תרגיל בית 1: שימוש באלגוריתמי חיפוש היוריסטיים לתכנון מסלולי חלוקה אופטימליים

מטרות התרגיל

- נתמודד עם בעיות פרקטיות ותיאורטיות של חיפוש במרחבי מצבים עצומים.
- נתרגל את הנלמד בהרצאות ובתרגולים.
- נתנסה בתכנות ב- python לפתרון בעיות פרקטיות.

הנחיות כלליות

- **תאריך הגשה:** יום שני, 18.05.2020, בשעה 23:59.
- את המטלה יש להגיש **בזוגות בלבד**.
- יש להגיש מטלות מוקלדות בלבד. פתרונות בכתב יד לא ייבדקו.
- ניתן לשלוח שאלות בנוגע לתרגיל לתיבת המייל הקורסית: ai.technion@gmail.com. אנו מבקשים לא לשלוח הודעות בנוגע לתרגיל לתיבות הדואר של הסגל. לפני שליחת שאלה, בדקו האם קיימת לה תשובה כבר ב- FAQ. נציין כי שאלות שנענו כבר ב- FAQ לא יענו שוב במייל.
- המתרגל האחראי על תרגיל זה: אלעד נחמיאס.
- בקשות דחיה **מוצדקות** (מילואים, אשפוז וכו') יש לשלוח למתרגל האחראי (גיא קושלביץ) בלבד.
- במהלך התרגיל ייתכן שנעלה עדכונים, תיקונים והבהרות לדף FAQ ייעודי באתר. העדכונים הינם **מחייבים**, ועליכם להתעדכן דרך עמוד זה.
- שימו לב, התרגיל מהווה כ- 13% מהציון הסופי במקצוע ולכן העתקות תטופלנה בחומרה.
- ציון המטלה יורכב מהגורמים הבאים:
  - **60% - המסמך היבש.** מעבר לתשובות הנכונות, אתם נבחנים גם על הצגת הנתונים והתוצאות בצורה קריאה ומסודרת. הניקוד המפורט בסעיפים של מסמך זה הינו מתוך הציון היבש בלבד.
  - **40% - הקוד המוגש.** הקוד שלכם ייבדק באופן מקיף ע"י מערכת בדיקות אוטומטיות. המערכת תבדוק את התוצאות שלכם לעומת התוצאות המתקבלות במימוש שלנו. אנו מצפים שתקבלו את אותם הערכים בדיוק. נבדוק את המסלול המתקבל, את עלותו ואת מס' הפיתוחים. לכן עליכם להיצמד להוראות בתרגיל זה. הבדיקות יהיו כמובן מוגבלות בזמן ריצה. ייתנן לכם זמן סביר ביותר להרצת כל טסט. אם תעקבו אחר ההוראות במסמך זה ובקוד אין סיבה שלא תעמדו בזמנים אלו. בנוסף, יש להקפיד על הגשת קוד מסודרת בהתאם להנחיות. יש לכתוב הערות במקומות חשובים בקוד כדי שיהיה קריא וקל לבדיקה ידנית.
- אנו יודעים שעבור חלקכם זו התנסות ראשונה בכתיבת קוד בפייתון ותרגיל זה מתוכנן בהתאם לכך.
- שימו לב שלא יענו שאלות בסגנון: "איך מוצאים את עלות הפתרון שהוחזר?" / "איך ניגשים למפות הכבישים מתוך המימוש של הפונק' ההיא?" / "באיזה שדה שמורה המהירות של הכביש?" / "אילו שדות מצפים לקבל אובייקט מטיפוס frozenset?" וכדומה. בכל מקום בקוד בהם אתם נדרשים להשלים את המימוש (לכתוב קוד כלשהו) השארנו לכם הערות מפורטות שמסבירות כיצד יש לעשות זאת. ברוב המקומות גם הכוונו אתכם במפורש לשמות השדות ולמתודות הרלוונטיות להם תזדקקו. בנוסף, כל הקוד כתוב עם type-annotations (למרות שאין הכרח לציין טיפוסים בפייתון) במטרה להקל עליכם בהתמצאות בקוד ובכדי שתוכלו להבין מה אמור לקבל כל שדה/ארגומנט. אנחנו מצפים מכם להשכיל ולהשתמש ב- IDE (ממליצים על PyCharm) שיוכל לסייע לכם להתמצא בקוד ביתר קלות ויזהה עבורכם שגיאות בצורה סטטית – זה יחסוך לכם הרבה זמן. ה- type-annotations שהוספנו לקוד עוזרות ל- IDEs לעזור לכם – נצלו את זה. בחלק מהמקומות החסרנו חלק מהפרטים בהסבר מתוך כוונה – אנחנו רוצים לעודד אתכם לעיין בקוד ולמצוא פרטים אלו בכוחות עצמכם. הכרת סביבת העבודה שסיפקנו לכם והתמצאות בה הן למעשה חלק מהתרגיל.
- בתרגילי הבית בקורס הרצת הניסויים עשויה לקחת זמן רב, ולכן מומלץ מאוד להימנע מדחיית העבודה על התרגיל ו/או כתיבת הדו"ח לרגע האחרון. לא תינתנה דחיות על רקע זה.
- מסמך זה כתוב בלשון זכר מטעמי נוחות בלבד, אך מתייחס לנשים וגברים כאחד.

הערות טכניות

- גרסת python איתה אתם נדרשים לעבוד הינה 3.7. גם קבצי המקור שקיבלתם מתאימים לגרסה זו.
- כאמור, הבדיקות האוטומטיות של הקוד שתגישו תהיינה מוגבלות בזמן פר טסט. היו סמוכים ובטוחים שמערכת הבדיקה הינה הוגנת ביותר. מימוש תקין שנצמד להוראות יעמוד במסגרת

הזמנים. הסיבה למגבלת הזמן היא פשוטה – לא ניתן להריץ כל טסט אינסוף זמן – אנחנו צריכים לבדוק את כל התרגילים שלכם במסגרת זמן סבירה. בכדי לעמוד במסגרת הזמנים אתם לא מתבקשים לחשוב על אופטימיזציות כאלו או אחרות, אלא רק לעקוב באדיקות אחר ההוראות. הבינו איך משתמשים ב- iterators בפייתון ונסו להשתמש בהם בכל מקום שתוכלו (במקום ליצור רשימות איפה שאין באמת צורך בכך). אנו מכווינים אתכם לעשות כך בחלק מהסעיפים. קשה לפרט דרישת זמנים קשיחה כי לכל אחד יש מחשב בעל מפרט אחר. נפרט כאן הערכה כללית לזמן הריצה הצפוי של מימוש תקין במחשב אישי מודרני סביר, וזאת רק בכדי שתוכלו לקבל סדר גודל ולוודא שאתם לא חורגים מכך באופן דראסטי. אם אתם חורגים מהאמור באופן דרסטי ייתכן שיש לכם טעות במימוש – היעזרו אחד בשני כדי למצוא אותה. הריצה הארוכה ביותר אמורה לקחת לכל היותר דקה. היעזרו בהערכה גסה זו כדי לוודא/לחשוד בתקינות המימוש שלכם.

