מבוא לבינה מלאכותית (570-89)

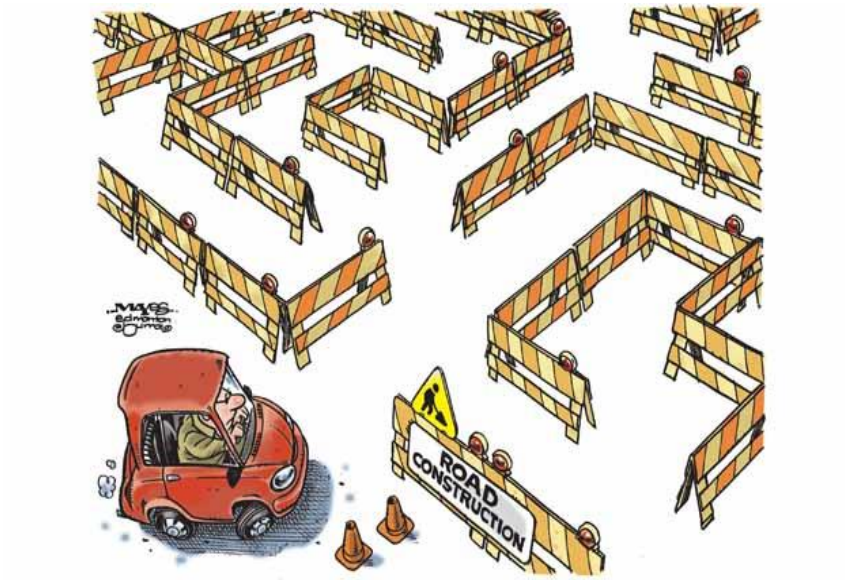
תרגיל בית 1: בעיות חיפוש

מטרות המשימה

- התנסות בייצוג בעיות אקטואליות ומציאותיות כמרחבי מצבים.
- התנסות באלגוריתמי חיפוש.
- תכנון וריאציה אלגוריתמית בהתאם לבעיה.

<u>הערות:</u>

- תאריך הגשה: 24.11.22
- המטלה להגשה **<u>ביחידים בלבד!</u>**



תיאור הבעיה

תוכנת MyWay מוצאת את המסלול בעל זמן הנסיעה הצפוי הקצר ביותר, כאשר זמן הנסיעה בכל קטע משוערך על ידי המהירות הממוצעת באותו קטע.

הנכם יזמים שהקימו את חברת BetterWay המנסה ליצר אלטרנטיבה לאפליקציית MyWay הותיקה.

התוכנה תעבוד עם קובץ נתונים המייצג את רשת הכבישים של ישראל. אנו ביצענו הורדה של מפת ישראל מאתר www.openstreetmap.org והמרה לפורמט שיהיה נוח לעבודה עם python ושיכלול רק את המידע הרלוונטי מתוך שלל הנתונים המקוריים, זהו הקובץ israel.csv.

חלק 1 – מבוא והקדמה *(10 נקודות)*

1. *שאלה יבשה:* פיתחו את הקבצים *db/israel.csv* ותארו את המבנה שלהם במדויק: מה מייצגת כל שורה ומה הפרמטרים בה. לצורך כך, עיינו בקוד של השגרה `load_map_from_csv` בקובץ הקוד המסופק *ways/graph.py*.
2. *שאלה רטובה:* מלאו את תוכן שגרת העזר `map_statistics` (הנמצאת בקובץ state.py) המחשבת פרמטרים המאפיינים את המפה הנתונה.

יש לאפשר להריץ את הקוד עבור הסעיף דרך שורת הפקודה `$ python stats.py` הפלט צריך להיות תוצאת הדפסה פשוטה של המילון אותו מחזירה הפונקציה.

