 11 ,13 יש לכתוב בשפת Python.

את התוכניות במטלות 15, 18 ניתן לכתוב ב- Python או ב-Java.

<u>תיעוד :</u>
בכל תוכנית הוסיפו תיעוד בגוף התוכנית המסביר מהו תפקידו של כל משתנה, מה מבצעת כל שגרה וכל הסבר נוסף החשוב להבנת מהלך פעולתה של התוכנית. יש לתת שמות משמעותיים למשתנים ולשגרות המופיעים בתוכניות.
יש להקפיד על קריאות ובהירות תוך שימוש בהיסח (אינדטציה) מסודרת ואחידה.

<u>במטלת הרצה עליכם לשלוח למנחה :</u>

א. קבצי המקור של התוכנית (source code).

ב. קובץ readme המתאר את העבודה שלכם.

| |
|---|
| **תוכניות שתוגשנה בכתב-יד או ללא תיעוד או ללא קובץ המקור - לא תבדקנה!** |

## 2.2 חומר הלימוד הדרוש לפתרון המטלות

בטבלה שלהלן תמצאו מהו חומר הלימוד הנדרש (לפי פרקי הספר) לפתרון כל אחת מהמטלות.

**שימו לב!**
אין להשתמש לפתרון המטלות בידע הנרכש בפרקי לימוד מתקדמים יותר מהפרקים בהם עוסקת המטלה.

| מטלה | חומר הלימוד הנדרש לפתרונה |
|---|---|
| ממ"ן 11 | פרקים 1-3 |
| ממ"ן 12 | פרקים 1-6 |
| ממ"ן 13 | פרק 6 |
| ממ"ן 14 | פרקים 5-9 |
| ממ"ן 15 | פרקים 1-9 |
| ממ"ן 16 | פרקים 12-13 |
| ממ"ן 17 | פרקים 16,19 |
| ממ"ן 18 | פרק 19 |

## 2.3 ניקוד המטלות

המשקל הכולל של ממ"נים 11-18 הוא 30 נקודות. עליכם לצבור לפחות 15 נקודות.

**ללא עמידה בדרישות המטלות לא ניתן יהיה לגשת לבחינת הגמר**

**הכנת המטלות 11-18 חייבת להיעשות ע"י כל סטודנט בנפרד.**
**מטלות שלא תבוצענה באופן עצמאי – תיפסלנה!!!**

להלן פירוט הניקוד לכל מטלה:

| ממ"ן | ניקוד |
|---|---|
| 11 | 5 |
| 12 | 3 |
| 13 | 5 |
| 14 | 3 |
| 15 | 5 |
| 16 | 2 |
| 17 | 2 |
| 18 | 5 |

**לתשומת לבכם:**

מדיניות קורס זה היא לאשר הזנת ציון אפס במטלות שלא הוגשו כנדרש בקורס.

סטודנטים אשר לא הגישו את מכסת המטלות המינימאלית לעמידה בדרישות הקורס ולקבלת זכאות להיבחן, ומבקשים שמטלות חסרות יוזנו בציון אפס, יפנו למוקד הפניות והמידע בטלפון **09-7782222** או **יעדכנו בעצמם** באתר שאילתא http://www.openu.ac.il/sheilta

**קורסים ⇦ ציוני מטלות ובחינות ⇦ הזנת ציון 0 למטלות רשות שלא הוגשו.**

יש לקחת בחשבון כי מטלות אשר יוזן להן ציון אפס ישוקללו בחישוב הציון הסופי ובכך יורידו ציון זה ולא ניתן יהיה להמירן במטלות חלופיות במועד מאוחר יותר. על כן קיימת אפשרות שסטודנט אשר יעבור את הבחינה בהצלחה ייכשל בקורס (כשהממוצע המשוקלל של המטלות והבחינה יהיה נמוך מ- 60).

**כלל זה חל רק על המטלות התאורטיות, ואינו חל על המטלות המעשיות, שעבורן ציון המינימום הינו 60.**

**לתשומת לבכם!**

פתרון המטלות הוא מרכיב מרכזי בתהליך הלמידה, לכן מומלץ שתשתדלו להגיש מטלות רבות ככל האפשר.

כדי לעודדכם להגיש לבדיקה מספר רב של מטלות הנהגנו את ההקלה שלהלן:

בחישוב הציון הסופי נשקלל את כל המטלות שציוניהן גבוהים מהציון בבחינת הגמר.
ציוני מטלות אלה תורמים לשיפור הציון הסופי.
ליתר המטלות נתייחס במידת הצורך בלבד. מתוכן נבחר רק את הטובות ביותר עד להשלמת המינימום ההכרחי לעמידה בתנאי הגשת המטלות. משאר המטלות נתעלם.

**זכרו!** ציון סופי מחושב רק לסטודנטים שעברו את בחינת הגמר בציון 60 ומעלה והגישו מטלות כנדרש באותו קורס.

## 3. התנאים לקבלת נקודות זכות בקורס

א. הגשת 2 מטלות לפחות מבין המטלות התיאורטיות (12,14,16,17) תוך צבירת 5 נק' לפחות.

ב. הגשת 2 מטלות לפחות מבין המטלות להרצה (11,13,15,18), קבלת ציון 60 על כל אחת מהן וצבירת 10 נק' לפחות.

ג. ציון 60 לפחות בבחינת הגמר.

ד. ציון סופי בקורס 60 לפחות.

# מטלת מנחה (ממ"ן) 11 - להרצה

**הקורס:** **20551 – מבוא לבינה מלאכותית**

**חומר הלימוד למטלה:** שפת Python ופרקים 1-3

**מספר השאלות:** 8 **משקל המטלה:** 5

**סמסטר:** 2023ב **מועד אחרון להגשה:** 24.3.2023

(אב)

**מטלת הרצה ניתן להגיש בדרך אחת ויחידה :**

- שליחת מטלות באמצעות מערכת המטלות המקוונת באתר הבית של הקורס

**הסבר מפורט ב"נוהל הגשת מטלות מנחה"**

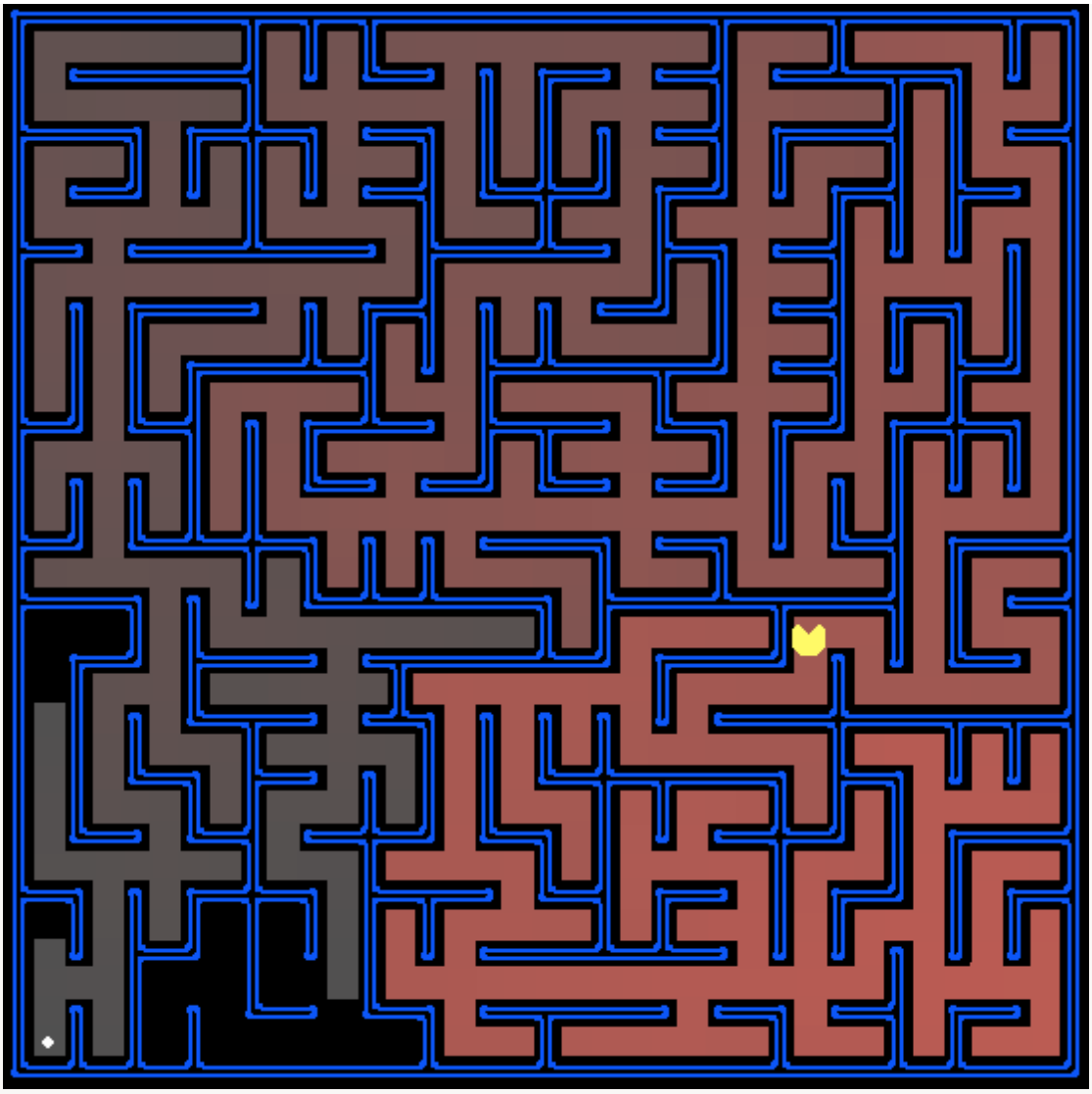
מטלות הריצה בקורס ייכתבו בשפת **Python**, שפת תכנות מונחה עצמים.
אנו לא מניחים שיש לכם נסיון בתכנות בשפה זו, אך מצפים שתלמדו את הבסיס שלה די מהר (במיוחד לאור הכרותכם ונסיונכם עם שפת Java). באינטרנט תוכלו למצוא שפע של מקורות ללימוד עצמאי של פייתון.

מטלות 11 ו-13 פותחו באוניברסיטת Berkeley על-ידי Dan Klein, John DeNero , Pieter Abbeel.

המטלה תיבדק ע"י ה-autograder (כמתואר בהמשך) והציון למטלה יינתן באופן הבא :

- הציון שה-autograder נותן יהווה 85%.
- 15% הנותרים ינתנו ע"פ התרשמות הבודק וקובץ ה-readme שעליכם לצרף, ובו תיאור שלכם את העבודה שעשיתם.
-

## פרויקט 1 : חיפוש



בפרויקט זה, סוכן הפאקמן שלכם ימצא את דרכו בתוך מבוך, הן כדי להגיע למיקום מסוים והן כדי לאסוף מזון ביעילות. המשימה שלכם היא לממש אלגוריתמי חיפוש כלליים ולהחיל אותם על תרחישי פאקמן.

כחלק מפרויקט זה עומד לרשותכם autograder, קובץ פייתון שהרצה שלו תאפשר לכם לבדוק את הפרויקט שלכם ולקבל הערכה אודות הציון שתקבלו עליו. כדי להריץ אותו הפעילו את הפקודה הבאה :

```
python autograder.py
```

הקוד לפרויקט זה מורכב מכמה קבצי פייתון, שאת חלקם תצטרכו לקרוא ולהבין על מנת להשלים את המטלה, ומחלקם תוכלו להתעלם. ניתן להוריד את כל הקוד והקבצים התומכים כארכיון zip. את הגרסה המקורית של פרויקט זה תוכלו למצוא כאן. פנו אליה כל פעם שתמצאו שהגרסה העברית אינה מובנת.

| | |
|---|---|
| **שני הקבצים היחידים שעליכם לערוך ולהגיש :** | |
| כל אלגוריתמי החיפוש שלכם. | search.py |
| כל הסוכנים המבוססים על החיפוש שלכם. | searchAgents.py |
| | |
| **קבצים שאולי תרצו להציץ לתוכם :** | |
| הקובץ הראשי שמריץ משחקי פאקמן. קובץ זה מתאר את הטיפוס GameState שבו תשתמשו בפרויקט זה | pacman.py |
| ההיגיון מאחורי העולם שבו פאקמן עובד. קובץ זה מתאר מספר טיפוסים תומכים כמו- AgentState, Agent, Direction Grid | game.py |
| מבני נתונים שימושיים ליישום אלגוריתמי חיפוש | util.py |
| | |
| **קבצים נוספים מהם מומלץ להתעלם :** | |
| גרפיקה עבור פאקמן | graphicsDisplay.py |
| תמיכה בגרפיקה של פאקמן | graphicsUtils.py |
| גרפיקת ASCII עבור פאקמן | textDisplay.py |
| הסוכנים השולטים ברוחות | ghostAgents.py |
| ממשקי מקלדת לשליטה בפאקמן | keyboardAgents.py |
| קוד לקריאת קבצי פריסה ואחסון תוכנם | layout.py |
| ה-autograder של הפרויקט | autograder.py |
| מנתח קבצי בדיקות ופתרונות אוטומטיים | testParser.py |
| מחלקות כלליות עבור ה-autograder | testClasses.py |
| ספרייה המכילה את מקרי המבחן עבור כל שאלה | test_cases/ |
| מחלקות ספציפיות עבור ה-autograder | searchTestClasses.py |

**קבצים לעריכה והגשה :** יש להוסיף קוד **אך ורק** בקבצים search.py searchAgents.py.

יש להגיש את שניהם, ביחד עם קובץ ה-readme שכתבתם עבור הפרויקט, ארוזים בקובץ zip. **אין לשנות את הקבצים האחרים, ואין לשלוח קבצים נוספים מעבר לשלושת אלו** שצוינו לעיל. נא לא לשנות את השמות של פונקציות או מחלקות כלשהן שסופקו בתוך הקוד, כיון שהדבר יפגע בעבודתו התקינה של ה-autograder.