- אלא אם נכתב אחרת, אין לשנות פונקציות מוכנות שקיבלתם. בנוסף, אין לשנות את החתימה של פונקציות שהתבקשתם לממש או אחרות. בפרט, אין לשנות תוכן קבצים בהם לא נתבקשתם לבצע שינויים. אין ליצור פונקציות עזר משלכם, אנא השלימו את המימושים אך ורק במקומות המסומנים. בנוסף, אין ליצור קבצים חדשים, אלא לערוך את הקבצים שהתבקשתם במפורש בלבד. ראו הוזהרתם - חריגה מכללים אלו יכולה להוביל לכישלון מידי בבדיקות האוטומטיות. אם יש בעיה נקודתית, ניתן לשלוח מייל לתיבה הקורסית.
- אין להוסיף ו/או לשנות פקודות import בקוד. כל מה שאתם צריכים כבר מיובא במקום הרלוונטי. שימו לב שלעיתים IDEs שונים עלולים לעיתים להוסיף לכם שורות import באופן אוטומטי. אחריותכם לוודא, טרם הגשת התרגיל, ששורות ה- import בקוד אותו אתם מגישים זהות לשורות בקבצים המקוריים שקיבלתם.
- אין לבצע בעצמכם טעינה של קלטים או מפות. אנחנו עשינו זאת עבורכם במקומות הנדרשים. בכל אזור בקוד בו שהתבקשתם להשלים את המימוש יש גישה לכל המבנים להם אתם זקוקים לצורך המימוש. ראו הוזהרתם - חריגה מכללים אלו יכולה להוביל לכישלון מידי בבדיקות האוטומטיות.
- לצורך ההרצות תצטרכו להתקין את החבילות הבאות של python: numpy, scipy, matplotlib, networkx. חלק מחבילות אלו מותקנות כברירת מחדל עם ההתקנה של Anaconda. את אלו שאינן מותקנות אפשר להתקין בעזרת הפקודה `pip install <package name>`.

ייתכן שמסמך זה יתעדכן באתר – הבהרות ועדכונים שנוספים אחרי הפרסום הראשוני יסומנו בצהוב.

חלק א' – מבוא והנחיות (2.5 נק' יבש)

במטלה זו נעסוק בהפעלת אלגוריתמי חיפוש על מרחבי מצבים גדולים במיוחד לבעיות ניווט. מומלץ לחזור על שקפי ההרצאות והתרגולים הרלוונטיים לפני תחילת העבודה על התרגיל.

במהלך התרגיל תתבקשו להריץ מספר ניסויים ולדווח על תוצאותיהם. אתם נדרשים לבצע ניתוח של התוצאות, כפי שיוסבר בהמשך.

מוטיבציה

ברקע התפרצות נגיף הקורונה בישראל, מד"א עובדים סביב השעון בביצוע בדיקות לאבחון הוירוס. מד"א מגיעים לביתו של כל מי שמדווח על תסמינים ובודקים אותו ואת כל הדיירים המתגוררים ביחד איתו. במקביל ללימודיו בטכניון, מוטי מתנדב במד"א והינו בעל הכשרה לנהג אמבולנס. בתחילת המשמרת מוטי מקבל רשימה של כל הבדיקות שיש לבצע ומיד יוצא לדרך.

באמבולנס יש מקרר מיוחד שבו ניתן לשמור את כל הבדיקות שנלקחו עד כה. המקום במקרר מוגבל, וכאשר מתמלא מוטי צריך לעבור באחת מהמעבדות האזוריות כדי להעביר להם את הבדיקות ולפנות מקום במקרר. בנוסף, עקב המחסור במטושים, מספר המטושים הזמינים (והנדרשים לצורך הבדיקות) הינו מוגבל. כאשר מוטי עובר במעבדה, פרט לפריקת הבדיקות, הוא גם לוקח משם את כל המטושים הזמינים. כאשר נגמרים למוטי המטושים באמבולנס הוא חייב לעבור במעבדה, גם אם המקרר שלו ריק ואין לו בדיקות לפרוק. בכל מעבדה יש מספר אחר של מטושים זמינים.

מוטי עמוס בלימודים ולכן הוא רוצה לסיים את המשמרת כמה שיותר מהר ולהגיע הביתה כדי לעבוד על ההגשות שלו. למזלו, חברים של מוטי (זה אתם!) במקרה לוקחים הסמסטר את הקורס "מבוא לבינה מלאכותית". מוטי מבקש מכם לעזור לו לתכנן מראש את הדרך היעילה ביותר לבצע את כל הבדיקות.

פורמאליזם – הגדרת הבעיה

נתונה רשת כבישים בצורת גרף $StreetsMap = (V_{map}, E_{map})$ שבה כל צומת מייצג צומת דרכים (junction), והקשתות מייצגות דרך (כביש) המקשרת בין צמתי דרכים (links).

לאמבולנס יש קיבולת מרבית של $AmbulanceTestsCapacity$ בדיקות.

נתונה נקודת מוצא על רשת הכבישים $v_0 \in V_{map}$, וכן נתונות $k \in \mathbb{N}$ דירות שאליהן יש להגיע ולבצע בדיקה: $Apartments = \{d_1, \ldots, d_k\}$, כאשר דירה i כוללת: מיקום $d_i.loc \in V_{map}$, ומספר הדיירים שיש לבדוק $d_i.roommates \in \{1,2, \ldots, AmbulanceTestsCapacity\}$.

נתונות $m \in \mathbb{N}$ מעבדות $Labs = \{l_1, \ldots, l_m\}$. לכל מעבדה יש מיקום $l_i.loc \in V_{map}$ וכן מספר מטושים זמינים $l_i.matoshim \in \mathbb{N}^+$.

לצורך פשטות, במהלך כל התרגיל נניח כי הדירות, המעבדות ונק' המוצא הינן נקודות זרות במפה. כלומר $\left|\{v_0\} \cup \{d_i.loc\}_{i\in[k]} \cup \{l_i.loc\}_{i\in[m]}\right| \equiv k + m + 1$.

<u>*סידור ביקורים*</u> הינו פרמוטציה $\pi = w_1, \ldots, w_k$ של הנקודות $\{d_i.loc\}_{i\in[k]}$.

את איכות סידור ביקורים π שיחושב ע"י התוכנית נמדוד לפי מספר מדדים שונים, כפי שיפורט בהמשך. הפתרון לבעיה לפי מדד איכות נתון הינו סידור ביקורים אצל לקוחות בעל מחיר מינימלי ע"פ מדד איכות זה.

הבנת קושי הבעיה

בשלב זה אנחנו רוצים לקבל קצת אינטואיציה לגבי הקושי של הבעיה. המטרה היא להשתכנע שאנחנו לא מסוגלים לפתור את הבעיה בעזרת חיפוש brute-force (בגלל מגבלת משאבים). לצורך זאת, ראשית ננסה להעריך את מספר הסידורים החוקיים השונים אותם יש לבחון במסגרת ריצת brute-force. לשם פשטות החישוב אנו מתעלמים כרגע משאר אילוצי הבעיה.