3. צרו 100 בעיות חיפוש אקראיות הבעיות צריכות להכיל את צומת ההתחלה וצומת הסיום. כתבו את צומת ההתחלה וצומת הסיום בקובץ problems.csv בפורמט הבא

s1, t1

s2, t2

...

s20, t20

<u>הערה חשובה: חלקים מהמפה אינם קשירים, לכן על אחריותכם לסנן בעיות חיפוש שאינן פתירות.</u>

חלק 2 – UCS *(20 נקודות)*

בחלק זה נממש את אלגוריתם UCS כפי שנלמד בכיתה.

לפני מענה על השאלות בחלק הזה מומלץ לעבור על כל הקוד המצורף עם התרגיל.

4. כתבו בפייתון פונקציה find_ucs_rout (הפו' מופיעה בקובץ main.py) המקבלת שני פרמטרים - צומת התחלה וצומת סיום. פונקצית מחיר (המוגדרת להיות זמן הנסיעה בין צמתים) ומחזירה את המסלול המהיר ביותר מנקודת המוצא ליעד בעזרת אלגוריתם UCS.

במימוש פו' המחיר השתמשו במרחק לצומת מהקובץ Israel.csv.

5. השתמשו באותן 100בעיות חיפוש שיצרתם בסעיף 3 והריצו עליהן חיפוש UCS. עבור כל אחת מהבעיות, פלטו לקובץ *results/UCSRuns.txt* את מסלול הנסיעה (כולל קצוות) וזמניהם. (כל בעית חיפוש בשורה כאשר רשימת הצמתים תחילה ולאחריהם ' - ' וזמן הנסיעה.)

הקובץ main.py מרכז את הממשק אל שורת הפקודה; יש לכתוב בו מעט ככל הניתן.
יש לאפשר להריץ את הקוד עבור הסעיף דרך שורת הפקודה.
למשל, אם נקודת המוצא היא 30 ונקודת היעד 55:
`$ python main.py ucs  30 55`
על הפלט להיות רשימה פשוטה של מספרי צמתים, כולל קצוות:
`30 21 44 73 55`

חלק 3 - A* (45 נקודות)

בחלק זה נממש את A* כמו שנלמד בכיתה.

6. כתבו בפייתון פונקציה find_astar_route המקבלת שני פרמטרים - צומת התחלה וצומת סיום. כמו בסעיף הקודם, השתמשו בפונקצית המחיר ~~הממומשת בקוד ונמצאת תחת tools.compute_distance~~ משאלה 4 ופונקציה יוריסטית אותה תממשו בשלב הבא הנמצאת בקובץ main.py בשם huristic_function. על הפונקציה להחזיר את המסלול המהיר ביותר מנקודת המוצא ליעד בעזרת אלגוריתם A* .
7. חשבו על פונקציה יוריסטית קבילה מתאימה וממשו אותה בפו' huristic_function המקבלת ערכי latitude ו- longitude מתאימים ומחזירה את הערכת המחיר למצב המטרה.
שימו לב – בקובץ info.py ישנה רשימה המכילה את טווח המהירות המותרת בכל סוג כביש. בחישוב זמן הנסיעה של הפונקציה היוריסטית, עליכם לחלק את הדרך במהירות המקסימלית האפשרית בין כל הכבישים.
8. הסבירו בדו"ח איזו פונקציה יוריסטית בחרתם ומדוע היוריסטיקה אכן קבילה.
9. השתמשו באותן 100 בעיות חיפוש שיצרתם בסעיף 3 והריצו עליהן חיפוש A^*. עבור כל אחת מהבעיות, פלטו לקובץ *results/AStarRuns.txt* את מסלול הנסיעה (כולל קצוות), זמניהם והזמן המשוערך ע"י היוריסטיקה מהמוצא ליעד.
(כל בעית חיפוש בשורה כאשר רשימת הצמתים תחילה ולאחריהם ' - ' זמן הנסיעה ' - ' הזמן המשוערך.)
הציגו בדו"ח גרף ובו נקודה לכל אחת מההרצות הנ"ל. (ציר ה X מייצג את זמן הנסיעה היוריסטי, וציר ה Y מייצג את זמן הנסיעה בפועל) מה ניתן ללמוד מהגרף על הקשר בין המשתנים?

