**מטלת מנחה (ממ"ן) 11 - להרצה**

**הקורס: 20551 – מבוא לבינה מלאכותית**

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

|  |  |  |
| --- | --- | --- |
| **מספר השאלות:**  8 | **משקל המטלה:**  6 | |
| **סמסטר:** 2021א | **מועד אחרון להגשה:** 3.11.2020 | |
| (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"** |

מטלות הריצה בקורס ייכתבו בשפת [**Python(2.7)**](https://www.python.org/), שפת תכנות מונחה עצמים.

אנו לא מניחים כלל שיש לכם נסיון בתכנות בשפה זו, אך מצפים שתלמדו את הבסיס שלה די מהר, (במיוחד לאור הכרותכם ונסיונכם עם שפת Java).

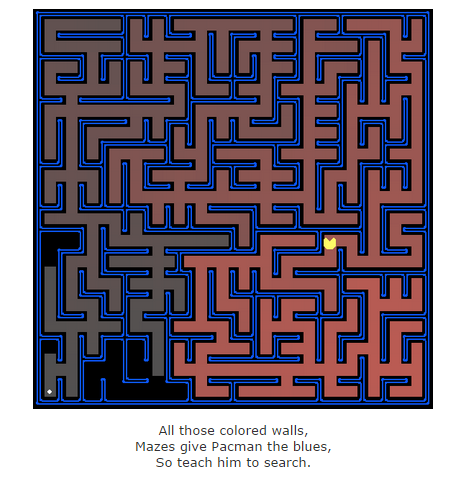
מצורף [כאן](http://ai.berkeley.edu/tutorial.html) קישור למדריך בסיסי ב- Python(2.7) ובו משולבים תרגילים שעליכם לפתור כדי להכיר השפה. אין צורך להגישם. הקובץ שבקישור מתייחס גם לעבודה בסביבת Unix וניתן להתעלם מההתייחסויות ל-Unix.

שתי מטלות הריצה הראשונות פותחו באוניברסיטת ברקלי על-ידי John DeNero ו-Dan Klein.

מדריך לPython בעברית ניתן למצוא [כאן](http://vlib.eitan.ac.il/python/).

מדריך מפורט ניתן למצוא [כאן](https://docs.python.org/2/tutorial/).

**חיפוש ב-Pac-Man**



**הקדמה**

במטלה זו, סוכן פקמן שלכם ימצא מסלולים בעולם המבוך שלו, הן כדי להגיע למיקום מסויים והן כדי ללקט מזון בצורה יעילה. עליכם לכתוב אלגוריתמי חיפוש כלליים ולממשם לתרחישים של פקמן.

הקוד עבור מטלה זו מורכב ממספר קבצי Python. לצורך ביצוע המטלה יהיה עליכם לקרוא ולהבין את חלקם ומהאחרים תוכלו להתעלם. את כל הקוד והקבצים הנלווים ניתן להוריד [כקובץ zip](https://inst.eecs.berkeley.edu/~cs188/sp10/projects/search/search.zip).

הקבצים שתערכו ולתוכם תוסיפו את הקוד שלכם:

כל אלגוריתמי החיפוש שכם יימצאו שם [search.py](https://inst.eecs.berkeley.edu/~cs188/sp10/projects/search/docs/search.html)

כל סוכני החיפוש שלכם יימצאו שם [searchAgents.py](https://inst.eecs.berkeley.edu/~cs188/sp10/projects/search/docs/searchAgents.html)

קבצים שכדאי לקרוא:

הקובץ העיקרי שמריץ משחקי פקמן. הקובץ מתאר את הטיפוס GameState [pacman.py](https://inst.eecs.berkeley.edu/~cs188/sp10/projects/search/docs/pacman.html)

שתשתמשו בו במטלה זו

הלוגיקה שמאחורי אופן פעולת עולם הפקמן. קובץ זה מתאר [game.py](https://inst.eecs.berkeley.edu/~cs188/sp10/projects/search/docs/game.html)