תרגיל

1. יבש (2.5 נק'): מלאו את הטבלה הבאה. הזינו את מספר הפרמוטציות האפשריות (וערכי $\log_2$ שלהן) עבור ערכי k (מספר ההובלות) המופיעים בטבלה. היעזרו בנוסחה שמצאתם בסעיף (1). נניח שמחשב יחיד יכול לבחון 2^{30} סידורים בשנייה. מלאו בעמודה האחרונה כמה זמן ייקח למחשב זה לבדוק כל אחד מהסידורים (לפי היחידות המפורטות).

k	$\#possiblePaths$	$\log_2(\#possiblePaths)$	Calculation time
5			< 1 sec
8			[sec]
9			[hours]
10			[years]
11			[years]
15			[million years]

חלק ב' – הגדרת מרחב החיפוש במפה

כאמור נתונה רשת כבישים בצורת גרף $StreetsMap = (V_{map}, E_{map})$. בעיית המפה עוסקת במציאת מסלול ברשת הכבישים $StreetsMap$ בעל עלות מינימלית (ביחס לפונק' עלות נתונה המוגדרת על כבישים במפה). בחלק זה נייצג את בעיית המפה כמרחב חיפוש. ניצמד להגדרה שלמדנו בכיתה עבור מרחבי חיפוש. אנו מתחילים בבעיית המפה משום שהיא בעיה יחסית פשוטה, הייצוג שלה כמרחב חיפוש הוא אינטואיטיבי ואנו אכן נעשה בה שימוש בחלקים הבאים.

בהינתן רשת הכבישים, נקודת מקור $v_{src} \in V_{map}$ ונקודת יעד $v_{dst} \in V_{map}$, נגדיר מרחב חיפוש עבור מציאת מסלול ביניהן:

$$\mathcal{S}_{map} \triangleq \langle S_{map}, O_{map}, I_{map}, G_{map} \rangle$$

- **קבוצת המצבים:**
  נרצה לייצג מצב כך שיחזיק את כל המידע שנחוץ לנו עליו במהלך החיפוש במרחב. במקרה המדובר מספיק לשמור את הצומת ברשת הכבישים.
  $$S_{map} \triangleq \{(v: u) | u \in V_{map}\}$$
- **קבוצת האופרטורים:**
  ניתן לעבור ממצב אחד לעוקבו בתנאי שיש כביש מהצומת המיוצג ע"י המצב הראשון לצומת המיוצג ע"י המצב העוקב.
  $$O_{map} \triangleq \{(s_1, s_2) | s_1, s_2 \in S_{map} \wedge (s_1.v, s_2.v) \in E_{map}\}$$
- **עלות אופרטור:**
  נגדיר את פונק' העלות עבור מעבר מצומת דרכים אחד $s_1 \in S_{map}$ לצומת דרכים עוקב שלו $o \in O_{map}, s_2 = o(s_1)$ באופן הבא:
  $$cost_{map}^{dist}((s_1, s_2)) = roadLength((s_1.v, s_2.v))$$
- **המצב ההתחלתי:**
  $$I_{map} \triangleq (v: v_{src})$$
- **מצבי המטרה:**
  $$G_{map} \triangleq \{(v: v_{dst})\}$$

חלק ג' – הגדרת מרחב החיפוש של בעיית ההובלות (9 נק׳ יבש)

בהינתן רשת הכבישים, נקודת המוצא ורשימת ההזמנות, נגדיר מרחב חיפוש עבור בעיית מד"א:

$$\mathcal{S}_{MDA} = \langle S_{MDA}, O_{MDA}, I_{MDA}, G_{MDA} \rangle$$

- **קבוצת המצבים:**

$$S_{MDA} \triangleq \{(v_0, \emptyset, \emptyset, InitialNrMatoshimAmb, \emptyset)\} \cup \left\{ \left(\begin{matrix} \underbrace{curLoc}_{\substack{\text{המיקום} \\ \text{הנוכחי}}}; \underbrace{Taken}_{\substack{\text{קבוצת בדיקות} \\ \text{על האמבולנס}}}; \underbrace{Transferred}_{\substack{\text{קבוצת בדיקות} \\ \text{שהועברו למעבדה}}}; \\ \underbrace{Matoshim}_{\substack{\text{מספר מטושים} \\ \text{זמינים באמבולנס}}}; \underbrace{VisitedLabs}_{\substack{\text{המעבדות} \\ \text{שבוקרו}}} \end{matrix} \right) \middle| \begin{matrix} curLoc \in \\ \{d.loc | d \in Loaded\} \cup \{l.loc | l \in LabsVisited\} \\ Taken \cup Transferred \subseteq Apartments \\ Taken \cap Transferred = \emptyset \\ Matoshim \in \mathbb{N} \\ VisitedLabs \subseteq Labs \end{matrix} \right\}$$

- **קבוצת האופרטורים:**

<u>אופרטורים עבור ביקור בדירה:</u>

ישנם k אופרטורים כאלו. לכל $i \in [k]$ נגדיר את האופרטור o_{d_i} להיות האופרטור שבהינתן מצב $s \in S_{MDA}$, המצב העוקב $o_{d_i}(s)$ (המתקבל מהפעלת האופרטור על המצב s) הינו מצב שבו המיקום הנוכחי הוא הנק׳ $d_i.loc$, מס׳ המטושים הזמינים באמבולנס קטן ב- $d_i.roommates$ וכן הבדיקות שנלקחו מדירה d_i נמצאות במקרר של האמבולנס ($d_i \in Taken$).

הפעלת האופרטור o_{d_i} על מצב $s \in S_{MDA}$ אפשרית אמ"מ הבדיקות של הדירה d_i לא נלקחו כבר ($d_i \notin s.Taken \cup s.Transferred$), יש באמבולנס מספיק מטושים זמינים בשביל לקחת בדיקות לכל הדיירים בדירה d_i, וכן יש די מקום פנוי במקרר באמבולנס עבור אחסון כל הבדיקות מדירה זו, כלומר מתקיים התנאי:

$$CanVisit(s, d_i) \triangleq \left[\begin{matrix} d_i \notin s.Taken \cup s.Transferred \\ \wedge \\ d_i.roommates \leq s.Matoshim \\ \wedge \\ d_i.roommates \leq AmbulaceTestsCapacity - \sum_{d \in s.Taken} d.roommates \end{matrix} \right]$$

הגדרה פורמלית: לכל $i \in [k]$ נגדיר את האופרטור o_{d_i} באופן הבא:

$$\forall_{s \in S_{MDA}}: o_{d_i}(s) \triangleq \begin{cases} \left(\begin{matrix} d_i.loc; s.Taken \cup \{d_i\}; s.Transferred, \\ s.Matoshim - d_i.roommates; s.VisitedLabs \end{matrix} \right) & ; \quad CanVisit(s, d_i) \\ \emptyset & ; \quad otherwise \end{cases}$$

וכן תחום הפעולה של האופרטור o_{d_i} מוגדר בהתאם:

$$Domain(o_{d_i}) = \{s \in S_{MDA} | o_{d_i}(s) \neq \emptyset\} = \{s \in S_{MDA} | CanVisit(s, d_i)\}$$

<u>אופרטורים עבור מעבר במעבדה:</u>

ישנם m אופרטורים כאלו. לכל $i \in [m]$ נגדיר את האופרטור o_{l_i} להיות האופרטור שבהינתן מצב $s \in S_{MDA}$, המצב העוקב $o_{l_i}(s)$ (המתקבל מהפעלת האופרטור על המצב s) הינו מצב שבו המיקום הנוכחי הוא הנק׳ $l_i.loc$, הבדיקות שבמקרר באמבולנס מועברות למעבדה (המקרר נותר ריק), וכן המטושים הזמינים במעבדה מאוחסנים באמבולנס.