עליכם לעבוד **בפייתון גרסה 3.6 – לא מוקדמת יותר ולא מאוחרת יותר**, אחרת הפרויקט שלכם עלול שלא לעבוד.

# ברוכים הבאים לפאקמן

לאחר הורדת הקוד (כאמור הקישור הוא search.zip) ופתיחתו אתם אמורים להיות מסוגלים לשחק בפאקמן על ידי הקלדת הפקודה הבאה בשורת הפקודה (cmd) בתיקייה אליה הורדתם את הקבצים :

```
python pacman.py
```

גם בהמשך כל אימת שמופיעה פקודת הרצה של קובץ פייתון, עליכם להקליד אותה בשורת הפקודה. כל הפקודות המופיעות בהמשך מרוכזות בקובץ commands.txt.

פאקמן חי בעולם כחול ונוצץ של מסדרונות מתפתלים ופינוקים עגולים וטעימים. ניווט בעולם הזה ביעילות יהיה הצעד הראשון של פאקמן בשליטה בתחום שלו.

הסוכן הפשוט ביותר בקובץ searchAgents.py מכונה GoWestAgent , שתמיד הולך מערבה (סוכן רפלקס טריוויאלי.) סוכן זה יכול לנצח מדי פעם :

```
python pacman.py --layout testMaze --pacman GoWestAgent
```

אבל, הדברים נעשים מכוערים עבור הסוכן הזה כשנדרשת פנייה :

```
python pacman.py --layout tinyMaze --pacman GoWestAgent
```

אם פאקמן נתקע, באפשרותכם לצאת מהמשחק על ידי הקלדת CTRL-c בחלון הטרמינל.

בסעיפים הבאים הסוכן שלכם ידע לפתור לא רק tinyMaze אלא כל מבוך שתרצו.

שימו לב : pacman.py תומך במספר אפשרויות שאת כל אחת מהן אפשר לכתוב בשורת הפקודה בצורה ארוכה (למשל--layout) או בצורה קצרה (למשל -l). אתם יכולים לראות את רשימת כל האפשרויות ואת ערכי ברירת המחדל שלהן באמצעות :

```
python pacman.py -h
```

---

## שאלה 1 (3 נקודות): מציאת נקודת מזון קבועה באמצעות חיפוש לעומק (DFS)

בקובץ searchAgents.py תמצאו יישום מלא של SearchAgent, אשר מתכנן מסלול דרך עולמו של פאקמן ולאחר מכן הולך במסלול הזה צעד אחר צעד. אלגוריתמי החיפוש אשר בונים את תוכנית המסלול אינם ממומשים - זה התפקיד שלכם.

ראשית, בדקו שהקובץ SearchAgent פועל כהלכה על ידי הפעלת :

```
python pacman.py -l tinyMaze -p SearchAgent -a fn=tinyMazeSearch
```

פקודה זאת אומרת לסוכן SearchAgent להשתמש ב- tinyMazeSearch (אשר מיושם בקובץ search.py) בתור אלגוריתם החיפוש שלו. פאקמן אמור לנווט במבוך בהצלחה.

עכשיו הגיע הזמן לכתוב פונקציות חיפוש גנריות מלאות כדי לעזור לפאקמן לתכנן מסלולים ! פסאודוקוד עבור כל אחד מאלגוריתמי החיפוש שתכתבו ניתן למצוא בספר. זיכרו שכל צומת במרחב המצבים בו מתנהל החיפוש חייב להכיל לא רק מצב, אלא גם את המידע הדרוש לשחזור המסלול שמגיע למצב זה.

***הערה חשובה :*** כל פונקציות החיפוש שלכם צריכות להחזיר רשימה של *פעולות* שיובילו את הסוכן מההתחלה אל המטרה. כל הפעולות הללו צריכות להיות מהלכים חוקיים (למשל, אין לעבור דרך קירות.)

***הערה חשובה :*** הקפידו **להשתמש במבני הנתונים** Stack, Queue, ו- PriorityQueue שמצויים בקובץ util.py. למבני הנתונים הללו יש מאפיינים מסוימים הנדרשים ל-autograder.

*הדרכה :* כל האלגוריתמים דומים מאוד. האלגוריתמים DFS, BFS, UCS ו -A* שונים רק באופן ניהול החזית. לכן מוטב להתרכז בכתיבה נכונה של האלגוריתם הראשון, DFS . אם תעשו את זה נכון, השאר אמור להיות פשוט יחסית.

ישמו את האלגוריתם חיפוש לעומק (DFS) בפונקציה depthFirstSearch בקובץ search.py. כתבו את גרסת graph search של DFS, אשר נמנעת מפיתוח מצבים שכבר ביקרתם בהם.

הקוד שלכם אמור למצוא במהירות פתרון בהרצת הפקודות הבאות :

```
python pacman.py -l tinyMaze -p SearchAgent

python pacman.py -l mediumMaze -p SearchAgent

python pacman.py -l bigMaze -z .5 -p SearchAgent
```

המבוך בו משוטט פאקמן יציג שכבת-על של המצבים שנחקרו, והסדר שבו הם נחקרו (אדום בהיר יותר פירושו חקירה מוקדמת יותר.) האם סדר החקירה הוא מה שהייתם מצפים ? האם פאקמן באמת הולך לכל המשבצות שנחקרו בדרכו אל המטרה ?

*רמז :* אם אתם משתמשים ב- Stack כמבנה הנתונים שלכם, הפתרון שמצא אלגוריתם ה-DFS שלכם עבור mediumMaze צריך להיות באורך של 130 (בתנאי שאתם דוחפים צמתים שכנים לחזית בסדר שסופק על ידי getSuccessors . אתם עשויים לקבל מסלול באורך 246 אם תדחפו אותם בסדר הפוך.) האם זה המסלול הקצר ביותר ? אם לא, חישבו מה החיפוש לעומק עושה לא נכון.

הפעילו את הפקודה למטה כדי לראות אם היישום שלכם עובר את כל הבדיקות של ה-autograder :

```
python autograder.py -q q1
```

## שאלה 2 (3 נקודות): Breadth First Search

ישמו את האלגוריתם חיפוש לרוחב (BFS) בפונקציה breadthFirstSearch בקובץ search.py. כתבו את גרסת graph search של BFS, אשר נמנעת מפיתוח מצבים שכבר ביקרתם בהם. בידקו את הקוד שלכם באותו אופן שעשיתם עבור חיפוש לעומק.

```
python pacman.py -l mediumMaze -p SearchAgent -a fn=bfs

python pacman.py -l bigMaze -p SearchAgent -a fn=bfs -z .5
```

האם BFS מוצא את הפתרון הקצר ביותר ? אם לא, בידקו את המימוש שלכם.

*רמז :* אם פאקמן זז לאט מדי, נסו את האפשרות --frameTime 0 .

*הערה :* אם כתבתם את קוד החיפוש שלכם באופן כללי, הקוד שלכם אמור לעבוד באותה מידה עבור בעיית החיפוש של שמונת האריחים ללא שינויים.

```
python eightpuzzle.py
```

הפעילו את הפקודה למטה כדי לראות אם היישום שלכם עובר את כל הבדיקות של ה-autograder :

```
python autograder.py -q q2
```

## שאלה 3 (3 נקודות): שינוי פונקציית העלות

בעוד ש- BFS ימצא את המסלול הקצר ביותר למטרה, אולי נרצה למצוא נתיבים שהם "הטובים ביותר" במובנים אחרים. בחנו את mediumDottedMaze ואת mediumScaryMaze .

על ידי שינוי פונקציית העלות, נוכל לעודד את פאקמן למצוא מסלולים שונים. לדוגמה, אנו יכולים לשלם מחיר גבוה יותר עבור צעדים מסוכנים באזורים מוכי רוחות רפאים, או לשלם פחות עבור צעדים באזורים עשירים במזון. סוכן פאקמן רציונלי צריך להתאים את התנהגותו לסיטואציה.

כיתבו את גרסת graph search של UCS בפונקציה uniformCostSearch בקובץ search.py.מומלץ לחפש בקובץ util.py כמה מבני נתונים שעשויים להיות שימושיים ביישום שלכם. כעת עליכם להבחין בהתנהגות מוצלחת בכל שלושת הבעיות הבאות, כאשר הסוכנים שלהלן הם כולם סוכני UCS שנבדלים זה מזה רק בפונקציית העלות שבה הם משתמשים (הסוכנים ופונקציות העלות נכתבו עבורכם) :

```
python pacman.py -l mediumMaze -p SearchAgent -a fn=ucs

python pacman.py -l mediumDottedMaze -p StayEastSearchAgent

python pacman.py -l mediumScaryMaze -p StayWestSearchAgent
```

*הערה :* אתם אמורים לקבל עלויות נתיב נמוכות מאוד וגבוהות מאוד עבור StayEastSearchAgent ו- StayWestSearchAgent בהתאמה, בשל פונקציות העלות האקספוננציאליות שלהם (ראו בקובץ searchAgents.py לפרטים.)

*הפעילו* את הפקודה למטה כדי לראות אם היישום שלכם עובר את כל הבדיקות של ה-autograder.

```
python autograder.py -q q3
```

## שאלה 4 (3 נקודות): חיפוש *A

ישמו חיפוש גרף *A בפונקציה הריקה aStarSearch בקובץ search.py. *A מקבל פונקציה היוריסטית כארגומנט. יוריסטיקה מקבלת שני ארגומנטים: מצב בבעיית החיפוש (הארגומנט העיקרי), והבעיה עצמה (כמידע לייחוס.) הפונקציה היוריסטית nullHeuristic בקובץ search.py היא דוגמה טריוויאלית.

אתם יכולים לבדוק את יישום ה- *A שלכם על הבעיה המקורית של מציאת נתיב דרך מבוך למיקום קבוע באמצעות יוריסטיקה שמחשבת מרחק מנהטן (מיושמת כ- manhattanHeuristic בקובץ searchAgents.py.)

```
python pacman.py -l bigMaze -z .5 -p SearchAgent -a fn=astar,heuristic=manhattanHeuristic
```

אתם אמורים לראות ש- *A מוצא את הפתרון האופטימלי מעט יותר מהיר מ-UCS (בערך 549 לעומת 620 צמתי חיפוש שהורחבו ביישום שלנו, אך מצבים של שוויון בערכם של צמתים עשויים לגרום למספרים שלכם להיות שונים מעט). מה קורה ב- openMaze לגבי אסטרטגיות החיפוש השונות ?

*הפעילו* את הפקודה למטה כדי לראות אם היישום שלכם עובר את כל הבדיקות של ה-autograder.

```
python autograder.py -q q4
```

## שאלה 5 (3 נקודות): למצוא את כל הפינות

הכוח האמיתי של A* יהיה ברור רק עם בעיית חיפוש מאתגרת יותר. הגיע הזמן לגבש בעיה חדשה ולעצב עבורה יוריסטיקה.

*במבוכי פינות* יש ארבע נקודות, אחת בכל פינה. בעיית החיפוש החדשה שלנו היא למצוא את הדרך הקצרה ביותר דרך המבוך שעוברת בכל ארבע הפינות (בין אם במבוך יש שם אוכל ובין אם לאו.) שימו לב שעבור מבוכים מסוימים כמו tinyCorners הדרך הקצרה ביותר לא תמיד מגיעה תחילה לאוכל הקרוב ביותר !

*רמז* : במסלול הקצר ביותר tinyCorners יש 28 צעדים.

*הערה: הקפידו להשלים את שאלה 2 לפני העבודה על שאלה 5, כי שאלה 5 מבוססת על תשובתכם לשאלה 2.*

ישמו את בעיית החיפוש CornersProblem בקובץ searchAgents.py. תצטרכו לבחור ייצוג מצב שמקודד את כל המידע הדרוש כדי לזהות אם הושגו כל ארבע הפינות.