מספר טיפוסי עזר כגון . AgentState , Agent, Direction, Grid

מבני נתונים שימושיים למימוש אלגוריתמי חיפוש [util.py](https://inst.eecs.berkeley.edu/~cs188/sp10/projects/search/docs/util.html)

קבצים שאין צורך להתעמק בקוד שלהם:

[GraphicsDisplay.py](https://inst.eecs.berkeley.edu/~cs188/sp10/projects/search/docs/graphicsDisplay.html) גרפיקה עבור פקמן

[GraphicsUtils.py](https://inst.eecs.berkeley.edu/~cs188/sp10/projects/search/docs/graphicsUtils.html) תמיכה לגרפיקה של פקמן

[TextDisplay.py](https://inst.eecs.berkeley.edu/~cs188/sp10/projects/search/docs/textDisplay.html) גרפיקת אסקי לפקמן

[GhostAgents.py](https://inst.eecs.berkeley.edu/~cs188/sp10/projects/search/docs/ghostAgents.html) סוכנים לבקרה על רוחות רפאים

[KeyboardAgents.py](https://inst.eecs.berkeley.edu/~cs188/sp10/projects/search/docs/keyboardAgents.html) ממשקי מקלדת לבקרה על פקמן

[Layout.py](https://inst.eecs.berkeley.edu/~cs188/sp10/projects/search/docs/layout.html) ואחסון תוכנם layout קוד לקריאת קבצי

עליכם להגיש

יש למלא חלקים חסרים בקוד [search.py](https://inst.eecs.berkeley.edu/~cs188/sp10/projects/search/docs/search.html) ו-[searchAgents.py](https://inst.eecs.berkeley.edu/~cs188/sp10/projects/search/docs/searchAgents.html) במהלך ביצוע המטלה. עליכם לשלוח למנחה את שני הקבצים הללו (בלבד) וכן את README.txt (שם, ת.ז ותיעוד) כקובץ zip. כלומר יש לשלוח בדיוק קובץ אחד.

הנכם מתבקשים לא לשנות שמות של פונקציות או מחלקות הנתונות בקוד.

**ברוכים הבאים לפקמן**

לאחר שהורדתם את הקוד ([search.zip](https://inst.eecs.berkeley.edu/~cs188/sp10/projects/search/search.zip)), עשיתם לו unzip, תוכלו לשחק משחק פקמן. לצורך כך הקלידו בשורת הפקודה (command line):

python pacman.py

*שימו לב*: אם קיבלתם הודעת שגיאה לגבי python-tk, התקינו python-tk או ראו הוראות מפורטות יותר [בעמוד זה](https://docs.python.org/3.4/library/tkinter.html).

פקמן חי בעולם כחול בוהק של מסדרונות פתלתלים וממתקים עגולים טעימים. ניווט בעולם זה ביעילות תיהיה המשימה הראשונה של פקמן.

הסוכן הפשוט ביותר בקובץ [searchAgents.py](https://inst.eecs.berkeley.edu/~cs188/sp10/projects/search/docs/searchAgents.html) נקרא GoWestAgent, שתמיד זז מערבה (סוכן של תגובה פשוטה). סוכן זה יכול לעתים לנצח:

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

אבל, העניינים מסתבכים עבור סוכן זה כאשר עליו להסתובב:

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

אם פקמן נתקע, תוכלו לצאת מהמשחק על ידי הקשת CTRL-c.

במהרה הסוכן שלכם יפתור לא רק tinyMaze אלא כל מבוך שתרצו. שימו לב ש- [pacman.py](https://inst.eecs.berkeley.edu/~cs188/sp10/projects/search/docs/pacman.html) תומך במספר אופציות. כל אחת מהן ניתנת לביטוי באריכות (למשל, --layout) או בקצרה (למשל, -1 ). תוכלו לראות את רשימת כל האפשרויות וערכי ברירת המחדל שלהן על ידי:

python pacman.py -h

כמו כן, כל הפקודות המופיעות במטלה זו, מופיעות גם ב-[commands.txt](https://inst.eecs.berkeley.edu/~cs188/sp10/projects/search/commands.txt), לצורך נוחיות העתקתן.