הפעלת האופרטור o_{l_i} על מצב $s \in S_{MDA}$ אפשרית רק אם המקרר באמבולנס אינו ריק **או** שהמעבר במעבדה יוסיף מטושים נוספים לאמבולנס (לא עברנו במעבדה זו בעבר). כלומר מתקיים התנאי:

$$CanVisit(s, l_i) = s.Taken \neq \emptyset \ \vee \ l_i \notin s.VisitedLabs > 0$$

הגדרה פורמלית: לכל $i \in [m]$ נגדיר את האופרטור o_{l_i} באופן הבא:

$$\forall_{s \in S_{MDA}}: o_{l_i}(s) \triangleq \begin{cases} \left(\begin{matrix} l_i.loc; \emptyset; s.Transferred \cup s.Taken; \\ s.Matoshim + l_i.matoshim \cdot \mathbb{I}_{l_i \notin s.VisitedLabs}; \\ s.VisitedLabs \cup \{l_i\} \end{matrix} \right) & ; \quad CanVisit(s, l_i) \\ \emptyset & ; \quad otherwise \end{cases}$$

וכן תחום הפעולה של האופרטור o_{l_i} מוגדר בהתאם:

$$Domain(o_{l_i}) = \{s \in S_{MDA} | o_{l_i}(s) \neq \emptyset\} = \{s \in S_{MDA} | CanVisit(s, l_i)\}$$

לבסוף, קבוצת כל האופרטורים הינה:

$$O_{MDA} \triangleq \{o_{d_i}\}_{i\in[k]} \cup \{o_{l_i}\}_{i\in[m]}$$

- **עלות אופרטור:**
במטלה נגדיר 2 פונקציות עלות עבור הפעלת אופרטור $o \in O_{MDA}$ על מצב $s \in Domain(o)$.
  1. אורך המסלול הקצר ביותר על גבי המפה מהנק' בה נמצא האמבולנס במצב s לנק' בה מצוי האמבולנס במצב $s(o)$:
  $$cost_{MDA}^{dist}(s,o) \triangleq optimalDistanceOnStreetsMap(s.curLoc, o(s).curLoc)$$
  2. מרחקי הנסיעה שעברו כל הבדיקות במקרר:
  $$cost_{MDA}^{test\ travel}(s,o) \triangleq \left[\sum_{d\in s.Taken} d.roommates\right] \cdot cost_{MDA}^{dist}(s,o)$$

  - כל אחת משתי פונק' העלויות הללו למעשה מגדירה ווריאציה לבעיה. בסופו של דבר כשפותרים בעיה צריך להחליט באיזו פונק' עלות משתמשים.
  - בחלקים הראשונים של התרגיל נשתמש בפונק' העלות $cost_{MDA}^{dist}$ ובחלקים מתקדמים נעשה שימוש ב- $cost_{MDA}^{test\ travel}$.
  - שימו לב: בהינתן אופרטור $o \in O_{MDA}$ ומצב $s \in Domain(o)$, על מנת לחשב את $cost_{MDA}^{dist}(s,o)$ או את $cost_{MDA}^{test\ travel}(s,o)$, **יש צורך בפתרון של בעיית המפה**.

- **המצב ההתחלתי:**
$$I_{MDA} \triangleq (v_0, \emptyset, \emptyset, InitialNrMatoshimAmb, \emptyset)$$
- **מצבי המטרה:**
$$G_{MDA} \triangleq \{(l_i.loc, \emptyset, Apartments, M, L) \in S | i \in [m], M \in \mathbb{N}, L \subseteq Labs\}$$

תרגילים

2. יבש (1.5 נק'): מהם ערכי הקיצון (המקסימלי והמינימלי) האפשריים של דרגת היציאה במרחב החיפוש? נמקו בקצרה.
3. יבש (1.5 נק'): האם ייתכנו מעגלים במרחב המצבים שלנו? אם כן תנו דוגמה למעגל כזה, אחרת נמקו.
4. יבש (1.5 נק'): כמה מצבים יש במרחב זה (כפי שהוגדר)? האם כולם ישיגים? נמקו.
5. יבש (1.5 נק'): האם ייתכנו בורות ישיגים מהמצב ההתחלתי שאינם מצבי מטרה במרחב המצבים? אם כן – איך זה ייתכן? אם לא – למה?
6. יבש (1.5 נק'): מהו טווח האורכים האפשריים של מסלולים במרחב ממצב התחלתי אל מצב סופי? (אורך מסלול = מס' הקשתות)
7. יבש (1.5 נק'): הגדירו פורמלית ובצורה ישירה את פונקציית העוקב $Succ: S \rightarrow \mathcal{P}(S)$ המתאימה לבעיה זו (ללא שימוש בקבוצת האופרטורים O).
שימו לב, אנו מצפים לביטוי מהצורה: $Succ(s) = \{(?,?,?,?,?)|?\} \cup \{(?,?,?,?,?)|?\}$

חלק ד' – מתחילים לתכנת (1 נק' יבש)

הורידו את *ai_hw1.zip* מהאתר וטענו את התיקייה שבתוכו לסביבת העבודה המועדפת עליכם.

מבנה מפת הדרכים

בתרגיל נעשה שימוש במפת רשת הכבישים של העיר תל אביב. את המפה אנו טוענים פעם אחת בקובץ main.py למשתנה גלובלי בשם streets_map. המפות מיוצגות ע"י אובייקט מטיפוס ***StreetsMap***. הטיפוס ***StreetsMap*** יורש מ- *dict*; כלומר ***StreetsMap*** הינו בבסיסו מיפוי ממזהה ייחודי של צומת במפה (מספר שלם) לאובייקט מטיפוס ***Junction*** שמייצג את אותו הצומת.

כל צומת הוא כאמור מטיפוס ***Junction***. לצומת יש את השדות הבאים: (1) מספר *index* ייחודי; (2+3) קואורדינטות lat, lon (קווי אורך ורוחב) של המיקום הגיאוגרפי של הצומת במפה; ו- (4) רשימה outgoing_*links* המכילה את כל הקשתות לשכניו. כל קשת כזו מייצגת כביש במפה. קשת היא אובייקט מטיפוס ***Link*** עם מאפיינים source ו- target – המזהים של צמתי המקור והיעד של הקשת, distance – אורך הכביש (במטרים).

שימו לב: אין לבצע באף שלב טעינה של מפות. טענו בשבילכם את המפות פעם אחת בתחילת קובץ ה- main.py שסיפקנו לכם. יש לכם גישה למפות בכל מקום בו תזדקקו להן. באופן כללי, טעינות מיותרות בקוד יגרמו להגדלת זמן הפתרון ואולי יובילו לחריגה מהזמן המקסימלי.