כעת סוכן החיפוש שלכם אמור לפתור :

```
python pacman.py -l tinyCorners -p SearchAgent -a fn=bfs,prob=CornersProblem

python pacman.py -l mediumCorners -p SearchAgent -a fn=bfs,prob=CornersProblem
```

כדי לקבל ניקוד מלא עליכם להגדיר ייצוג מצב מופשט *שאינו* מקודד מידע לא רלוונטי (כמו מיקום רוחות רפאים, היכן אוכל נוסף וכו'.) בפרט, אל תשתמשו ב- GameState של פאקמן כמצב חיפוש. הקוד שלכם יהיה מאוד מאוד איטי (וגם שגוי) אם תעשו זאת.

*רמז 1 :* החלקים היחידים של מצב המשחק שאתם צריכים להתייחס אליהם ביישום שלכם הם המיקום ההתחלתי של פאקמן והמיקום של ארבע הפינות.

*רמז 2 :* בעת הקידוד של getSuccessors הקפידו להוסיף צמתים בנים לרשימת הצאצאים שלכם בעלות של 1.

היישום שלנו של breadthFirstSearch מפתח קצת פחות מ-2000 צמתי חיפוש ב-mediumCorners . עם זאת, חיפוש A* עם יוריסטיקה יכול להפחית את כמות החיפוש הנדרשת.

*הפעילו* את הפקודה למטה כדי לראות אם היישום שלכם עובר את כל הבדיקות של ה-autograder.

```
python autograder.py -q q5
```

## שאלה 6 (3 נקודות): בעיית פינות היוריסטית

*הערה : הקפידו להשלים את שאלה 4 לפני העבודה על שאלה 6, כי שאלה 6 מבוססת על תשובתכם לשאלה 4.*

ישמו יוריסטיקה עקבית לא טריוויאלית עבור ה-CornersProblem ב-cornersHeuristic.

```
python pacman.py -l mediumCorners -p AStarCornersAgent -z 0.5
```

*הערה* : AStarCornersAgent הוא קיצור דרך עבור

```
-p SearchAgent -a fn=aStarSearch,prob=CornersProblem,heuristic=cornersHeuristic
```

***קבילות לעומת עקביות :*** שימו לב, יוריסטיקות הן רק פונקציות שמקבלות מצבי חיפוש ומחזירות מספרים שמעריכים את העלות ליעד הקרוב ביותר. יוריסטיקה יעילה יותר תחזיר ערכים קרובים יותר לעלויות בפועל של היעד. כדי להיות *קבילים*, הערכים היוריסטיים חייבים להיות נמוכים מעלות הנתיב הקצר ביותר בפועל ליעד הקרוב ביותר (אך לא שליליים.) כדי *שיוריסטיקה תהיה עקבית*, יש לדאוג לכך שאם פעולה עלתה c, אזי נקיטת פעולה זו תגרום לירידה ביוריסטיקה של *לכל היותר c*.

שימו לב, קבילות אינה מספיקה כדי להבטיח נכונות בחיפוש גרף - אתם צריכים את התנאי החזק יותר של עקביות. עם זאת, יוריסטיקות קבילות הן בדרך כלל גם עקביות, במיוחד אם הן נגזרות מהקלה (relaxation) של בעיות. לכן בדרך כלל הכי קל להתחיל בסיעור מוחות של יוריסטיקות קבילות. ברגע שיש לכם יוריסטיקה קבילה שעובדת היטב, תוכלו לבדוק אם היא גם עקבית. הדרך היחידה להבטיח עקביות היא באמצעות הוכחה פורמלית. עם זאת, לעתים קרובות ניתן לזהות חוסר עקביות על ידי אימות שלכל צומת שאתם מפתחים, הצמתים הבנים שלו שווים או גבוהים יותר ב-f-value. יתרה מכך, אם UCS ו -A* יחזירו אי פעם נתיבים באורכים שונים, היוריסטיקה שלכם לא עקבית. אכן, לא פשוט.

***יוריסטיקה לא טריוויאלית: יוריסטיקה*** טריוויאלית אחת היא זו שמחזירה אפס לכל צומת (מה שנותן לנו בעצם את UCS), ולא תחסוך כל זמן חישוב. יוריסטיקה טריוויאלית אחרת היא זו שמחשבת את עלות ההשלמה האמיתית, והיא תיקח כל כך הרבה זמן עד שה-autograder יפסיק את עבודתו כיון שיש לו מגבלת זמן ריצה. אנו רוצים יוריסטיקה שמפחיתה את זמן החישוב הכולל, אם כי עבור משימה זו ה-autograder יבדוק רק את ספירת הצמתים (כאמור, מלבד אכיפת מגבלת זמן סבירה).

היוריסטיקה שלכם חייבת להיות יוריסטיקה עקבית לא טריוויאלית ולא שלילית כדי לקבל נקודות כלשהן. ודאו שיוריסטיקה שלכם מחזירה 0 בכל מצב יעד ולעולם לא מחזירה ערך שלילי. הציון שלכם יהיה בהתאם למספר הצמתים שהיוריסטיקה שלכם מפתחת :

| ציון | מספר הצמתים שפותחו |
|---|---|
| 0/3 | יותר מ-2000 |
| 1/3 | לכל היותר 2000 |
| 2/3 | לכל היותר 1600 |
| 3/3 | לכל היותר 1200 |

*אם היוריסטיקה שלכם לא עקבית*, לא תקבלו כלל נקודות על סעיף זה.

*הפעילו* את הפקודה למטה כדי לראות אם היישום שלכם עובר את כל הבדיקות של ה-autograder.

```
python autograder.py -q q6
```

## שאלה 7 (4 נקודות): לאכול את כל הנקודות

עכשיו נפתור בעיית חיפוש קשה : לאכול את כל האוכל של פאקמן בכמה שפחות שלבים .לשם כך נזדקק להגדרה חדשה של בעיית חיפוש אשר מנסחת את בעיית פינוי המזון : FoodSearchProblem בקובץ searchAgents.py (מיושמת עבורכם.) פתרון מוגדר כמסלול שאוסף את כל האוכל בעולם הפאקמן. עבור הפרויקט הנוכחי, הפתרונות אינם לוקחים בחשבון רוחות רפאים או כדורי כוח. הפתרונות תלויים רק במיקום הקירות, מזון רגיל ופאקמן. לרוחות רפאים, שיכולות להחריב את הפתרון נגיע בפרויקט הבא. אם כתבתם נכון את שיטות החיפוש הכלליות שלכם, A* עם יוריסטיקה ריקה (שווה ערך ל-UCS) אמור למצוא במהירות פתרון אופטימלי לבעיית testSearch ללא שינוי קוד מצידכם (עלות כוללת של 7).

```
python pacman.py -l testSearch -p AStarFoodSearchAgent
```

*הערה* : AStarFoodSearchAgent הוא קיצור דרך עבור

```
-p SearchAgent -a fn=astar,prob=FoodSearchProblem,heuristic=foodHeuristic
```

אתם אמורים לגלות ש-UCS מתחיל להאט אפילו עבור הבעיה הפשוטה יחסית tinySearch. להשוואה, היישום שלנו לוקח 2.5 שניות כדי למצוא נתיב באורך 27 לאחר הרחבת 5057 צמתי חיפוש.

*הערה : הקפידו להשלים את שאלה 4 לפני העבודה על שאלה 7, כי שאלה 7 מבוססת על תשובתכם לשאלה 4.*

כתבו בתוך foodHeuristic בקובץ searchAgents.py יוריסטיקה עקבית עבור בעיית FoodSearchProblem : נסו את הסוכן שלכם במבוך trickySearch .

```
python pacman.py -l trickySearch -p AStarFoodSearchAgent
```

סוכן ה- UCS שלנו מוצא את הפתרון האופטימלי תוך כ-13 שניות, ובוחן למעלה מ-16,000 צמתים.

כל יוריסטיקה עקבית לא טריוויאלית לא שלילית תקבל לפחות נקודה אחת. ודאו שהיוריסטיקה שלכם מחזירה 0 בכל מצב יעד ולעולם לא מחזירה ערך שלילי. הציון שלכם יהיה בהתאם למספר הצמתים שהיוריסטיקה שלכם מפתחת :

| ציון | מספר הצמתים שפותחו |
|---|---|
| 1/4 | יותר מ-15000 |
| 2/4 | לכל היותר 15000 |
| 3/4 | לכל היותר 12000 |
| 4/4 | לכל היותר 9000 |
| 5/4 (בונוס) | לכל היותר 7000 |

*שימו לב : אם היוריסטיקה שלכם לא עקבית, לא* תקבלו ניקוד על סעיף זה.

האם הפתרון שלכם ל- mediumSearch מסתיים תוך בזמן קצר ? אם כן, או שעשיתם עבודה מצויינת, או שהיוריסטיקה שלכם לא עקבית.

*הפעילו* את הפקודה למטה כדי לראות אם היישום שלכם עובר את כל הבדיקות של ה-autograder.

```
python autograder.py -q q7
```

## שאלה 8 (3 נקודות): חיפוש לא אופטימלי

לפעמים, אפילו עם A* ויוריסטיקה טובה, קשה למצוא את הדרך האופטימלית דרך כל הנקודות. במקרים שכאלה, עדיין נרצה למצוא – במהירות - דרך טובה למדי. בחלק זה תכתבו סוכן שתמיד אוכל בתאווה את הנקודה הקרובה ביותר. ClosestDotSearchAgent מיושם עבורכם בקובץ searchAgents.py, אך חסרה בה פונקציית מפתח שמוצאת נתיב לנקודה הקרובה ביותר.

יישמו את הפונקציה findPathToClosestDot בקובץ searchAgents.py. הסוכן שלנו מוצא פתרון (לא אופטימלי!) עבור המבוך הזה תוך פחות משנייה עם עלות נתיב של 350 :

```
python pacman.py -l bigSearch -p ClosestDotSearchAgent -z .5
```

*רמז* : הדרך המהירה ביותר להשלים את findPathToClosestDot היא למלא את AnyFoodSearchProblem , שחסרה בו הבדיקה האם מצב הוא מצב מטרה. לאחר מכן פתרו את הבעיה עם פונקציית חיפוש מתאימה. הפתרון צריך להיות קצר מאוד !

ClosestDotSearchAgent לא תמיד תמצא את הדרך הקצרה ביותר דרך המבוך. ודאו שאתם מבינים מדוע, ונסו להמציא דוגמה קטנה שבה הליכה חוזרת ונשנית לנקודה הקרובה ביותר אינה מביאה למציאת הדרך הקצרה ביותר לאכילת כל הנקודות.

*הפעילו* את הפקודה למטה כדי לראות אם היישום שלכם עובר את כל הבדיקות של ה-autograder.

```
python autograder.py -q q8
```

# מטלת מנחה (ממ"ן) 12

**הקורס:** **20551 – מבוא לבינה מלאכותית**

**חומר הלימוד למטלה:** פרקים 1-6

**מספר השאלות:** 4 **משקל המטלה:** 3 נקודות

**סמסטר:** 2023ב **מועד אחרון להגשה:** 7.4.2023

(אב)

**קיימות שתי חלופות להגשת מטלות:**

- שליחת מטלות באמצעות מערכת המטלות המקוונת באתר הבית של הקורס (מומלץ מאוד.)
- שליחת מטלות באמצעות הדואר או הגשה ישירה למנחה במפגשי ההנחיה (מאוד לא מומלץ.)

**הסבר מפורט ב"נוהל הגשת מטלות מנחה"**

**שאלה 1 (20 נק')**

נתונה מסילה מעגלית המחולקת למספר כלשהו n של מקטעים (n>2) הממוספרים מ-0 עד n-1.

בתחילה הסוכן נמצא בתחילת מקטע מספר 0 (בגבול שבין מקטע n-1 ומקטע 0), והמהירות שלו היא 0 מקטעים לדקה.

הסוכן יכול לבצע פעולה בתחילתה של כל דקה. הפעולות הן "האץ" ו"האט" או "הישאר באותה המהירות":

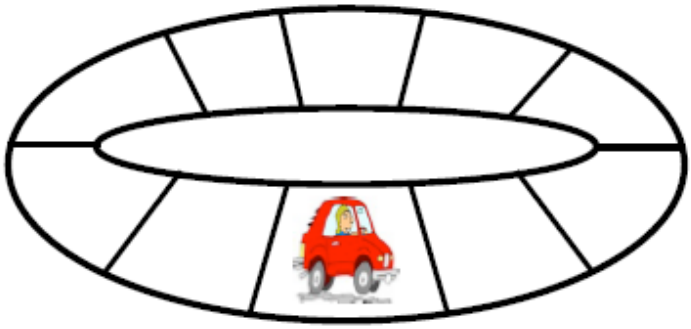
פעולת "האץ" גורמת לכך שבדקה הבאה הסוכן יעבור m+1 מקטעים, כאשר m הוא מספר המקטעים שהסוכן עבר בדקה הנוכחית.