> הקובץ main.py מרכז את הממשק אל שורת הפקודה; יש לכתוב בו מעט ככל הניתן.
> למשל, אם נקודת המוצא היא 30 ונקודת היעד 55:
> ```
> $ python main.py astar 30 55
> ```
> על הפלט להיות רשימה פשוטה של מספרי צמתים, כולל קצוות:
> ```
> 30 21 44 73 55
> ```

10. בהנחה שהרצתם את אלגוריתם A* בדיוק לפני שהעומס בכביש התחיל, האם בהכרח המסלול המתקבל יהיה אופטימלי? הסבירו את קביעתכם. (התשובה לשאלה זו אינה תכנותית)

חלק 4 - *IDA** (15 נקודות)

11. כתבו בפייתון פונקציה find_*idastar*_route המקבלת שני פרמטרים - צומת התחלה וצומת סיום. כמו בסעיף הקודם, השתמשו בפונקצית המחיר ~~הממומשת בקוד ונמצאת תחת tools.compute_distance~~ ופונקציה יוריסטית אותם מימשתם בסעיפים קודמים ~~הקודם (שנמצאת בקובץ main.py - huristic_function בשם~~. על הפונקציה להחזיר את המסלול המהיר ביותר מנקודת המוצא ליעד בעזרת אלגוריתם IDA* כ - TreeSearch.

חלק 5 – סיכום (10 נקודות)

12. השתמשו במתודה *draw.plot_path* בכדי ליצר מפה של 10 מהפתרונות שיצרתם (אותם תבחרו באופן אקראי). צרפו את התמונות תחת תקייה *sulotions_img*.
(ההדפסה תתבצע רק עבור אלגוריתם *IDA**)
13. לכל אחד מהאלגורתמים (*UCS*, *A**, *IDA**), כתבו את זמן ריצת האלגוריתם הממוצע (ללא טעינת המפה). מי מהאלגוריתם רץ בזמן הקצר ביותר? מדוע?

הוראות הגשה:

דו"ח

- **את הדו"ח יש להגיש כקובץ *PDF* בשם *report.pdf***
- **הקפידו לכתוב את האימיילים שלכם גם בראש הדו"ח.**
- **נמקו היטב את כל תשובותיכם.**

הקוד

- עליכם להגיש כל קוד שנכתב לצורך ביצוע המטלה.
- בראש כל קובץ רשמו את שימכם ות.ז.
- הנחיות לגבי קונבנציות בפייתון מופיעות כאן: *www.python.org/dev/peps/pep-0008/*
מומלץ לעבוד לפיהן אך אי עמידה בהן לא תפגע בציון.
- תיעוד למבנה התיקיות מופיע בקובץ *README.md*
- בהגשת התרגיל, תגישו את כל הקוד שסופק מלבד קובץ המפה בשל גודלו.

טרם הגשה:

- עדכנו את הקובץ *details.txt* עם הפרטים שלכם.
- שימו לב שכל הפנייה למיקום קובץ/תיקייה כלשהי בקוד תהיה רלטיבית ולא אבסולוטית (*relative path*), כך שהקוד יעבוד כפי שהוא על כל מחשב בכל מיקום שנבחר לתיקיית הפרויקט. הקפידו לבדוק זאת לפני ההגשה!
- נא לא להשתמש בחבילות נוספות על מה שכבר קיים בקוד שניתן לכם
- הסירו את קובץ המפה מהתיקייה *db/israel.csv*.
- הוסיפו את קובץ הדו"ח
- כווצו את התיקייה לארכיון *ZIP\RAR*.
- את הארכיון שיצרתם יש להגיש אלקטרונית דרך ה *submit*
- אנא וודאו שאתם מקבלים מייל על אישור הגשה.

בהצלחה ... ותיהנו!