הכרת תשתית הקוד הכללית (שסופקה לכם בתרגיל זה) לייצוג ופתרון בעיות גרפים

המחלקות **GraphProblem**, **GraphProblemState** (בקובץ graph_search/graph_problem_interface.py) מגדירות את הממשק (interface) בו נשתמש על מנת לייצג מרחב מצבים. אלו הן מחלקות אבסטרקטיות – כלומר מוגדרות בהן מתודות שאינן ממומשות. לכן, בפרט, לא ניתן ליצור ישירות אובייקט מטיפוסים אלו (ואין לכך שום משמעות). כדי להגדיר מרחב מצבים חדש יש לרשת (inherit) משתי המחלקות הנ"ל. בהמשך התרגיל תראו דוגמא למימוש של מרחב מצבים באופן הנ"ל (שסיפקנו עבורכם) ותממשו מרחב נוסף כזה בעצמכם.

המחלקה **GraphProblemSolver** (באותו הקובץ) מגדירה את הממשק בו נשתמש בכדי לחפש בגרפים. למחלקה יש מתודה אבסטרקטית אחת בשם ()solve_problem שמקבלת כפרמטר בעיה (אובייקט מטיפוס שיורש מ- GraphProblem) ומחזירה את תוצאות החיפוש (אובייקט מטיפוס SearchResult). כל אלג' חיפוש שנממש ישתמש בממשק הנ"ל (ירש ממחלקה זו או ממחלקה שיורשת ממנה).

שימו לב: אלגוריתמי החיפוש אותם נממש לאורך התרגיל יהיו כלליים בכך שלא יניחו כלום על הבעיות אותן יפתרו, פרט לכך שהן תואמות לממשק המוגדר ע"י GraphProblemState, GraphProblem. כלומר, בעתיד תוכלו לקחת את המימוש שלכם מקורס זה כפי שהוא בכדי לפתור בעיות חדשות.

המחלקה **BestFirstSearch** (בקובץ graph_search/best_first_search.py) יורשת מהמחלקה GraphProblemSolver (שתוארה לעיל) ומייצגת אלגוריתמי חיפוש מהמחלקה Best First Search. כפי שנלמד בכיתה, אלו הם אלגוריתמים שמתחזקים תור עדיפויות בשם **open** של צמתים (פתוחים) הממתינים לפיתוח. כל עוד תור זה אינו ריק, האלג' בוחר את הצומת הבא בתור העדיפויות ומפתח אותו. המחלקה מממשת את המתודה ()solve_problem בהתאם. דוגמאות לאלגוריתמים ממשפחה זו: *Uniform Cost, Greedy Best Search, A. כאמור, Best First Search הינה <u>משפחה</u> של אלגוריתמי חיפוש (מכונה גם "אלגוריתם גנרי"), כלומר היא מגדירה שלד כללי של מבנה האלגוריתם, ומשאירה מספר פרטי מימוש חסרים. לכן, המחלקה BestFirstSearch אף היא <u>אבסטרקטית</u>. גם בה מוגדרות מספר מתודות אבסטרקטיות שעל היורש (אלגוריתם החיפוש הקונקרטי) לממש. המתודה האבסטרקטית ()_calc_node_expanding_priority מאפשרת ליורש להגדיר את אופן חישוב ערך ה- f-score של צומת. כזכור, ערך זה משמש כעדיפות של צומת בתור העדיפויות open (בתרגיל זה אנו מכנים ערך זה בשם expanding priority). המתודה האבסטרקטית ()_open_successor_node מאפשרת ליורש להגדיר את אופן הטיפול בצומת חדש שזה עתה נוצר ומייצג מצב עוקב של המצב המיוצג ע"י הצומת שנבחר אחרון לפיתוח (הכנסה ל- open, בדיקה ב- close במידת הצורך). בנוסף, האלגוריתם מאפשר מצב של חיפוש-גרף כפי שנלמד בכיתה, ע"י תחזוק אוסף **סגור / close** של צמתים שכבר פיתחנו במהלך החיפוש (ה- constructor של BestFirstSearch מקבל פרמטר בולאני בשם use_close שקובע האם להשתמש ב- close).

מבנה הקלטים לבעיית מד"א ואופן טעינתם

המחלקה **MDAProblemInput** (בקובץ problems/mda_problem_input.py) מייצגת <u>קלט</u> לבעיית מד"א. מחלקה זו אחראית לטעינה של קלטים שסיפקנו לכם כקבצי טקסט. המחלקה שמייצגת את בעיית מד"א (נראה בהמשך) תקבל אובייקט מסוג זה. בקובץ הראשי main.py כבר כתובות שורות הקוד שאחראיות להשתמש במחלקה זו ע"מ לטעון את הקלטים הנדרשים במקומות הנדרשים. <u>הבהרה</u>: אין לבצע טעינות נוספות של הקלטים. אנו עשינו זאת בשבילכם בכל המקומות הנדרשים.

תרגילים

8. רטוב + יבש: סעיף זה נועד על מנת להתחיל להכיר את מבנה הקוד.
   a. חלצו את תוכן התיקייה ai_hw1.zip.
   b. אם אתם משתמשים ב- IDE לכתיבת והרצת קוד פייתון (אנחנו ממליצים מאוד על PyCharm), פתחו פרויקט חדש שתיקיית האם שלו היא התיקייה הראשית של קובץ ה- zip שחולץ (אמור להיות שם קובץ בשם main.py).
   c. פתחו את הקובץ main.py, קראו את החלק בקוד שמעליו מופיעה הערה המתאימה למספר סעיף זה. שורות קוד אלו מבצעות: יצירת בעיית מפה חדשה, יצירת אובייקט מסוג אלג׳ חיפוש uniform cost, הרצת אלג׳ החיפוש על הבעיה ולבסוף הדפסת התשובה שהתקבלה מההרצה. הריצו את הקובץ. וודאו שמודפסת לכם שורה למסך שמתארת את פתרון בעיית החיפוש במפה. זאת גם הזדמנות טובה לוודא שהחבילות numpy, scipy, networkx, matplotlib מותקנות אצלכם כראוי.
   d. פתחו את הקובץ problems/map_problem.py. בתוכו יש לכם שתי משימות (המסומנות ע"י הערות **TODO** כמו בעוד מקומות רבים לאורך המטלה). אחת במתודה בשם expand_state_with_costs() והשנייה במתודה בשם is_goal(). בשתי משימות אלו אתם מתבקשים לבצע שינוי בקוד של המחלקה MapProblem כדי לתקן ולהשלים את המימוש שסיפקנו לכם.
   e. זוהי בעיה פשוטה ולכן נוח להתחיל בה כדי להתמצא בקוד שסופק לכם. עיינו במימוש של המחלקות בקובץ זה זו וודאו שאתם מבינים את החלקים השונים. שימו לב שמחלקה זו יורשת מהמחלקה GraphProblem (שתוארה מקודם) וממשת את המתודות האבסטרקטיות הנדרשות.
   f. עתה, לאחר תיקון קוד המחלקה MapProblem, הריצו בשנית את main.py.
   g. יבש (1 נק׳): הוסיפו לדו״ח את פלט הריצה המתוקנת.

חלק ה' – אלגוריתם *A (3 נק׳ יבש)

עתה נתחיל במימוש *Weighted A.