פעולת "האט" גורמת לכך שבדקה הבאה הסוכן יעבור m-1 מקטעים, כאשר m הוא מספר המקטעים שהסוכן עבר בדקה הנוכחית.

אי אפשר להשתמש בפעולה "האט" אם המהירות הנוכחית היא 0 מקטעים לדקה.

מחיר כל פעולה הוא 1.

מטרת הסוכן היא להסתובב במסילה (תוך נסיעה במכונית) k פעמים (k>=1) ואז לחנות בנקודת המוצא (במהירות 0 מקטעים לדקה). הסוכן צריך לבצע את הנדרש <u>במינימום פעולות</u>.



א. תארו את מרחב המצבים עבור בעיה זו: קבוצת המצבים האפשריים (השתדלו שהיצוג של מצב יהיה קומפקטי ככל האפשר), הפעולות האפשריות (בהתאם ליצוג שבחרתם למצבים), המצב ההתחלתי והמצב הסופי (מבחן המטרה).

ב. האם מובטח שחיפוש לעומק הינו שלם עבור בעיה זו? נמקו את תשובתכם.

ג. האם מובטח שחיפוש לרוחב יהיה אופטימלי עבור בעיה זו? נמקו את תשובתכם.

ד. נתונה היוריסטיקה הבאה לבעיה:

מספר המקטעים בין המיקום הנוכחי של הסוכן ובין מיקומו הסופי.

(פורמלית, אם הסוכן נמצא במקטע ה-z, אז היוריסטיקה מחזירה:

n-z אם $z \neq 0$

0 אם $z=0$. )

האם יוריסטית זו קבילה?

האם יוריסטיקה זו עקבית?

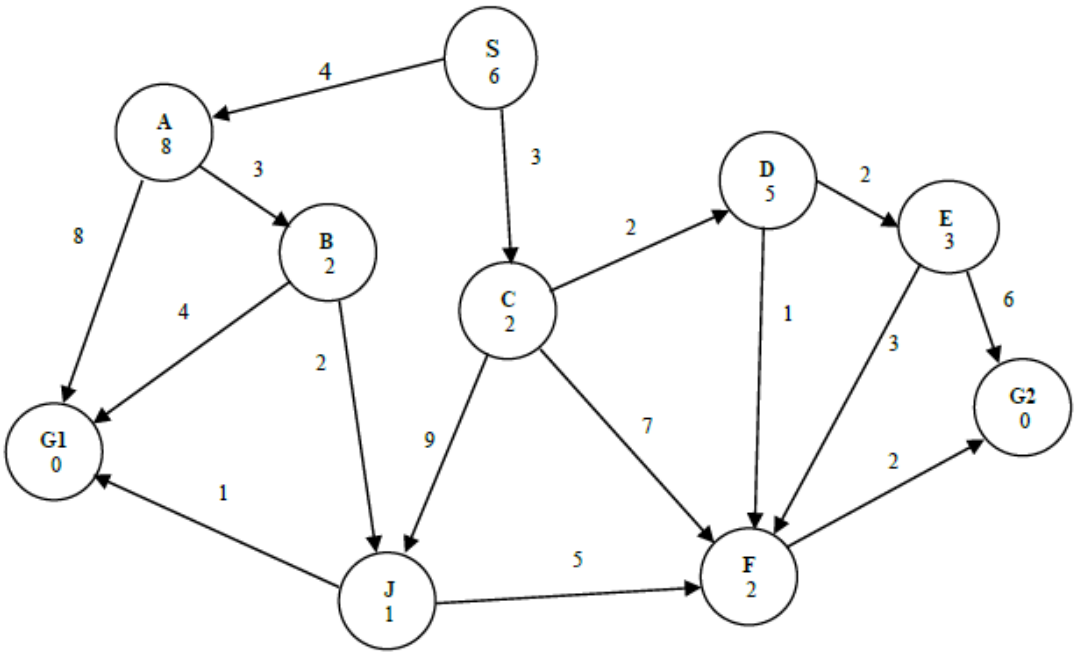
הוכיחו את תשובותיכם.

**שאלה 2 (35 נק')**

א. נתון גרף מרחב מצבים שלהלן.

S הוא המצב (צומת) ההתחלתי ו- G1, G2 הם מצבי סיום (מקיימים את מבחן המטרה).

מחירי המעברים בין המצבים רשומים על הקשתות; הערכים היוריסטיים (ערכי הפונקציה h=עלות משוערת למטרה) רשומים בצמתים.




עבור כל אחת מאסטרטגיות החיפוש שלהלן, כתבו לאילו מצבי מטרה ניתן להגיע (אם בכלל) על-ידי האלגוריתם ורשמו על-פי הסדר את הצמתים המוצאים מהחזית (frontier) במהלך ריצת האלגוריתם.

הניחו כי כל האלגוריתמים משתמשים ב-explored set (למעט עבור Iterative Deepeninig) ובמידה ולשני צמתים או יותר יש עדיפות שווה, יש לבחור בסדר אלפביתי (A עדיף על B).

הניחו שלא מתבצעות בדיקות למניעת מופעים כפולים של צמתים במסלול.

- האם הפונקציה היוריסטית הנתונה h קבילה (admissible)? הסבירו את תשובתכם.
- האם הפונקציה היוריסטית הנתונה h עקבית (consistent)? הסבירו את תשובתכם.

1. BFS
2. Iterative Deepening
3. Uniform Cost Search
4. Greedy Best First Search
5. A*
6. Hill Climbing
7. Local Beam Search (עם k=2)

ב. השתמשו במרחב המצבים שבסעיף א' תוך התייחסות לשיטת החיפוש הדמיית חישול.
הניחו כי הטמפרטורה הנוכחית היא 100.

1. אם נמצאים בצומת C ואלגוריתם הדמיית חישול בחר באופן אקראי בצומת J, מהי ההסתברות שיבוצע מהלך זה?
2. אם נמצאים בצומת C ואלגוריתם הדמיית חישול בחר באופן אקראי לבחון את צומת D, מהי ההסתברות שיבוצע מהלך זה?

**שאלה 3 (23 נק': 5 נק' לסעיפים א',ב',ד'; 8 נק' לסעיף ג')**

נתייחס לגרף לא מכוון שקבוצת הצמתים שלו היא V וקבוצת הקשתות שלו היא E.

נרצה לחלק את קבוצת צמתי הגרף לשתי קבוצות זרות: V1 ו-V2 כך ש:

- מספר הצמתים ב-V1 קרוב ככל האפשר ל- מספר הצמתים ב-V2
- מספר הקשתות שהקצה האחד שלהן הוא צומת ב-V1 והקצה האחר שלהן הוא צומת ב-V2 – קטן ככל האפשר.

שימו לב כי הדרישות (בהגדרת הבעיה) אינן מוגדרות באופן חד משמעי וזאת באופן מכוון.

**למשל**: בהנחה ש-n הוא מספר הקשתות בין שתי הקבוצות הזרות, מה עדיף:

גרף בו |V1|<|V2| ו-n=x

**או**

גרף בו |V1|=|V2| ו-n>x ?

הפונקציה היוריסטית שתבחרו היא זו שתקבע כיצד לשקלל את שתי הדרישות.

א. כתבו פונקציה יוריסטית שתוכל לשמש להערכת מצב.

ב. הסבירו כיצד יתבצע אלגוריתם טיפוס גבעה על פונקציה זו: מהם המצבים ומהי קבוצת המצבים השכנים של מצב נתון?

ג. הסבירו כיצד ניתן לקודד בעיה זו עבור אלגוריתם גנטי:

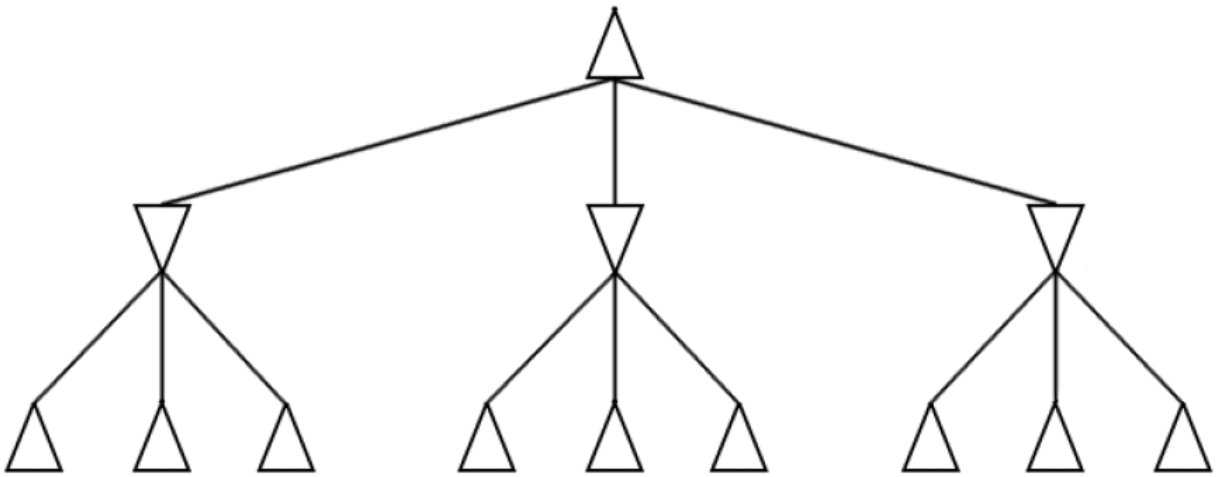
כיצד יקודדו הפרטים (individuals), מהי פונקציית ההתאמה (fitness), ומהן פעולות המוטציה וההצלבה.

ד. באיזה מהאלגוריתמים הבאים עדיף להשתמש לפתרון בעיה זו? נמקו את תשובתכם.

- o טיפוס גבעה
- o הדמיית חישול
- o אלגוריתמים גנטיים

**שאלה 4 (22 נק׳)**

נתון עץ המשחק הבא:



הניחו כי העץ מפותח משמאל לימין.

א. הציבו לעלי העץ את הערכים הבאים: 1,2,3,4,5,6,7,8,9 כך שאלגוריתם אלפא-ביתא יגזום מספר מקסימלי של צמתים.

ב. הציבו לעלי העץ את הערכים הבאים: 1,2,3,4,5,6,7,8,9 כך שאלגוריתם אלפא-ביתא יגזום מספר מינימלי של צמתים.

ג. הציעו משחק שבו ניתן להגיע לאותו מצב במשחק דרך מסלולים שונים בעץ המשחק (כלומר, צמתים שונים בעץ המשחק מייצגים את אותו מצב במשחק).

נניח שהשתמשנו בגיזום אלפא-ביתא על עץ המשחק שהצעתם והאלגוריתם קבע שניתן לגזום מספר צמתים שהם עוקבים לצומת v בעץ. נניח שכאשר נבצע חיפוש בחלק אחר של אותו העץ, נגיע לאותו מצב של המשחק בצומת 'v.

הוכיחו (או הביאו דוגמה נגדית) לכך שהאלגוריתם בוודאות יגזום את העוקבים של הצומת 'v מעץ החיפוש.

# מטלת מנחה (ממ"ן) 13 - להרצה

**הקורס:** **20551 – מבוא לבינה מלאכותית**

**חומר הלימוד למטלה:** פרק 6

**מספר השאלות:** 5 **משקל המטלה:** 5

**סמסטר:** 2023ב **מועד אחרון להגשה:** 21.4.2023

(אב)

**מטלת הרצה ניתן להגיש בדרך אחת ויחידה :**

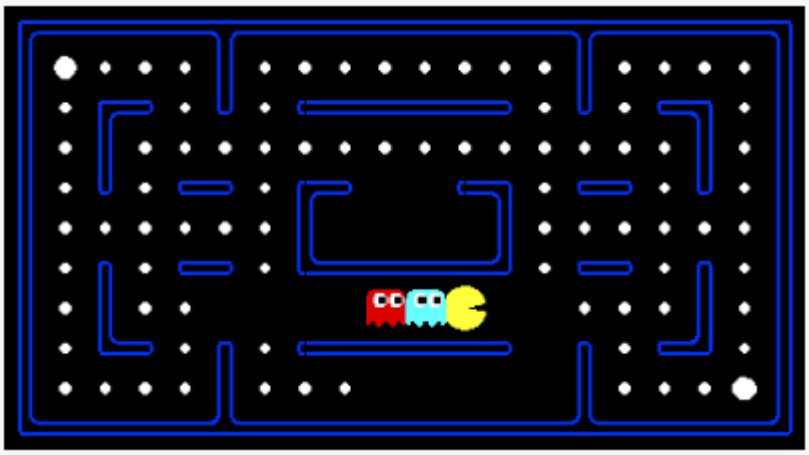
- שליחת מטלות באמצעות מערכת המטלות המקוונת באתר הבית של הקורס
**הסבר מפורט ב"נוהל הגשת מטלות מנחה"**

קראו את הפתיח לממ"ן 11.