**שימוש באלגוריתמי חיפוש למציאת נקודת מזון קבועה**

בקובץ [searchAgents.py](https://inst.eecs.berkeley.edu/~cs188/sp10/projects/search/docs/searchAgents.html), תמצאו searchAgent ממומש במלואו, המתכנן מסלול דרך עולם פקמן ולאחר מכן מבצע מסלול זה צעד אחר צעד. אלגוריתמי החיפוש עבור התכנית (plan) אינם ממומשים. משימה זו מוטלת עליכם. במהלך פתרון השאלות שיופיעו להלן, תצטרכו כנראה להתייחס לרשימת תיאור סוגי האובייקטים שבקוד המופיעה בסוף מטלה זו.

תחילה, בידקו שה- searchAgentפועל היטב על ידי כך שתריצו:

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

הפקודה שלעיל מורה ל- searchAgentלהשתמש ב-tinyMazeSearch (הממומש בקובץ [search.py](https://inst.eecs.berkeley.edu/~cs188/sp10/projects/search/docs/search.html)) כאלגוריתם החיפוש שלו. פקמן אמור לנווט במבוך בהצלחה.

כעת ניתן לכתוב פונקציות חיפוש כדי לעזור לפקמן לתכנן מסלולים. הפסאודו-קוד לאלגוריתמי החיפוש נמצא בספר הלימוד. זכרו כי צומת במרחב החיפוש צריך להכיל בנוסף למצב עצמו, את כל המידע הדרוש לבנייה מחדש של המסלול (plan) המוביל למצב זה.

*הערה חשובה*: כל פונקציות החיפוש שלכם צריכות להחזיר רשימת *פעולות* שתוביל את הסוכן מהמצב ההתחלתי אל מצב המטרה. פעולות אלה צריכות להיות (כולן) מהלכים חוקיים (כיוונים מותרים, לא להיכנס בתוך קירות).

*רמזים*:

1. האלגוריתמים דומים אחד לשני. אלגוריתמים עבור DFS, BFS, A\*,Uniform-Cost search נבדלים זה מזה רק בפרטי אופן ניהול החזית. כלומר, לאחר שתכתבו את DFS כנדרש, יהיה לכם קל מאד לכתוב את יתר האלגוריתמים. אכן, אחד המימושים האפשריים (אינו חובה) דורש יצירת מתודה לחיפוש כללי שמקבלת את מבנה הנתונים לאחזקת המצבים הממתינים להיסרק.

2. בדקו את טיפוסי Queue, Stack ו-PriorityQueue הנתונים לכם בקובץ [util.py](https://inst.eecs.berkeley.edu/~cs188/sp10/projects/search/docs/util.html)!

**שאלה 1 (10%)**

ממשו את אלגוריתם חיפוש לעומק **(**DFS) בפונקציה depthFirstSearch אשר בקובץ [search.py](https://inst.eecs.berkeley.edu/~cs188/sp10/projects/search/docs/search.html)**.** כדי שהאלגוריתם שלכם יהיה שלם, כתבו את גירסת חיפוש-גרף של חיפוש לעומק, המונעת פיתוח של מצבים שנסרקו כבר. (ראו סעיף 3.3 בספר הלימוד).

הקוד שתכתבו אמור למצוא פתרון עבור:

python pacman.py -l tinyMaze -p SearchAgent

python pacman.py -l mediumMaze -p SearchAgent

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

הלוח של פקמן יראה את המצבים שנסרקו ואת הסדר על פיו נסרקו (אדום בהיר יותר פירושו שנסרקו מוקדם יותר). האם סדר הסריקה תואם לסדר שהייתם מצפים?