עיינו בקובץ framework/graph_search/astar.py. שם מופיע מימוש חלקי למחלקה AStar. שימו לב: המחלקה AStar יורשת מהמחלקה האבסטרקטית BestFirstSearch (הסברנו עליה בחלק ד׳). זהו את החלק בהצהרת המחלקה AStar בו הירושה מוגדרת. המחלקה AStar צריכה לממש את המתודות האבסטרקטיות שמוגדרות ע״י BestFirstSearch. הכותרות של מתודות אלו מופיעות כבר במימוש החלקי של המחלקה AStar, אך ללא מימושן. בסעיף זה נרצה להשלים את המימוש של המחלקה AStar ולבחון אותה.

שימו לב: לאורך התרגיל כולו אין לשנות את החתימות של המתודות שסיפקנו לכם. בנוסף, אין לשנות קבצים שלא התבקשתם באופן מפורש.

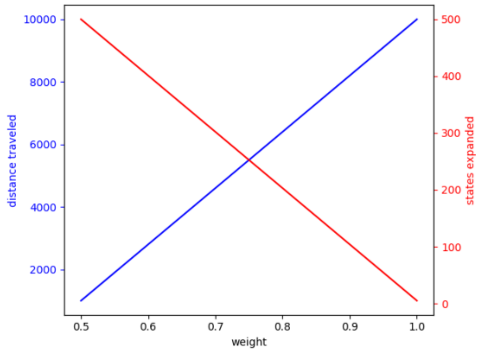
תרגילים

9. רטוב: השלימו את המשימות הדרושות תחת הערות ה- **TODO** בקובץ framework/graph_search/astar.py כך שנקבל מימוש תקין לאלגוריתם *Weighted A, כפי שראיתם בהרצאות. בכדי להבין את מטרת המתודות השונות שעליכם לממש, הביטו במימוש המחלקה BestFirstSearch שעושה בהן שימוש. בנוסף, היעזרו במימוש שסיפקנו לכם ל- UniformCost (בקובץ framework/graph_search/uniform_cost.py). שימו לב בשקפים מההרצאה להבדלים בין אלג׳ UniformCost לבין אלג׳ *A.
10. רטוב: בכדי לבחון את האלג׳ שזה עתה מימשתם, השלימו את המשימות הדרושות תחת הערות ה- **TODO** הרלוונטיות לסעיף זה בקובץ main.py. כידוע, לצורך הרצת *A יש צורך בהיוריסטיקה. ה- constructor של המחלקה AStar מקבל את טיפוס ההיוריסטיקה שמעוניינים להשתמש בה. לצורך בדיקת שפיות, הפעילו את ה- *A על בעיית המפה שפתרתם בסעיף הקודם עם NullHeuristic (מסופקת בקובץ framework/graph_search/graph_problem_interface.py. מחלקה זו כבר מוכרת מ- main.py ללא צורך בביצוע import נוסף. באופן כללי אין לעשות imports בתרגיל זה כלל). וודאו שהתוצאה המודפסת זהה לזו שקיבלתם בעזרת Uniform Cost.
11. רטוב + יבש (1 נק׳): כפי שראינו בהרצאות ובתרגולים, היוריסטיקה פשוטה לבעיית המפה היא מרחק אווירי לפתרון. היכנסו לקובץ problems/map_heuristics.py וממשו את ההיוריסטיקה הזו במחלקה AirDistHeuristic (מלאו את המקומות החסרים תחת ההערות שהשארנו לכם שם). כעת הריצו שוב את הבעיה שפתרתם בסעיף הקודם, אך כעת בעזרת ההיוריסטיקה (מלאו ב- main.py את המשימות שקשורות לסעיף זה). העתיקו לדו"ח את פלט הריצה. כתוב בדו״ח את מס׳ פיתוחי המצבים היחסי שחסכנו לעומת הריצה העיוורת (ההפרש חלקי מס׳ הפיתוחים בריצה עם ההיוריסטיקה).

שימו לב: בכדי לחשב מרחק בין זוג Junctions, **אין לחשב את המרחק האווירי ישירות על ידי קווי רוחב ואורך**, אלא יש להשתמש במתודה ()calc_air_distance_from של המחלקה Junction.

12. רטוב + יבש (2 נק׳): כעת נרצה לבחון את השפעת המשקל w על ריצת *wA. מלאו בקובץ main.py את המשימות הרלוונטיות לסעיף זה. בנוסף, ממשו את הפונק׳ ()run_astar_for_weights_in_range שחתימתה מופיעה בקובץ main.py. פונק׳ זו מקבלת היוריסטיקה ובעיה לפתרון ומשתמשת באלג׳ *wA בכדי לפתור את בעיה זו תוך שימוש בהיוריסטיקה הנתונה ועם n משקולות שונות בתחום הסגור $[0.5,0.95]$. את התוצאות של ריצות אלו היא אמורה לשמור ברשימות ולאחר מכן היא אמורה לקרוא לפונק׳ בשם ()plot_distance_and_expanded_wrt_weight_figure (שגם בה עליכם להשלים את המימוש באיזורים החסרים). פונק׳ זו אחראית ליצור גרף שבו מופיעות 2 עקומות: אחת מהעקומות (הכחולה) מתארת את טיב הפתרונות (בציר y) כפונק׳ של המשקל (אורך המסלול במקרה של בעיית המפה הבסיסית). העקומה השנייה (האדומה) מתארת את מספר המצבים שפותחו כפונק׳ של המשקל. עתה השתמשו בפונק׳ ()run_astar_for_weights_in_range מהמקום הרלוונטי ב- main.py (מספר סעיף זה מצוין במקום זה) ע״מ ליצור את הגרף המתאים עבור פתרון בעיית המפה תוך שימוש בהיוריסטיקה AirDistHeuristic. צרפו את הגרף שנוצר לדו"ח. הסבירו את הגרף שהתקבל. ציינו באיזה ערך w הייתם בוחרים ולמה. בכיתה למדתם כלל אצבע לפיו ״ככל ש- w קטן יותר כך הפתרון איכותי יותר ומס׳ הפיתוחים גדול יותר״. הכלל הנ״ל מצביע על מגמה כללית, אך איננו נכון באופן גורף (כלומר ייתכנו זוג ערכים $w_1 < w_2$ עבורם הפתרון המתקבל עם w_1 פחות טוב מאשר הפתרון המתקבל עם w_2 ו/או מס׳ הפיתוחים עם w_2 גדול יותר ממס׳ הפיתוחים עם w_1). כיצד הכלל שהוזכר והדגש הנ״ל באים לידי ביטוי בתרשים שקיבלתם? על התרשים להראות כמו

בדוגמה הזו (צורת העקומות עצמן עשויה להשתנות כמובן):



חלק ו' – מימוש בעיית ההובלות (16 נק' יבש)

כעת נרצה לממש את המחלקה שמייצגת את מרחב המצבים של בעיית ההובלות. בבעיה זו נרצה למצוא סדר אופטימאלי להעמסת ופריקת ההובלות תוך התחשבות באילוצי הבעיה כפי שתוארה בחלק ג'.

בשאלות הוכח / הפרך קבילות של היוריסטיקה: אם אתם סבורים שההיוריסטיקה קבילה יש לספק הוכחה לכך. אם אתם סבורים שהיא איננה קבילה יש לספק דוגמא של מרחב חיפוש קטן ככל שתוכלו (ציירו גרף בו הצמתים הם נקודות במפה) עבורו הערך ההיוריסטי על אחד המצבים לפחות גדול ממש מעלות הפתרון האופטימלי למטרה.