המטלה תיבדק ע"י ה-autograder (כמתואר בהמשך) והציון למטלה יינתן באופן הבא:

- הציון שה-autograder נותן יהווה 90%.
- 10% הנותרים ינתנו ע"פ התרשמות הבודק וקובץ ה-readme שעליכם לצרף, ובו תיאור שלכם את העבודה שעשיתם.

## פרוייקט 2 : חיפוש מרובה סוכנים



# מבוא

בפרויקט זה תעצבו סוכנים לגרסה הקלאסית של פאקמן, כולל רוחות רפאים. לאורך הדרך, תממשו גם מינימקס וגם חיפוש Expectimax ותנסו את כוחכם בעיצוב פונקציית הערכה.

בסיס הקוד לא השתנה הרבה מהפרויקט הקודם, אבל אנא התחילו עם התקנה חדשה, במקום ערבוב קבצים מפרויקט 1.

כמו בפרוייקט 1, פרויקט זה כולל autograder, קובץ פייתון שהרצה שלו תאפשר לכם לבדוק את הפרויקט שלכם ולקבל הערכה אודות הציון שתקבלו עליו. כדי להריץ אותו הפעילו את הפקודה הבאה :

```
python autograder.py
```

ניתן גם להריץ אותו עבור שאלה מסוימת אחת, למשל q2, כך :

```
python autograder.py -q q2
```

ניתן להריץ אותו עבור בדיקה מסוימת אחת על ידי פקודות מהצורה :

```
python autograder.py -t test_cases/q2/0-small-tree
```

כברירת מחדל, ה-autograder מציג גרפיקה עם האפשרות -t, אך לא עם האפשרות -q.

באפשרותכם לאלץ הופעת גרפיקה באמצעות האפשרות --graphics, או לאלץ העדר גרפיקה באמצעות האפשרות --no-graphics.

הקוד לפרויקט זה מורכב מכמה קבצי פייתון, שאת חלקם תצטרכו לקרוא ולהבין על מנת להשלים את המטלה, ומחלקם תוכלו להתעלם. ניתן להוריד את כל הקוד והקבצים התומכים כארכיון zip. את הגרסה המקורית של פרויקט זה תוכלו למצוא כאן. פנו אליה כל פעם שתמצאו שהגרסה העברית אינה מובנת.

| | |
|---|---|
| **הקובץ היחיד שעליכם לערוך ולהגיש :** | |
| כאן נמצאים כל סוכני החיפוש מרובי הסוכנים שלכם. | multiAgents.py |
| **קבצים שאולי תרצו להסתכל עליהם :** | |
| הקובץ הראשי שמריץ משחקי Pacman . קובץ זה מתאר גם סוג Pacman GameState, שבו תשתמשו בהרחבה בפרויקט זה. | pacman.py |
| ההיגיון מאחורי איך שעולם פאקמן עובד. קובץ זה מתאר מספר טיפוסים תומכים כמו AgentState, Agent, Direction ו-Grid. | game.py |

| | |
|---|---|
| מבני נתונים שימושיים ליישום אלגוריתמי חיפוש. אינכם צריכים להשתמש באלה עבור פרויקט זה, אך עשויות להיות פונקציות אחרות המוגדרות כאן כדי להיות שימושיות. | util.py |
| **קבצים תומכים שאתם יכולים להתעלם מהם :** | |
| גרפיקה עבור פאקמן | graphicsDisplay.py |
| תמיכה בגרפיקה של Pacman | graphicsUtils.py |
| גרפיקה ASCII עבור Pacman | textDisplay.py |
| סוכנים לשלוט ברוחות | ghostAgents.py |
| ממשקי מקלדת לשליטה בפאקמן | keyboardAgents.py |
| קוד לקריאת קבצי פריסה ואחסון תוכנם | layout.py |
| ה-autograder של הפרויקט | autograder.py |
| מנתח קבצי בדיקות ופתרונות אוטומטיים | testParser.py |
| מחלקות כלליות עבור ה-autograder | testClasses.py |
| ספרייה המכילה את מקרי הבדיקה לכל שאלה | test_cases/ |
| מחלקות ספציפיות עבור ה-autograder לפרויקט זה | multiagentTestClasses.py |

יש להוסיף קוד **אך ורק** בקובץ multiAgents.py.

יש להגיש אותו, ביחד עם קובץ ה-readme שכתבתם עבור הפרויקט, ארוזים בקובץ zip. **אין לשנות את הקבצים האחרים, ואין לשלוח קבצים נוספים מעבר לשני אלו** שצוינו לעיל. נא לא לשנות את השמות של פונקציות או מחלקות כלשהן שסופקו בתוך הקוד, כיון שהדבר יפגע בעבודתו התקינה של ה-autograder.

עליכם לעבוד **בפייתון גרסה 3.6 – לא מוקדמת יותר ולא מאוחרת יותר**, אחרת הפרויקט שלכם עלול שלא לעבוד.

---

# ברוכים הבאים לפאקמן מרובה סוכנים

ראשית, שחקו משחק של פאקמן הקלאסי על ידי הפעלת הפקודה הבאה :

```
python pacman.py
```

השתמשו במקשי החיצים כדי לזוז. כעת הפעילו את ReflexAgent שנמצא בקובץ multiAgents.py

```
python pacman.py -p ReflexAgent
```

שימו לב שהוא משחק די גרוע אפילו בבעיות פשוטות :

```
python pacman.py -p ReflexAgent -l testClassic
```

בדקו את הקוד שלו בקובץ multiAgents.py וודאו שאתם מבינים מה הוא עושה.

## שאלה 1 (4 נקודות) : סוכן רפלקס

שפרו את ReflexAgent בקובץ multiAgents.py כך שהוא ישחק בצורה מכובדת. קוד סוכן הרפלקס שנתון לכם מספק כמה דוגמאות מועילות לשאילתות השואבות מידע מה- GameState. סוכן רפלקס בעל יכולת יצטרך לשקול הן את מיקומי המזון והן את מיקומי רוחות הרפאים כדי להגיע לביצועים טובים. הסוכן testClassic שלכם צריך לנקות את השטח בקלות ובאמינות :

```
python pacman.py -p ReflexAgent -l testClassic
```

נסו את סוכן הרפלקס שלכם mediumClassic בלוח ברירת המחדל עם רוח רפאים אחת או שתיים (ואנימציה כבויה כדי להאיץ את התצוגה) :

```
python pacman.py --frameTime 0 -p ReflexAgent -k 1

python pacman.py --frameTime 0 -p ReflexAgent -k 2
```

איך הסוכן שלכם מסתדר ? סביר להניח שהוא ימות לעתים קרובות עם 2 רוחות רפאים בלוח ברירת המחדל, אלא אם כן פונקציית ההערכה שלכם טובה למדי.

*הערה :* זיכרו של- newFood יש את הפונקציה asList().

*הערה :* כתכונות, נסו את ההדדיות של ערכים חשובים (כגון מרחק לאוכל) ולא רק את הערכים עצמם.

*הערה :* פונקציית ההערכה שאתם כותבים היא הערכת צמדי מצב-פעולה. בחלקים מאוחרים יותר של הפרויקט, אתם תעריכו מצבים.

*הערה :* ייתכן שיהיה שימושי להציג את התוכן הפנימי של אובייקטים שונים לצורך איתור באגים. אתם יכולים לעשות זאת על ידי הדפסת ייצוגי המחרוזת של האובייקטים. לדוגמה, אתם יכולים להדפיס את newGhostStates עם print(newGhostStates).

*אפשרויות :* רוחות ברירת מחדל הן אקראיות. אתם יכולים גם לשחק בשביל הכיף עם רוחות רפאים כיווניות קצת יותר חכמות באמצעות -g DirectionalGhost . אם האקראיות מונעת מכם לדעת אם הסוכן שלכם

משתפר, אתם יכולים להשתמש ב- -f כדי לרוץ עם seed אקראי קבוע (אותן בחירות אקראיות בכל משחק.) אתם יכולים גם לשחק במספר משחקים ברציפות עם -n . כבו את הגרפיקה עם -q כדי להפעיל הרבה משחקים במהירות.

*ניקוד :* ה-autograder מריץ את הסוכן שלכם על openClassic 10 פעמים. תקבלו 0 נקודות אם לסוכן שלכם לקח יותר מדי זמן, או אם הוא לעולם לא מנצח. תקבלו נקודה אחת אם הסוכן שלכם מנצח לפחות 5 פעמים, או 2 נקודות אם הסוכן שלכם מנצח בכל 10 המשחקים. תקבלו תוספת של נקודה אחת אם הציון הממוצע של הסוכן שלכם גבוה מ-500, או 2 נקודות אם הוא גדול מ-1000. אתם יכולים לנסות את הסוכן שלכם בתנאים אלה עם

```
python autograder.py -q q1
```

כדי להפעיל אותו ללא גרפיקה, השתמשו ב :

```
python autograder.py -q q1 --no-graphics
```

עם זאת, אל תבזבזו יותר מדי זמן על השאלה הזו, מכיוון שעיקר הפרויקט עוד לפנינו.

## שאלה 2 (5 נקודות) : Minimax

כעת עליכם לכתוב סוכן חיפוש יריב MinimaxAgent במסגרת המחלקה המצויה בקובץ multiAgents.py . סוכן המינימקס שלכם צריך לעבוד עם כל מספר של רוחות רפאים, ולכן תצטרכו לכתוב אלגוריתם שהוא מעט יותר כללי ממה שלמדתם בשיעור. בפרט, לעץ המינימקס שלכם יהיו מספר שכבות מינימום (אחת לכל רוח רפאים) עבור כל שכבת מקסימום.

הקוד שלכם צריך גם לפתח את עץ המשחק לעומק שרירותי. נקדו את העלים של עץ המינימקס שלכם עם self.evaluationFunction הנתונה לכם שברירת המחדל שלה היא scoreEvaluationFunction. שמרחיבה את MinimaxAgent, מה שנותן גישה ל MultiAgentSearchAgent -self.depth וכן ל- self.evaluationFunction . ודאו שקוד המינימקס שלכם מתייחס לשני המשתנים הללו במידת הצורך, שכן בהם נמצאת התגובה לאפשרויות המופיעות בשורת הפקודה.

*חשוב :* רמת חיפוש בודדת נחשבת למהלך אחד של פאקמן וכל התגובות של הרוחות. מכאן שחיפוש עומק 2 יכלול שתי תזוזות הן של פאקמן והן של כל רוח רפאים.

*ניקוד :* ה-autograder יבדוק את הקוד שלכם כדי לקבוע אם הוא בוחן את המספר הנכון של מצבי משחק. זוהי הדרך האמינה היחידה לזהות כמה באגים עדינים מאוד ביישומים של מינימקס. כתוצאה מכך, ה-autograder יהיה בררן מאוד לגבי מספר הפעמים שבו הקוד שלכם קורא ל-

GameState.generateSuccessor. אם תקראו לה יותר או פחות מהנדרש, ה-autograder יתלונן. כדי לבדוק ולזהות באגים בקוד שלכם, הפעילו

```
python autograder.py -q q2
```

זה יראה מה האלגוריתם שלכם עושה על מספר עצים קטנים, כמו גם על משחק פאקמן. כדי להפעיל אותו ללא גרפיקה, השתמשו ב :

```
python autograder.py -q q2 --no-graphics
```

### *רמזים ותובנות*

- רמז : ישמו את האלגוריתם באופן רקורסיבי באמצעות פונקציות מסייעות.
- יישום נכון של מינימקס יוביל לכך שפאקמן יפסיד את המשחק בחלק מההרצות. זו לא בעיה אלא התנהגות נכונה.
- פונקציית ההערכה עבור בדיקת פאקמן בחלק זה כבר כתובה (self.evaluationFunction ) . אינכם צריכים לשנות את הפונקציה הזו, אלא רק להכיר בכך שעכשיו אנחנו מעריכים *מצבים* ולא פעולות, כפי שעשינו עבור סוכן הרפלקס. סוכני מבט קדימה מעריכים מצבים עתידיים ואילו סוכני רפלקס מעריכים פעולות מהמצב הנוכחי.
- ערכי המינימקס של המצב ההתחלתי minimaxClassic בפריסה הם 9, 8, 7, 492- עבור עומקים 1, 2, 3 ו-4 בהתאמה. שימו לב שסוכן המינימקס שלכם לרוב ינצח (665/1000 משחקים עבורנו) למרות התחזית הקשה של עומק 4 מינימקס.

```
python pacman.py -p MinimaxAgent -l minimaxClassic -a depth=4
```

- פאקמן הוא תמיד סוכן 0, והסוכנים זזים כל אחד בתורו בהתאם לסדר התקדמות אינדקס הסוכן.
- כל המצבים במינימקס צריכים להיות GameStates המועברים אל getAction או נוצרים באמצעות GameState.generateSuccessor . בפרויקט זה לא תבצעו הפשטה למצבים פשוטים.
- על לוחות גדולים יותר כגון openClassic ו- mediumClassic (ברירת המחדל) תגלו שפאקמן טוב ב-לא למות, אבל די גרוע ב-לנצח. לעתים קרובות הוא ישוטט סביב בלי להתקדם. הוא יכול אפילו להסתובב ממש ליד נקודה בלי לאכול אותה כי הוא לא יודע לאן הוא ילך אחרי שאכל את הנקודה הזו. אל תדאגו אם אתם רואים התנהגות שכזו, שאלה 5 תטפל בכל הבעיות הללו.
- כשפאקמן מאמין שמותו הוא בלתי נמנע, הוא ינסה לסיים את המשחק בהקדם האפשרי בגלל העונש הקבוע על תקופת החיים. לפעמים זה הדבר הלא נכון לעשות עם רוחות רפאים אקראיות, אבל סוכני מינימקס תמיד מניחים את הגרוע ביותר :

```
python pacman.py -p MinimaxAgent -l trappedClassic -a depth=3
```

ודאו שאתם מבינים מדוע פאקמן ממהר לרוח הרפאים הקרובה ביותר במקרה הזה.