*רמז*: אם תשתמשו ב-Stack כבמבנה הנתונים שלכם, הפתרון שימצא אלגוריתם חיפוש לעומק שכתבתם עבור mediumMaze יהיה באורך 130 (בהנחה שהוספתם עוקבים לחזית על פי הסדר הנתון על ידי getSuccessors ; אתם אמורים לקבל 244 אם הוספתם אותם בסדר ההפוך).

האם זה פתרון זול ביותר? נמקו.

**שאלה 2(10%)**

ממשו אלגוריתם חיפוש לרוחב (BFS) בפונקציית breadthFirstSearch שבקובץ [search.py](https://inst.eecs.berkeley.edu/~cs188/sp10/projects/search/docs/search.html)**.** שוב, עליכם לכתוב אלגוריתם חיפוש-גרף המונע פיתוח של מצבים שנסרקו כבר. בדקו את הקוד שלכם באותה הדרך שבדקתם 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 אלגוריתם חיפוש לרוחב מוצא פתרון זול ביותר? נמקו.

*רמז*: אם נראה לכם שפקמן זז לאט מדי, נסו את האופציה --frameTime 0.

שימו לב: אם כתבתם את קוד החיפוש באופן כללי, הוא אמור לעבוד באותה מידה גם עבור בעיית

8-puzzle, ללא כל שינוי.

python eightpuzzle.py

**שינוי פונקצית המחיר**

BFS ימצא את המסלול למטרה בעל מספר פעולות מינימלי, אך יתכן שנרצה למצוא מסלולים שהם "טובים יותר" במובנים אחרים.

נתייחס ל-[mediumDottedMaze](https://inst.eecs.berkeley.edu/~cs188/sp10/projects/search/layouts/mediumDottedMaze.lay) ול-[mediumScaryMaze](https://inst.eecs.berkeley.edu/~cs188/sp10/projects/search/layouts/mediumScaryMaze.lay). על ידי שינוי פונקצית המחיר, אנו יכולים לעודד את פקמן למצוא מסלולים אחרים.

למשל, אנו יכולים לתת מחיר גבוה לצעדים באיזורים בהם יש הרבה רוחות רפאים או לתת מחיר

נמוך לצעדים באיזורים עתירי מזון וסוכן פקמן רציונלי אמור להתאים את התנהגותו לשינויים אלה.

**שאלה 3 (10%)**

ממשו אלגוריתם לחיפוש-גרף מונחה-מחיר בפונקציה uniformCostSearch אשר בקובץ [search.py](https://inst.eecs.berkeley.edu/~cs188/sp10/projects/search/docs/search.html). כדאי לעבור על [util.py](https://inst.eecs.berkeley.edu/~cs188/sp10/projects/search/docs/util.html) כדי לראות מבני נתונים שיכולים להיות לכם לעזר במימוש האלגוריתם. כל הסוכנים שלהלן הינם סוכני חיפוש מונחה מחיר וההבדל ביניהם הוא בפונקציית המחיר בה כל אחד משתמש (הסוכנים ופונקציות המחיר נתונים לכם).

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

python pacman.py -l mediumDottedMaze -p StayEastSearchAgent

python pacman.py -l mediumScaryMaze -p StayWestSearchAgent

*שימו לב*: אתם אמורים לקבל מחירי מסלול מאד נמוכים ומאד גבוהים עבור StayEastSearchAgent ו- StayWestSearchAgent בהתאמה, עקב פונקציות המחיר האקספוננציאליות שלהם (פרטים בקובץ [searchAgents.py](https://inst.eecs.berkeley.edu/~cs188/sp10/projects/search/docs/searchAgents.html)).