13. רטוב: התבוננו בקובץ problems/mda_problem.py והשלימו את המימושים החסרים במתודות הבאות:
    a. MDAState.__eq__()
    b. MDAState.get_total_nr_tests_taken_and_stored_on_ambulance()
    c. MDAProblem.get_reported_apartments_waiting_to_visit()
    d. MDAProblem.get_operator_cost()
    e. MDAProblem.expand_state_with_costs()
    f. MDAProblem.is_goal()

    הערה: המתודה MDAProblem.get_operator_cost() אמורה לחשב את עלות האופרטור שהופעל. כזכור, בחלק ג' ציינו כי בכדי לחשב את עלות האופרטור יש לפתור בעיה על רשת הכבישים. במימוש אנחנו אכן עושים זאת. בהערות בקוד (במתודה *get_operator_cost*()) הורנו לכם להשתמש בשדה (של הבעיה) בשם *map_distance_finder* בו שמור אובייקט מטיפוס ***CachedMapDistanceFinder***, שלו יש מתודה בשם *get_map_cost_between*() המחשבת ומחזירה את עלות פתרון אופטימלי על בעיית מפות הכבישים. מאחורי הקלעים המתודה הזו למעשה אמורה ליצור בעיית MapProblem חדשה ולקרוא ל- AStar.solve_problem() בכדי לפתור אותה. אך לפני זה, לטובת היעילות, היא בודקת האם כבר פתרנו בעיה זו בעבר ואם כן מאתרת את הפתרון שדאגנו לשמור כשפתרנו בעיה זאת לראשונה ומחזירה אותו מיד וללא חישובים נוספים. במובן זה המחלקה CachedMapDistanceFinder שומרת ב- cache הפנימי שלה תוצאות של חישובים קודמים. השלימו את הקוד של המתודה *get_map_cost_between*() של המחלקה CachedMapDistanceFinder בקובץ problems/cached_map_distance_finder.py.

14. רטוב: השלימו את הקוד ב- main.py תחת ההערה הרלוונטית לסעיף זה. הריצו את הקוד הנ"ל (הרצת UniformCost על בעיית ההובלות עם הקלט הקטן). המטרה היא לוודא שהקוד שרשמתם בסעיף הקודם באמת רץ בהצלחה.

15. שאלה יבש (2 נק'): בתכנות לפעמים אנחנו רוצים לכפות על מבני נתונים / טיפוסים מסוימים להיות immutable/frozen. הכוונה היא שאחרי יצירת אובייקט מטיפוס שכזה לא יהיה ניתן לשנותו. הצהרה על טיפוס כ"קפוא" מגבילה אותנו, אך יחד עם זאת היא גם מגינה עלינו. (i) העתק לדו"ח את שורת הקוד הרלוונטית שקובעת שאובייקטים מהטיפוס MDAState יהיו בלתי ניתנים לשינוי. (ii) האם שורה זו מספיקה? מה עוד מבטיח שלא יהיה ניתן לשנות בטעות את האובייקט ו/או את המבנים שהוא מחזיק? (iii) הסבר למה אנחנו רוצים לעשות זאת ספציפית עבור הטיפוס MDAState – תן דוגמא למימוש שגוי של המתודה expand_state_with_costs במחלקה MDAProblem שממחיש את הצורך בטיפוסים "קפואים". טיפ: על הבאג להיגרם מכך שבפיתון משתנה מחזיק בפועל מצביע לאובייקט ולא העתק שלו.

עתה, כדי להריץ את A* על הבעיה, יש ראשית להגדיר (ולממש) היוריסטיקות עבור הבעיה.

16. רטוב: השלימו את המימוש עבור המתודה MDAProblem.get_all_certain_junctions_in_remaining_ambulance_path() (בקובץ problems/mda_problem.py). לאחר מכן, השלימו את המימוש עבור ההיוריסטיקה MDAMaxAirDistHeuristic (בקובץ problems/mda_heuristics.py). היוריסטיקה זו מתבוננת בכל הצמתים (במפת הכבישים) שיש לאמבולנס עוד לעבור בהם (כולל המיקום הנוכחי), ולוקחת את המרחק האווירי הגדול ביותר בין כל זוג מתוך קב' צמתים זו.

17. רטוב: השלימו את הקוד ב- main.py תחת ההערה הרלוונטית לסעיף זה. הריצו את הקוד הנ"ל (הרצת AStar על בעיית ההובלות עם ההיוריסטיקה שמומשה בסעיף הקודם). המטרה היא לוודא שהקוד שרשמתם בסעיף הקודם באמת רץ בהצלחה ולבחון את התוצאות המתקבלות.

18. יבש (4 נק'): הוכח/הפרך: ההיוריסטיקה MDAMaxAirDistHeuristic הינה קבילה (עבור פונק' המחיר $cost_{MDA}^{dist}$). לטובת סעיף זה, הנח שכל הנקודות במפת הכבישים הן נקודות ב- $\mathbb{R}^2$ והמרחק בין זוג נק' הוא המרחק האוקלידי.

19. רטוב: השלימו את המימוש עבור ההיוריסטיקה MDASumAirDistHeuristic (בקובץ problems/mda_heuristics.py). היוריסטיקה זו מתבוננת בכל הצמתים (במפת הכבישים) שיש לאמבולנס עוד לעבור בהן (כולל המיקום הנוכחי), ומחשבת את עלות המסלול הבא: מסלול זה מתחיל בנק' הנוכחית בה נמצא האמבולנס. הנקודה ה- $i+1$ במסלול היא הקרובה ביותר לנק' i במסלול (מבחינת מרחק אווירי) מתוך כל הנק' שנותרו לביקור וטרם נבחרו למסלול זה.
20. רטוב: השלימו את הקוד ב- main.py תחת ההערה הרלוונטית לסעיף זה. הריצו את הקוד הנ"ל (הרצת Astar על בעיית ההובלות עם ההיוריסטיקה שמומשה בסעיף הקודם). המטרה היא לוודא שהקוד שרשמתם בסעיף הקודם באמת רץ בהצלחה ולבחון את התוצאות המתקבלות.
21. יבש (4 נק'): הוכח/הפרך: ההיוריסטיקה MDASumAirDistHeuristic הינה קבילה (עבור פונק' המחיר $cost_d^{dist}$). לטובת סעיף זה, הנח שכל הנקודות במפת הכבישים הן נקודות ב- $\mathbb{R}^2$ והמרחק בין זוג נק' הוא המרחק האוקלידי.
22. רטוב: השלימו את המימוש עבור ההיוריסטיקה MDAMSTAirDistHeuristic (בקובץ problems/mda_heuristics.py). היוריסטיקה זו מתבוננת בכל הצמתים (במפת הכבישים) הנותרים שעל האלבולנס לעבור בהם (מיקומי הדירות שלא עברנו בהן עדיין, כולל המיקום הנוכחי של האמבולנס וללא מיקומי מעבדות נוספות), ובונה גרף שכולל את כל צמתים אלו וקשת בין כל זוג צמתים שמשקלה מוגדר להיות המרחק האווירי בין זוג צמתים אלו. בשלב זה מחושב עץ פורס מינימלי על הגרף הנ"ל. משקל העץ שחושב הוא הערך ההיוריסטי.
23. רטוב: השלימו את הקוד ב- main.py תחת ההערה הרלוונטית לסעיף זה. הריצו את הקוד הנ"ל (הרצת AStar על בעיית ההובלות עם ההיוריסטיקה שמומשה בסעיף הקודם). המטרה היא לוודא שהקוד שרשמתם בסעיף הקודם באמת רץ בהצלחה ולבחון את התוצאות המתקבלות.
24. יבש (4 נק'): הוכח/הפרך: ההיוריסטיקה MDAMSTAirDistHeuristic הינה קבילה (עבור פונק' המחיר $cost_d^{dist}$). לטובת סעיף זה, הנח שכל הנקודות במפת הכבישים הן נקודות ב- $\mathbb{R}^2$ והמרחק בין זוג נק' הוא המרחק האוקלידי.
25. רטוב + יבש (2 נק'): עתה נריץ את wA* עם ערכי w שונים כדי לצייר גרף שמציג את מגמת מחיר הפתרון מגמת מס' הפיתוחים כאשר w משתנה בתחום [0.5,0.95]. לצורך כך נשתמש בפונק' run_astar_for_weights_in_range() שכבר מימשנו בשלבים מוקדמים. השלימו בקובץ main.py את הקוד תחת ההערה הרלוונטית לסעיף זה. צרפו את הגרף שנוצר לדו"ח. הסבירו את הגרף שהתקבל. ציינו באיזה ערך w הייתם בוחרים ולמה.