---

## שאלה 3 (5 נקודות) : גיזום אלפא ביתא

צרו ב-AlphaBetaAgent סוכן חדש שמשתמש בגיזום אלפא ביתא כדי לחקור ביעילות רבה יותר את עץ המינימקס. גם כאן האלגוריתם שלכם יהיה מעט יותר כללי מהפסאודוקוד מהשיעור, כך שחלק מהאתגר הוא להרחיב את היגיון החיתוך אלפא-ביתא כראוי למספר סוכני מזער.

אתם אמורים לראות הגברת מהירות (אולי עומק 3 אלפא-ביתא יפעל במהירות כמו עומק 2 מינימקס). באופן אידיאלי, עומק 3 על smallClassic צריך לפעול תוך מספר שניות בלבד לכל מהלך, אם לא מהר יותר.

```
python pacman.py -p AlphaBetaAgent -a depth=3 -l smallClassic
```

ערכי ה-AlphaBetaAgent מינימקס צריכים להיות זהים לערכי ה-MinimaxAgent מינימקס, אם כי הפעולות שהוא בוחר עשויות להשתנות בגלל התנהגות שונה במצב שבו יש צורך בשובר שוויון. כמקודם ערכי המינימקס של המצב ההתחלתי בפריסה minimaxClassic הם 9, 8, 7, 492- עבור עומקים 1, 2, 3 ו-4 בהתאמה.

*ניקוד :* מכיוון שה-autograder בודק את הקוד שלכם כדי לקבוע אם הוא חוקר את המספר הנכון של מצבים, חשוב שתבצעו גיזום אלפא ביתא בלי לסדר מחדש את הבנים של כל צומת. במילים אחרות, יש לטפל במצבים עוקבים תמיד לפי הסדר המוחזר על ידי GameState.getLegalActions . שוב, אל תקראו ל-GameState.generateSuccessor יותר מהנדרש.

***שלא כמו שנלמד בשיעור, כאן אסור לכם לגזום במקרה של שוויון. הסיבה לכך היא הצורך להתאים את הקוד שלכם ל-autograder.***

הפסאודוקוד שלהלן מייצג את האלגוריתם שעליכם ליישם עבור שאלה זו.

## Alpha-Beta Implementation

α: MAX's best option on path to root
β: MIN's best option on path to root

```
def max-value(state, α, β):
    initialize v = -∞
    for each successor of state:
        v = max(v, value(successor, α, β))
        if v > β return v
        α = max(α, v)
    return v
```

```
def min-value(state , α, β):
    initialize v = +∞
    for each successor of state:
        v = min(v, value(successor, α, β))
        if v < α return v
        β = min(β, v)
    return v
```

שימו לב שאי השוויון בין v לבין אלפא וביתא הוא עתה אי שוויון **חזק**.

```
python autograder.py -q q3
```

זה יראה מה האלגוריתם שלכם עושה על מספר עצים קטנים, כמו גם על משחק פאקמן. כדי להפעיל אותו ללא גרפיקה, השתמשו ב :

```
python autograder.py -q q3 --no-graphics
```

ושוב, יישום נכון של גיזום אלפא ביתא יוביל לכך שפאקמן יפסיד את המשחק בחלק מההרצות. זו לא בעיה אלא התנהגות נכונה.

## שאלה 4 (5 נקודות) : Expectimax

מינימקס ואלפא-ביתא עושים עבודה מצוינת, אבל שניהם מניחים שאתם משחקים נגד יריב שמקבל החלטות אופטימליות. כפי שיודע כל מי שניצח אי פעם ב-איקס עיגול (איקס מיקס דריקס), זה לא תמיד המקרה. בשאלה זו תיישמו את ה-ExpectimaxAgent שהוא שימושי למודלים של התנהגות הסתברותית של סוכנים שעשויים לעשות בחירות לא אופטימליות.

בדומה לבעיות חיפוש וסיפוק אילוצים שנלמדו בשיעור, היופי באלגוריתמים הללו הוא הישימות הכללית שלהם. כדי לזרז את הפיתוח שלכם, סיפקנו כמה מקרי בדיקה המבוססים על עצים גנריים. אתם יכולים למצוא ולפתור באגים ביישום שלכם בעצי המשחק הקטנים באמצעות הפקודה :

```
python autograder.py -q q4
```

איתור באגים ופתרונם במקרי בדיקה קטנים וניתנים לניהול מומלץ, ויעזור לך בהמשך כשדוגמאות תלכנה ותסתבכנה.

ברגע שהאלגוריתם שלכם עובד על עצים קטנים, אתם יכולים לראות את הצלחתו בפאקמן. רוחות רפאים אקראיות הן כמובן לא סוכני מינימקס אופטימליים, ולכן התמודדות איתן באמצעות חיפוש מינימקס עשויה להיות לא מתאימה. ExpectimaxAgent לא ייקח עוד את המינימום על כל פעולות הרפאים, אלא את התוחלת בהתאם להבנה של הסוכן שלכם את אופן פעולת הרוחות. כדי לפשט את הקוד שלכם, נניח שתתמודדו רק מול יריב שבוחר באקראי ובהתאם להתפלגות אחידה מתוך ה-getLegalActions שלו.

כדי לראות כיצד ExpectimaxAgent מתנהג בפאקמן, הפעילו :

```
python pacman.py -p ExpectimaxAgent -l minimaxClassic -a depth=3
```

כעת עליכם לנקוט בגישה פעילה יותר בכל סיטואציה של קרבה עם רוחות רפאים. במיוחד, אם פאקמן יבין שהוא עלול להיות לכוד אבל עשוי להימלט כדי לתפוס עוד כמה חתיכות מזון, הוא לפחות ינסה. חיקרו את התוצאות של שני תרחישים אלה :

```
python pacman.py -p AlphaBetaAgent -l trappedClassic -a depth=3 -q -n 10

python pacman.py -p ExpectimaxAgent -l trappedClassic -a depth=3 -q -n 10
```

אתם צריכים לגלות ש-ExpectimaxAgent שלכם מנצח בערך במחצית מהזמן, בעוד ש- AlphaBetaAgent שלכם תמיד מפסיד. ודאו שאתם מבינים מדוע ההתנהגות כאן שונה ממקרה המינימקס.

כרגיל, יישום נכון של expectimax יוביל לכך שפאקמן יפסיד את המשחק בחלק מההרצות. זו לא בעיה אלא התנהגות נכונה.

## שאלה 5 (6 נקודות) : פונקציית הערכה

כיתבו פונקציית הערכה טובה יותר עבור פאקמן ב- betterEvaluationFunction הנתונה לכם. פונקציית ההערכה צריכה להעריך מצבים, ולא פעולות כמו פונקציית הערכת סוכן הרפלקס שלכם. עם חיפוש בעומק 2, פונקציית ההערכה שלכם צריכה לנקות את הפריסה smallClassic עם רוח רפאים אקראית אחת יותר

ממחצית מהזמן ועדיין לפעול בקצב סביר (כדי לקבל ניקוד מלא על סעיף זה, פאקמן אמור להגיע לממוצע של כ-1000 נקודות כשהוא מנצח.)

*ניקוד :* ה-autograder יפעיל את הסוכן שלכם בפריסה smallClassic עשר פעמים. הניקוד שתקבל פונקציית ההערכה שלכם יהיה כדלקמן :

- אם אתם מנצחים לפחות פעם אחת בלי לעבור את מגבלת הזמן, תקבלו נקודה אחת. כל סוכן שלא יעמוד בקריטריונים אלה יקבל 0 נקודות.
- +1 עבור זכייה לפחות 5 פעמים, +2 עבור זכייה בכל 10 הפעמים
- +1 עבור ציון ממוצע של לפחות 500, +2 עבור ציון ממוצע של לפחות 1000 (כולל ציונים על משחקים אבודים)
- +1 אם המשחקים שלכם נמשכים בממוצע פחות מ-30 שניות על מחשב חזק (ובהתאמה למחשב ביתי.)
- הנקודות הנוספות עבור ציון ממוצע וזמן חישוב יוענקו רק אם תנצחו לפחות 5 פעמים.
- נא לא להעתיק קבצים כלשהם מפרויקט 1, מכיוון שהוא לא יעבור את ה autograder-ב-Gradescope.

אתם יכולים לנסות את הסוכן שלכם בתנאים אלה עם

```
python autograder.py -q q5
```

כדי להפעיל אותו ללא גרפיקה, השתמשו ב :

```
python autograder.py -q q5 --no-graphics
```

# מטלת מנחה (ממ"ן) 14

**הקורס:** **20551 – מבוא לבינה מלאכותית**

**חומר הלימוד למטלה:** פרקים 5-9

**מספר השאלות:** 5 **משקל המטלה:** 3 נקודות

**סמסטר:** 2023ב **מועד אחרון להגשה:** 28.4.2023

(אב)

**קיימות שתי חלופות להגשת מטלות:**

- שליחת מטלות באמצעות מערכת המטלות המקוונת באתר הבית של הקורס (מומלץ מאוד.)
- שליחת מטלות באמצעות הדואר או הגשה ישירה למנחה במפגשי ההנחיה (מאוד לא מומלץ.)

**הסבר מפורט ב"נוהל הגשת מטלות מנחה"**

**שאלה 1 (29 נק')**

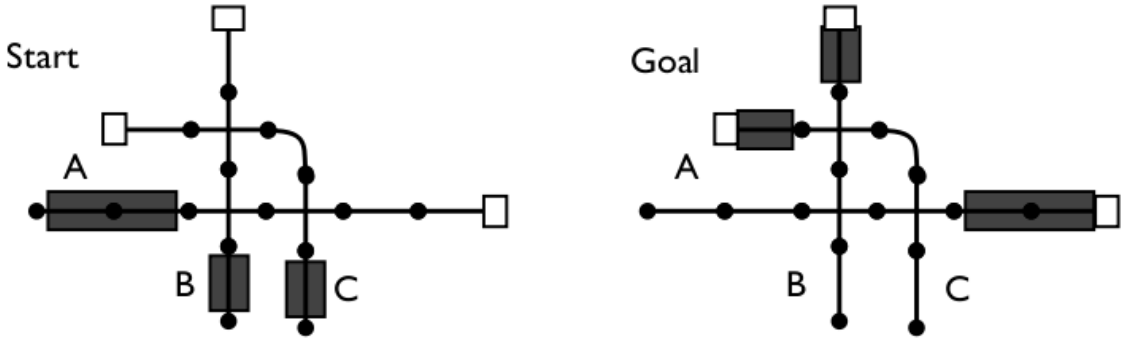
נתונה בעיית בניית לוח זמנים עבור שלוש רכבות: A (בה יש שני קרונות), C,B (קרון אחד בכל אחת) כשרכבת יוצאת לדרכה היא עוברת בנסיעתה יחידת מרחק אחת על מסילתה, בכל יחידת זמן (של שעה, בקפיצות בדידות), עד שמגיעה ליעדה.

זמני יציאת הרכבות הם: 8:00, 9:00 או 10:00.

הרכבות חייבות לצאת בזמנים שונים זו מזו.

שתי רכבות לא יכולות לשהות בצומת באותה יחידת זמן.

מצב ההתחלה ומצב המטרה נתונים באיורים שלהלן:




א. תארו את הבעיה הנתונה כבעיית CSP. המשתנים A,B,C מייצגים את זמני יציאת הרכבות.

ב. ציירו את גרף האילוצים עבור ה-CSP שהגדרתם בסעיף א'.

ג. לאחר בחירת A=9, מחקו את כל הערכים עבור B ו-C שהיו נמחקים על-ידי בדיקה קדימה.