**חיפוש \*A**

**שאלה 4 (15%)**

ממשו אלגוריתם חיפוש-גרף \*A בפונקציה aStarSearch (הריקה) אשר בקובץ [search.py](https://inst.eecs.berkeley.edu/~cs188/sp10/projects/search/docs/search.html). **\***A מקבל כפרמטר פונקציה יוריסטית. יוריסטיקה מקבלת שני פרמטרים: מצב בבעיית חיפוש (הפרמטר הראשי) והבעיה עצמה. דוגמה טריוויאלית היא הפונקציה היוריסטית nullHeuristic שבקובץ [search.py](https://inst.eecs.berkeley.edu/~cs188/sp10/projects/search/docs/search.html). אתם יכולים לבדוק את מימוש **\***A על הבעיה המקורית של מציאת מסלול למיקום קבוע במבוך תוך שימוש ביוריסטיקת מרחק מנהטן (הממומשת כבר כ-manhattanHeuristic בקובץ [searchAgents.py](https://inst.eecs.berkeley.edu/~cs188/sp10/projects/search/docs/searchAgents.html)).

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

heuristic=manhattanHeuristic

תוכלו לראות ש\*A מוצא את הפתרון האופטימלי קצת מהר יותר מחיפוש מונחה-מחיר (בסביבות 549 לעומת 629 צמתי חיפוש מפותחים במימוש שלנו אבל כשיש עדיפויות זהות, הבחירות שהאלגוריתם יבצע עלולות לשנות מעט את מספרם של המצבים.

מה קורה ב-openMaze לאסטרטגיות החיפוש השונות?

**מציאת כל הפינות**

היתרון האמיתי של A\* יבוא לידי ביטוי בבעית חיפוש מאתגרת יותר. ננסח כעת בעיה חדשה ונתכנן יוריסטיקה עבורה.

נתון מבוך עם ארבע פינות ובכל אחת מהן יש נקודה. בבעית החיפוש החדשה יש למצוא את המסלול הקצר ביותר במבוך העובר דרך ארבע הפינות. (בין אם במבוך יש שם מזון או לא).שימו לב לכך שעבור מבוכים מסויימים כגון [tiny Corners](https://inst.eecs.berkeley.edu/~cs188/sp10/projects/search/layouts/tinyCorners.lay), המסלול הקצר ביותר לא תמיד עובר תחילה דרך נקודת המזון הקרובה ביותר! *רמז*: אורך המסלול הקצר ביותר דרך tinyCorners הוא 28 צעדים.

**שאלה 5 (10%)**

ממשו את בעיית החיפוש CornersProblem בקובץ [searchAgents.py](https://inst.eecs.berkeley.edu/~cs188/sp10/projects/search/docs/searchAgents.html). עליכם לבחור יצוג למצבים אשר יקודד את כל המידע הדרוש כדי לדעת האם נסרקו כל ארבע הפינות. כעת סוכן החיפוש שלכם יפתור:

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 בשאלה זו, עליכם להגדיר את ייצוג המצב כך שלא יקודד מידע לא רלבנטי (כגון מיקום רוחות הרפאים, היכן יש מזון נוסף וכד'). במיוחד, אל תשתמשו ב-GameState של פקמן כבמצב של מרחב החיפוש. אחרת, הקוד שלכם יהיה איטי מאד מאד וגם שגוי).

*רמז*: החלקים היחידים במצב במשחק שאליהם עליכם להתייחס במימושכם הם המיקום ההתחלתי של פקמן ומיקומן של ארבע הפינות.

המימוש של breadthFirstSearch מפתח פחות מ-2000 צמתים ב- [medium Corners](https://inst.eecs.berkeley.edu/~cs188/sp10/projects/search/layouts/mediumCorners.lay). ואולם, יוריסטיקה (שמשתמשים בה ב-A\*) יכולה להפחית את החיפוש הנדרש.

**שאלה 6 (15%)**

ממשו יוריסטיקה עבור בעיית החיפוש CornersProblemב-CornersHeuristic.