ד. מחקו את כל הערכים שהיו נמחקים על-ידי עקביות קשת לפני ביצוע הצבות למשתנים.

ה. לאחר בחירת A=9, מחקו את כל הערכים עבור B ו-C שהיו נמחקים על-ידי עקביות קשת.

ו. מצאו פתרון (זמני היציאה של כל הרכבות) בעזרת חיפוש backtracking עם בדיקה קדימה (forward checking) ויוריסטיקות MRV ו-LCV.

הראו באיזה סדר מוצבים ערכים למשתנים ואילו ערכים הם מקבלים.
התחילו עם A.

**שאלה 2** **(21 נק':** 6 נק' לסעיף א'; 6 נק' לסעיף ב'; 9 נק' לסעיף ג')

נתונות שלוש קופסאות המסומנות במספרים 1, 2, 3. מתחת לאחת מהקופסאות יש ערימת כסף ומתחת לשתי האחרות לא מונח דבר.

על כל קופסה יש תווית כלהלן:

קופסה 1: "קופסה זו ריקה".

קופסה 2: "קופסה זו ריקה".

קופסה 3: "הכסף מונח מתחת לקופסה2".

המידע הכתוב על גבי אחת התוויות הינו אמת ואילו על שתי התוויות האחרות המידע הכתוב הינו שקרי.

היכן (מתחת לאיזו קופסה) מונח הכסף?

א. ייצגו את המידע הנתון בתחשיב הפסוקים. (הסבירו את משמעות משתנים בהם הנכם משתמשים.)

ב. המירו כל משפט בבסיס המידע לצורת CNF.

ג. לכל אחת מהשאילתות הבאות, הראו בעזרת רזולוציה האם היא נובעת מבסיס המידע:

- הכסף נמצא מתחת לקופסה 1?
- הכסף נמצא מתחת לקופסה 2?
- הכסף נמצא מתחת לקופסה 3?

**שאלה 3 (10 נק')**

ברזולוצית קלט (input resolution) מרשים שימוש בכלל הרזולוציה רק אם לפחות אחת משתי הפסוקיות המשתתפות ברזולוציה שייכת לפסוק המקורי. (כלומר, לא מרשים שימוש בכלל הרזולוציה אם שתי הפסוקיות המשתתפות בגזירת הרזולוציה אינן שייכות לפסוק המקורי).

האם רזולוצית קלט שלמה להפרכה? הוכיחו.

שלמה להפרכה הכוונה: אם לוקחים טענה הנובעת מבסיס הידע ומוסיפים את שלילתה לבסיס הידע, אזי ניתן להסיק פסוק ריק (סתירה) בעזרת כללי ההיסק.

<u>רמז</u>: התבוננו בשלב האחרון של גזירת הפסוקית הריקה מן הפסוק המקורי.

**שאלה 4 (11 נק'**: 3 נק' לסעיף 1.א'; 4 נק' לסעיף 1.ב'; 1 נק' לכל אחד מהסעיפים 2.א' - 2.ד')

1. הציגו את הפסוקים הבאים בלוגיקה מסדר ראשון.

א. יש סַפָּר שמספר את כל האנשים שאינם מספרים את עצמם.

ב. פוליטיקאים יכולים לרמות חלק מן האנשים כל הזמן, והם יכולים לרמות את כל האנשים חלק מהזמן, אך הם אינם יכולים לרמות את כל האנשים כל הזמן.

2. לכל זוג של פסוקים אטומים שלהלן, מצאו את המאחד הכללי ביותר (MGU), אם הוא קיים:

א. $Q(One, Two, Two)$, $Q(x, y, z)$

ב. $R(x, F(A, B))$, $R(F(y, y), x)$

ג. $Younger(Mother(y), y)$, $Younger(Mother(x), Hadar)$

ד. $Likes(Mother(x), x)$, $Likes(y, y)$

**שאלה 5 (29 נק'**: 8 נק' לסעיף א'; 8 נק' לסעיף ב'; 8 נק' לסעיף ג'; 5 נק' לסעיף ד')

אנשי בטחון שומרים על בטחוננו על ידי מעקב אחר מקום הימצאם של פרופסורים, והסקת יחסי יועץ-נועץ. הם מגייסים לעזרתם את ידיעותיהם בתחום הבינה המלאכותית ובעיקר בלוגיקה מסדר ראשון כדי לבצע את עבודתם בצורה מיטבית.

בסיס הידע מורכב מהמשפטים הבאים:

- כל פרופסור מייעץ לסטודנט אחד לפחות.
- לכל סטודנט יש יועץ, שהוא פרופסור.
- כל יועץ נפגש עם כל הנועצים שלו.
- פגישות היעוץ מתקיימות בקמפוס.
- לירן הוא סטודנט.
- הדר היא פרופסור.

א. תרגמו את המשפטים שלעיל ללוגיקה מסדר ראשון.

ב. המירו לצורת CNF.

ג. השתמשו בבסיס הידע שלעיל והוכיחו בעזרת רזולוציה כי "הדר היתה בקמפוס".

ד. נתון המשפט הנוסף הבא:

הדר מייעצת ללירן.

האם ניתן להוכיחו בעזרת רזולוציה מבסיס הידע?

אם כן, הוכיחו; אחרת, הסבירו מדוע לא.

# מטלת מנחה (ממ"ן) 15 - להרצה

**הקורס:** **20551 – מבוא לבינה מלאכותית**

**חומר הלימוד למטלה** : פרקים 1-3

**מספר השאלות:** 1 **משקל המטלה:** 5

**סמסטר:** 2023ב **מועד אחרון להגשה:** 12.5.2023

(אב)

**מטלת הרצה ניתן להגיש בדרך אחת ויחידה :**

- שליחת מטלות באמצעות מערכת המטלות המקוונת באתר הבית של הקורס

**הסבר מפורט ב"נוהל הגשת מטלות מנחה"**

**השוואה בין אלגוריתמי חיפוש במרחב מצבים**

בעיית המיסיונרים והקניבלים היא חידה לוגית קלאסית.

הבעיה עוסקת בשלושה מיסיונרים ושלושה קניבלים הנמצאים בגדה השמאלית של נהר.
בגדה הזו מצויה סירה, שיכולה לשאת אדם אחד או שני אנשים.
אנו מעוניינים להעביר את ששת האנשים לגדה הימנית של הנהר בעזרת הסירה.
מסיבות מובנות, אין לאפשר, אפילו לרגע אחד, מצב שבו מספר הקניבלים גדול ממספר המיסיונרים באחת הגדות של הנהר.
העברת הסירה מגדה לגדה איננה יכולה להתבצע בלי שיהיה בה לפחות אדם אחד.

עליכם לתכנת ארבעה אלגוריתמים המתוארים להלן אשר יפתרו את הבעיה כבעיית חיפוש במרחב מצבים. האלגוריתמים הם :

1. BFS
2. IDDFS
3. GBFS
4. A*

כאשר התוכנית שלכם מופעלת (ללא קבלת קלט כלשהו), היא אמורה להריץ - ממצב ההתחלה ועד מצב הסיום -כל אחד מארבעת האלגוריתמים, בזה אחר זה. הפלט של התוכנית אמור להיות מספר הצמתים שפותחו בכל אחד מארבעת האלגוריתמים, והמסלול שכל אלגוריתם מצא.

עליכם להגיש קבצי מקור בשפת ג'אווה או פייתון. צרפו לעבודה קובץ readme ובו :

1. תיאור כללי של התוכנית – אלגוריתמים, פונקציות ומבני נתונים עיקריים.
2. תיאור של הייצוג שבחרתם למרחב המצבים : מהם המצבים ומהם המעברים.
3. עבור GBFS ו-A* עליכם לפרט מהי היוריסטיקה שבחרתם, והאם היא קבילה ועקבית.
4. עבור כל אלגוריתם עליכם לציין האם המסלול שהוא מצא הוא אופטימלי, ואם לא – מדוע.
5. תיאור של אופן הרצת התוכנית וצילום מסך של פלט מהרצת התוכנית שלכם.

ציון 80 תקבל עבודה שעונה לכל הנדרש לעיל (כולל קובץ readme).

עוד 10 נקודות תקבל עבודה שבה GBFS מוצא מסלול אופטימלי.

עוד 10 נקודות תקבל עבודה שבה A* מוצא מסלול אופטימלי.

חשוב מאוד : אם תשתמשו ביוריסטיקה אשר מריצה BFS ממצב מסויים כדי למצוא את המרחק שלו אל מצב היעד (או כל יוריסטיקה לא יעילה וערמומית בדומה לזאת), הציון שלכם יהיה 50 לכל היותר.

**<u>התראה</u>**

באינטרנט יש שפע של פתרונות תכנותיים של הבעייה הנ״ל. עם זאת העבודה הנדרשת מכם כאן היא של תיכנות במו ידיכם, ולא של חיפוש פתרון מן המוכן והעתקתו (תוך התאמות מזעריות.) פתרונות שיימצאו כמועתקים ממקור כלשהו (גם אם לאחר שעברו שינויים מסויימים) לא יקבלו כל ניקוד שהוא.

# מטלת מנחה (ממ"ן) 16

**הקורס:** **20551 – מבוא לבינה מלאכותית**

**חומר הלימוד למטלה:** פרקים 10, 12-13

**מספר השאלות:** 3 **משקל המטלה:** 2 נקודות

**סמסטר:** 2023ב **מועד אחרון להגשה:** 26.05.2023

(אב)

**קיימות שתי חלופות להגשת מטלות:**

- שליחת מטלות באמצעות מערכת המטלות המקוונת באתר הבית של הקורס
- שליחת מטלות באמצעות הדואר או הגשה ישירה למנחה במפגשי ההנחיה

**הסבר מפורט ב"נוהל הגשת מטלות מנחה"**

**שאלה 1** (30 נק')

נתונה בעיית התכנון הבאה:

נתון שולחן T, מנוף C, ו-n קוביות.

כל קוביה יכולה להיות על פני השולחן, בכף המנוף, או על גבי קוביה אחרת.

כף המנוף יכולה:

- להרים (pick) קוביה מהשולחן (אם אינה אוחזת בקוביה כלשהי)
- להניח (drop) קוביה על השולחן (אם היא אוחזת בקוביה זו)
- לערום (stack) קוביה x מעל קוביה y (אם היא אוחזת בקוביה x ואין קוביה מעל y)
- להסיר בעזרת כף המנוף (unstack) קוביה x מקוביה y (אם אינה אוחזת בקוביה כלשהי וקוביה x נמצאת מעל y).

בתחילה הקוביות 1..n מסודרות במגדל זו על גבי זו ונרצה להחליף את 2 הקוביות התחתונות (n ו-n-1).

ייצגו את הבעיה בשפת PDDL בדומה לדוגמאות שבסעיף 10.1 בספר הלימוד ובמדריך הלמידה.

**שאלה 2** (30 נק')

נתייחס לבעיית התכנון הבאה:

נתון רכב בירושלים (J) ורוצים להגיע בנסיעה בו לים המלח (DS).

כדי שניתן יהיה לנהוג ברכב, צריך להיות מפתח במפסק ההתנעה (switch).

נתונים 4 אופרטורים :

- Drive(J) – נהג (סע) לירושלים
- Drive(DS) – נהג (סע) לים המלח
- Insert(Key) – הכנס את המפתח למפסק ההתנעה
- Remove(Key) – הוצא את המפתח ממפסק ההתנעה

בנוסף לכך נתונים מספר פרדיקטים לתיאור מאפייני הבעיה :

- InPocket(Key) (המפתח בכיס)
- InIgnition(Key) (המפתח במפסק ההתנעה)
- At(Car, J) (הרכב בירושלים)
- At(Car, Ds) (הרכב בים המלח)

במצב ההתחלתי , המפתח בידנו ונרצה שיהיה לנו את המפתח גם בסיום התכנית (plan).

א. תארו את 4 האופרטורים ב-PDDL.

ב. בנו את גרף התכנון והגדירו את בעית התכנון המוחלשת (Relaxed Plan).
מהן ההערכות היוריסטיות של המצב ההתחלתי?

**שאלה 3 (40 נק':** 5 נק' ל-1א'; 5 נק' ל-1ב'; 5 נק' לסעיף 2; 25 נק' לסעיף 3**)**

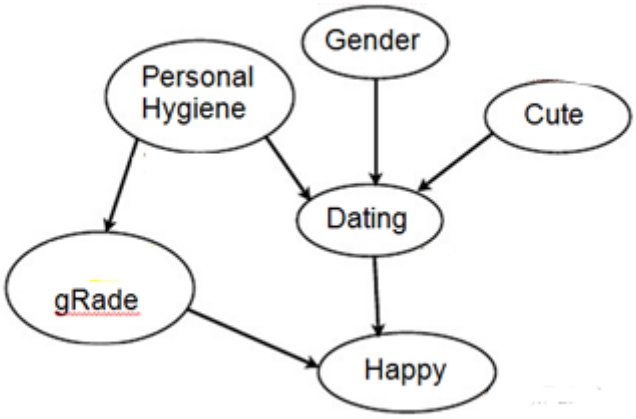
להלן נתונה רשת בייסיאנית המראה שביעות רצון (רמת האושר) של סטודנטים בקורס "מבוא לבינה מלאכותית".

המשתנים Cute(C), Dating(D), Happy(H), הם משתנים בוליאניים שערכיהם {*true, false*}.

המשתנה Gender(G) הוא משתנה בוליאני שערכיו הם {*male, female*}.

המשתנה Personal Hygiene(PH) הוא משתנה בוליאני שערכיו הם {*good, poor*}.

תחום הערכים של המשתנה gRade(R) הוא {A/B,C,F}.




1. השתמשו בהסתברויות המותנות הנובעות מהרשת הנתונה לעיל כדי לכתוב את הביטוי הנדרש עבור התפלגות ההסתברות בכל סעיף:

   א. P(G,C,D,P,R,H)

   ב. P(H|R,D,C)

   פשטו ככל שניתן את הביטוי שהתקבל.

2. מה תוכלו להסיק מסעיף 1-ב' לגבי התלות המותנית שבין Happy ו-Cute , בהינתן Dating ו-gRade?

3. להלן נתונות טבלאות ה-CPT עבור הרשת בייסיאנית שלעיל.

   חשבו בעזרתן את ההסתברויות הבאות:

   א. מהי ההסתברות שסטודנט שמח בהינתן שהוא ממין זכר, יוצא לדייט, ההיגיינה האישית שלו ירודה והציון שלו בקורס מבוא לבינה מלאכותית הוא A?

   P( H=true | D=true, G=male, P=poor, R=A/B)

   ב. מהי ההסתברות שקיים סטודנט כמתואר בסעיף א'?

   P( H=true, D=true, G=male, P=poor, R=A/B)

   ג. מהי ההסתברות שסטודנט נחמד בהינתן שהוא ממין זכר, שמח, יוצא לדייט וההיגיינה האישית שלו טובה P(C | D=true, G=male, P=good, H=true) ?

| P(PH) |
|---|
| 0.7 |

| P(G) |
|---|
| 0.8 |

| P(C) |
|---|
| 0.6 |

| C | G | PH | P(Dating) |
|---|---|---|---|
| *t* | *male* | *good* | 0.6 |
| *t* | *male* | *poor* | 0.3 |
| *t* | *female* | *good* | 0.9 |
| *t* | *female* | *poor* | 0.8 |
| *f* | *male* | *good* | 0.3 |
| *f* | *male* | *poor* | 0.1 |
| *f* | *female* | *good* | 0.7 |
| *f* | *female* | *poor* | 0.6 |

| PH | P(R)=A/B | P(R)=C | P(R)=F |
|---|---|---|---|
| *good* | 0.7 | 0.25 | 0.05 |
| *poor* | 0.6 | 0.3 | 0.1 |

| D | R | P(H) |
|---|---|---|
| *t* | A/B | 0.8 |
| *t* | C | 0.7 |
| *t* | F | 0.4 |
| *f* | A/B | 0.7 |
| *f* | C | 0.6 |
| *f* | F | 0.3 |

# מטלת מנחה (ממ"ן) 17

**הקורס:** **20551 – מבוא לבינה מלאכותית**

**חומר הלימוד למטלה:** פרקים 16,19

**מספר השאלות:** 2 **משקל המטלה:** 2 נקודות

**סמסטר:** 2023ב **מועד אחרון להגשה:** 9.6.2023

(אב)

**קיימות שתי חלופות להגשת מטלות:**

- שליחת מטלות באמצעות מערכת המטלות המקוונת באתר הבית של הקורס
- שליחת מטלות באמצעות הדואר או הגשה ישירה למנחה במפגשי ההנחיה

**הסבר מפורט ב"נוהל הגשת מטלות מנחה"**

**שאלה 1 (50 נק')**

פתרו את שאלה 10 מתוך השאלות של פרק 16 באתר הקורס.

**שאלה 2 (50 נק')**

כדי להחליט באיזה סרט לצפות, החלטתם לבנות עץ החלטה. אתם עושים זאת בהתבסס על נסיונכם המוצג בקבוצת האימון (training set) בטבלה שלהלן. לכל סרט מופיעים בטבלה 4 ערכים בוליאניים: קומדיה (האם הסרט הוא קומדיה), ליצנים (האם לליצנים יש תפקיד בסרט), אקדחים (האם בסרט משתמשים באקדחים) ובעמודה האחרונה האם אהבתם את הסרט או לא אהבתם.

דוגמאות האימון:

| סרט | קומדיה | ליצנים | אקדחים | אוהב |
|---|---|---|---|---|
| 1 | לא | כן | לא | לא |
| 2 | כן | לא | לא | כן |
| 3 | לא | לא | לא | לא |
| 4 | לא | לא | כן | כן |
| 5 | כן | לא | כן | לא |
| 6 | כן | לא | לא | כן |
| 7 | לא | כן | כן | לא |
| 8 | כן | כן | לא | כן |
| 9 | כן | כן | כן | לא |

א. איזו תכונה תיבחר עבור שורש עץ ההחלטה? נמקו ופרטו את החישובים.

במקרה של שוויון, העדיפו 'אקדחים' על פני 'ליצנים', 'ליצנים' על פני 'קומדיה'.
השלימו בניית עץ ההחלטה ופרטו את כל שלבי הבנייה.

# מטלת מנחה (ממ"ן) 18 - להרצה

**הקורס:** **20551 – מבוא לבינה מלאכותית**

**חומר הלימוד למטלה**: פרק 18 - למידה מדוגמאות

**מספר השאלות:** 1 **משקל המטלה:** 5 נקודות

**סמסטר:** 2023ב **מועד אחרון להגשה:** 23.6.2023

(אב)

**מטלת הרצה ניתן להגיש בדרך אחת ויחידה :**

- שליחת מטלות באמצעות מערכת המטלות המקוונת באתר הבית של הקורס

**הסבר מפורט ב"נוהל הגשת מטלות מנחה"**

**המטלה בעמודים הבאים**

# מימוש עצי החלטה

המטלה נלקחה מקורס מבוא לבינה מלאכותית CS 540 של אוניברסיטת ויסקונסין.

כל הקבצים הנדרשים לפתרון המטלה יועלו לאתר הקורס.

**אופן ההגשה:**

עליכם להעלות למערכת המטלות קובץ zip המכיל **<u>שני קבצים בדיוק</u>**:

את הקובץ DecisionTreeImpl.java (המכיל את כל הקוד שכתבתם) וכן קובץ README (בפורמט טקסטואלי כלשהו, כפי שנדרש במטלות ההרצה הקודמות).

In this problem you are to implement a program that builds a decision tree for categorical attributes and 2-class classification tasks. The programming part only requires building a tree from a training dataset and classifying instances of the test set with the learned decision tree.

You are required to implement four methods and one member for the class `DecisionTreeImpl`:

```
1. private DecTreeNode root;
2. DecisionTreeImpl ( DataSet train );
3. public String classify ( Instance instance );
4. public void rootInfoGain ( DataSet train );
5. public void printAccuracy (DataSet test)
```

`DecisionTreeImpl(DataSet train)` learns the decision tree from the training set, `train`.

`classify(Instance instance)` predicts the example, `instance`'s, label using the trained decision tree.

`rootInfoGain(DataSet train)` prints the information gain (one in each line) for all the attributes at the root based on the training set, `train`. The root of your tree should be stored in the member root that has been declared for you. The next sections describe other aspects in detail.

`printAccuracy(DataSet test)` prints the classification accuracy for the instances in the test set, `test`, using the learned decision tree.

**Dataset**

Our datasets come from a risk loan dataset, which is being used to predict the risk quality of a loan application. There are altogether 1000 examples (also called instances) and we chose 10 categorical attributes to use for this assignment. Each instance is classified as good (class G) or bad (class B), so this is a 2-class classification problem. You can assume other datasets used for testing will also be 2-class classification tasks.

The tables of attributes and their possible values are shown in the table below:

| | |
|---|---|
| A1: Checking status | x(no checking)<br>n(x<0, negative)<br>b(0<=x<200, bad)<br>g(200<=x, good) |
| A2: Saving status | n(no known savings)<br>b(x<100)<br>m(100<=x<500)<br>g(500<=x<=1000)<br>w(1000<=x) |
| A3: Credit history | a(all paid)<br>c(critical/other existing credit)<br>d(delayed previously)<br>e(existing paid)<br>n(no credits) |
| A4: Housing | r(rent)<br>o(own)<br>f(free) |
| A5: Job | h(high qualified/self-employed/management)<br>s(skilled)<br>n(unemployed)<br>u(unskilled) |
| A6: Property magnitude | c(car)<br>l(life insurance)<br>r(real estate)<br>n(no known property) |
| A7: Number of dependents | 1, 2 |
| A8: Number of existing credits | 1, 2, 3, 4 |
| A9: Own telephones or not | y(yes), n(no) |
| A10: Foreign workers or not | y(yes), n(no) |

In each file, there will be a header that gives information about the dataset; an example header and the first example in the dataset is shown below. First, there will be several lines starting with `//` that provide some description and comments about the dataset. Next, the line starting with %% will list all the class labels. Each line starting with ## will give the name of one attribute and all its possible values. We have written the dataset loading part for you according to this header, so do NOT change it. Following the header are the examples in the dataset, one example per line. The first example is shown below and corresponds to the feature vector (A1=x, A2=n, A3=e, A4=r, A5=h, A6=l, A7=1, A8=1, A9=y, A10=y) and its class is G.

```
// Description of the data set
%%,G,B
##,A1,x,n,b,g
##,A2,n,b,m,g,w
##,A3,a,c,d,e,n
##,A4,r,o,f
##,A5,h,s,n,u
##,A6,c,l,r,n
##,A7,1,2
##,A8,1,2,3,4
##,A9,y,n
##,A10,y,n
x,n,e,r,h,l,1,1,y,y,G
…
```

## Implementation Details

## Predefined Data Types

We have defined four data types to assist your coding, called `Instance`, `DataSet`, `DecTreeNode` and `DecisionTreeImpl`. Their data members and methods are all commented, so it should not be hard to understand their meaning and usage.

## Building the Tree

In the `DecisionTreeImpl(DataSet train)` method, you are required to build the decision tree using the training data. Refer to the pseudocode in Figure 18.5 on page 702 of the textbook to see what your code should do.
To finish this part, you may need to write a recursive function corresponding to the `DecisionTreeLearning` function in the textbook.

## Classification

`public String classify(Instance instance)` takes an example (called an instance) as its input and computes the classification output (as a string) of the previously-built decision tree. You do not need to worry about printing. That part is already handled in the provided code.

## Printing and Information Gain at the Root

The only printing you need to do is in the method

`public void rootInfoGain(DataSet train)` and

`public void printAccuracy(DataSet test)`.

In `rootInfoGain`, for each attribute print the output one line at a time: first the name of the attribute and then the information gain achieved by selecting that attribute at the root. The output order of the attributes and associated information gain values must be the *same* as the order that the attributes appear in the training set's header. Print your results with 5 decimal places using `System.out.format("%.5f\n", arg)`

In `printAccuracy`, you should only print out the accuracy with 5 decimal places.

## Testing

We will test your program using several training and testing datasets using the command line format:

```
java HW3 <modeFlag> <trainFile> <testFile>
```

where `trainFile` and `testFile` are the names of the training and testing datasets, respectively. `modeFlag` is an integer from 0 to 3, controlling what the program will output. Only `modeFlag` = {0, 1, 2, 3} are required to be implemented for this assignment.
The requirements for each value of `modeFlag`= {0, 1, 2, 3} are described as following:

0: Print the information gain for each attribute at the root node based on the training set

1: Create a decision tree from the training set and print the tree

2: Create a decision tree from the training set and print the classification for each example in the test set

3: Create a decision tree from the training set and print the accuracy of the classification for the test set

To facilitate debugging, we have provided three input files called `example1.txt, example2.txt and example3.txt`. It is highly recommended that you write for yourself a small application that produces random input files in the expected format. Actually make them not so random, so that you'll know what to expect when you build your decision trees.

So, here is an example command:

```
java HW3 0 train1.txt test1.txt
```

You are *NOT* responsible for any file input or console output other than public void `rootInfoGain (DataSet train)` and `printAccuracy(DataSet test)`. We have written the class HW3 for you, which will load the data and pass it to the method you are implementing.

The format of `rootInfoGain (modeFlag == 0)` should look like

```
A1 0.11111
A2 0.11111
…
A10 0.11111
```

The format of `printAccuracy (modeFlag == 3)` should look like

```
0.12345
```

As part of our testing process, we will unzip the file you submit, call javac`*.java` to compile your code, and then call the main method HW3 with parameters of our choosing.