אופן מתן הציון לשאלה זו: לא ינתן ניקוד עבור פונקציה יוריסטית שאינה קבילה**.** תינתן נקודה אחת לכל פונקציה יוריסטית קבילה. נקודה אחת אם יפותחו פחות מ-1200 צמתים. נקודה אחת אם יפותחו פחות מ-900 צמתים. ניקוד מלא אם יפותחו פחות מ-800 צמתים.

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

*רמז*: זכרו, פונקציות יוריסטיות מחזירות רק מספרים, אשר צריכים להוות חסמים תחתונים על מחיר המסלול (האמיתי) הקצר ביותר למטרה הקרובה ביותר, כדי שהיוריסטיקות תהיינה קבילות.

*שימו לב*:AStarCornerAgent הוא קיצור עבור

-p SearchAgent -a fn=aStarSearch,prob= Cornersproblem, heuristic=CornersHeuristic

**"אכילת" כל הנקודות**

נפתור כעת בעית חיפוש קשה: "אכילת" כל המזון של פקמן במספר צעדים קטן ככל האפשר. לצורך כך נזדקק להגדרה חדשה עבור בעית החיפוש המבטאת את בעית "אכילת" כל המזון.

FoodSearchProblemבקובץ [searchAgents.py](https://inst.eecs.berkeley.edu/~cs188/sp10/projects/search/docs/searchAgents.html) (הממומשת עבורכם).נגדיר פתרון כמסלול האוסף את כל המזון שבעולם הפקמן. לצורך ביצוע מטלה זו, הפתרונות אינם מביאים בחשבון רוחות רפאים כלשהן או חטיפי (טבליות) אנרגיה ; הפתרונות תלויים רק במיקום הקירות, מזון רגיל ובפקמן. (כמובן שרוחות רפאים יכולות להרוס את ביצוע הפתרון! בכך נטפל במטלת הריצה הבאה). אם כתבתם נכון את מתודות החיפוש הכלליות, A\* ללא יוריסטיקה (השקול לחיפוש מונחה-מחיר) יוכל בקלות למצוא פתרון אופטימלי ל-[test Search](https://inst.eecs.berkeley.edu/~cs188/sp10/projects/search/layouts/testSearch.lay) ללא שינוי בקוד מצדכם (מחיר כולל של 7).

python pacman.py -l testSearch -p AStarFoodSearchAgent

*שימו לב*:AStarFoodSearchAgent הוא קיצור עבור

-p SearchAgent -a fn=aStar,prob=FoodSearchProblem, heuristic=FoodHeuristic

אתם אמורים לגלות שחיפוש מונחה מחיר מתחיל להיות איטי אפילו ה-[tiny Search](https://inst.eecs.berkeley.edu/~cs188/sp10/projects/search/layouts/tinySearch.lay) הפשוט. לצורך השוואה, המימוש שלנו מוצא מסלול באורך 27 ב-2.5 שניות, לאחר פיתוח של 4902 צמתים במרחב החיפוש.

**שאלה 7 (20% + בונוס 5%\*)**

השלימו את foodHeuristic בקובץ[searchAgents.py](https://inst.eecs.berkeley.edu/~cs188/sp10/projects/search/docs/searchAgents.html)עם פונקציה יוריסטית עקבית עבור FoodSearchProblem. נסו את הסוכן שלכם על לוח trickySearch:

python pacman.py -l trickySearch -p AStarFoodSearchAgent

סוכן החיפוש המונחה-מחיר שלנו מוצא את הפתרון האופטימלי ב-13 שניות לערך, תוך שהואסורק 16,000 צמתים.אם היוריסטיקה שלכם קבילה, תקבלו את הניקוד הבא, התלוי במספר הצמתים שפותחו בעזרתה.

|  |  |
| --- | --- |
| מספר צמתים קטן מ- | ניקוד |
| 15000 | 5% |
| 12000 | 10% |
| 9000 | 15% |
| 7000 | 5% - בונוס**\*** |

*שימו לב*: אם היוריסטיקה שלכם אינה קבילה, לא תקבלו ניקוד כלל. חשבו על ענין הקבילות בזהירות, שכן יוריסטיקה לא קבילה יכולה להצליח ליצור חיפושים מהירים ואף מסלולים אופטימליים. האם תוכלו לפתור [medium Search](https://inst.eecs.berkeley.edu/~cs188/sp10/projects/search/layouts/mediumSearch.lay) בזמן קצר? אם כן, הדבר מרשים ביותר או... שהיוריסטיקה שלכם אינה קבילה.

**\***הציון הכולל במטלה לא יעלה על 100.

קבילות לעומת עקביות:

טכנית, קבילות אינה מספקת כדי להבטיח נכונות בחיפוש-גרף. יש צורך בקיום תנאי חזק יותר, של עקביות. כדי שיוריסטיקה תהיה עקבית, צריך להתקיים שאם מחירה של פעולה הוא c, אזי ביצוע פעולה זו יכול לגרום רק לירידה ביוריסטיקה של c לכל היותר. אם היוריסטיקה שלכם אינה רק קבילה אלא גם עקבית, תקבלו 5% נוספים על תשובתכם לשאלה 7.

כמעט תמיד, יוריסטיקות קבילות הן גם עקביות, במיוחד אם הן נגזרו מבעיות מוחלשות. לכן קל יותר, כפי הנראה, להתחיל במחשבה על יוריסטיקות קבילות. לאחר שתהיה לכם יוריסטיקה קבילה שעובדת היטב, תוכלו לבדוק האם היא אכן גם עקבית. לעתים ניתן לגלות חוסר עקביות על ידי בדיקה האם ערכי ה-f של הפתרונות המוחזרים שלכם אינם יורדים. יתר על כן, אם חיפוש מונחה מחיר ו-A\* מחזירים מסלולים באורך שונה, היוריסטיקה שלכם אינה עקבית.

**חיפוש תת-מיטבי (Suboptimal Search)**

לעתים, גם עם A\* ופונקציה יוריסטית טובה, מציאת מסלול אופטימלי דרך כל הנקודות הינה משימה קשה. במקרים כאלה, נרצה עדיין למצוא מסלול טוב, בזמן קצר.

עליכם לכתוב כעת קוד לסוכן פקמן אשר תמיד אוכל באופן חמדני את הנקודה הקרובה ביותר. ClosestDotSearchAgent ממומש עבורכם בקובץ [searchAgents.py](https://inst.eecs.berkeley.edu/~cs188/sp10/projects/search/docs/searchAgents.html)**,** אך חסרה לו הפונקציה העיקרית המוצאתמסלול לנקודה הקרובה ביותר.

**שאלה 8 (10%)**

ממשו את הפונקציה findPathToClosestDot שבקובץ[searchAgents.py](https://inst.eecs.berkeley.edu/~cs188/sp10/projects/search/docs/searchAgents.html)**.** הסוכן שלנו פותר מבוך זה (באופן תת-מיטבי) בפחות משניה אחת ובמחיר מסלול של 350:

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

*רמז*: הדרך המהירה ביותר להשלים את findPathToClosestDot היא להשלים את החסר ב- AnyFoodSearchProblem (שחסר בו מבחן המטרה). לאחר מכן, פתרו בעיה זו עם פונקציית חיפוש מתאימה. הפתרון אמור להיות מאד קצר!

ה-ClosestDotSearchAgent שלכם לא תמיד ימצא את המסלול האפשרי הקצר ביותר דרך המבוך. אך אם תנסו, תוכלו לשפרו.

לנוחיותכם מרוכזים ברשימה שלהלן **סוגי האובייקטים העיקריים המופיעים בקוד הבסיסי הקשורים לבעיות חיפוש:**