שימו לב: הסעיפים האחרונים יכולים לעזור לכם לוודא שהאלגוריתמים שלכם אכן עובדים כשורה. ודאו שהתוצאות שקיבלתם <u>הגיוניות</u>.

חלק ז' – מימוש והשוואת פונק' עלות שונות (25.5 נק')

מסתבר שהמקרר באמבולנס אינו אידיאלי עבור אחסון ממושך של הדגימות. ככל שעובר יותר זמן שבו הבדיקות מאוחסנות באמבולנס (ולפני שהן עוברות לאחסון נאות במעבדה), כך יורדת אפקטיביות ואמינות הבדיקה. פונק' העלות $cost_d^{test\ travel}$ (שהוגדרה בחלק ג') מתארת את המדד הנ"ל. בחלק זה ננסה לשלב מדד זה בפתרון הבעיה.

26. יבש (2 נק'): הוכח/הפרך: ההיוריסטיקה MDAMaxAirDistHeuristic הינה קבילה (עבור פונק' המחיר $cost_d^{test\ travel}$). לטובת סעיף זה, הנח שכל הנקודות במפת הכבישים הן נקודות ב- $\mathbb{R}^2$ והמרחק בין זוג נק' הוא המרחק האוקלידי.
27. יבש (2 נק'): הוכח/הפרך: ההיוריסטיקה MDASumAirDistHeuristic הינה קבילה (עבור פונק' המחיר $cost_d^{test\ travel}$). לטובת סעיף זה, הנח שכל הנקודות במפת הכבישים הן נקודות ב- $\mathbb{R}^2$ והמרחק בין זוג נק' הוא המרחק האוקלידי.
28. יבש (2 נק'): הוכח/הפרך: ההיוריסטיקה MDAMSTAirDistHeuristic הינה קבילה (עבור פונק' המחיר $cost_d^{test\ travel}$). לטובת סעיף זה, הנח שכל הנקודות במפת הכבישים הן נקודות ב- $\mathbb{R}^2$ והמרחק בין זוג נק' הוא המרחק האוקלידי.

הערה טכנית לגבי שימוש בפונקציות עלות שונות בקוד: כאשר פותרים את הבעיה יש לקבע פונק' עלות אחת שאיתה עובדים (היא תקבע את עלות האופרטורים והמסלולים). היינו רוצים דרך לקבוע בקוד באיזו פונק' עלות להשתמש עבור בעיית ההובלות. איך זה נעשה? ה- constructor של המחלקה ***MDAProblem*** מקבל פרמטר בשם *optimization_objective* מטיפוס ***MDAOptimizationObjective*** (זהו *enum* שערכיו האפשריים הם *Distance*, *TestsTravelDistance*). העברת הערך בעת יצירת בעיה מגדיר ווריאנט של הבעיה (קובע את פונק' העלות להיות אחת מ- $cost_d^{dist}$, $cost_d^{test\ travel}$). בסעיפים הקודמים כאשר יצרנו בעיית מד"א העברנו לפרמטר *optimization_objective* את הערך *MDAOptimizationObjective.Distance* ובכך הורנו למחלקה *MDAProblem* להשתמש בפונק' העלות $cost_d^{dist}$. ערך זה נשמר באובייקט הבעיה תחת השדה *optimization_objective*. עתה, נוכל להשתמש בערך *MDAOptimizationObjective.TestsTravelDistance*.

29. רטוב: השלימו את המימוש עבור ההיוריסטיקה MDATestsTravelTimeToNearestLabHeuristic (בקובץ problems/mda_heuristics.py) בהתאם להערות המפורטות שם. היוריסטיקה זו מניחה מקרה קיצון שבו נוסעים למעבדה מיד אחרי כל ביצוע של בדיקה.
30. יבש (3 נק'):
    (i) הוכח/הפרך: ההיוריסטיקה *MDATestsTravelTimeToNearestLabHeuristic* הינה קבילה (עבור פונק' המחיר $cost_d^{test\ travel}$). לטובת סעיף זה, הנח שכל הנקודות במפת הכבישים הן נקודות ב- $\mathbb{R}^2$ והמרחק בין זוג נק' הוא המרחק האוקלידי.
    (ii) מה אפשר לומר על היחס בין *MDATestsTravelTimeToNearestLabHeuristic* לבין $h^*_{test\ travel}$?
31. רטוב + יבש (1 נק'): השלימו בקובץ main.py את הקוד תחת ההערה הרלוונטית לסעיף זה. צרפו את התוצאות שקיבלתם לדו"ח (רק את העלויות של שלושת הפתרונות השונים עם ציון של איזו פונק' עלות הייתה בשימוש בכל תוצאה). הדגישו איך רואים בתוצאות שהפתרון המתקבל אכן ממזער את המדד הרלוונטי בהתאם לפונק' העלות שהופעלה.

שילוב בין 2 המדדים

נציג הצעה לשילוב בין 2 המדדים: נניח שעלות הפתרון שממזער את מדד המרחק הינו C^*_{dist}. נקבע ערך $\varepsilon > 0$. פתרון אופטימלי ע"פ <u>*הקריטריון המשולב*</u> הוא פתרון הממזער את המדד *TestsTravelDistance* מבין כל הפתרונות האפשריים שעלות המרחק שלהם שווה/קטנה מ- $(1+\varepsilon)\cdot C^*_{dist}$.

<u>הצגה פורמלית</u>: נניח כי נתון ערך $\varepsilon > 0$ כלשהו ועבורו נגדיר את הבאים:

$$P_{MDA}^{I\to G} \triangleq \left\{\langle s_0,\dots,s_t\rangle \middle| t\in\mathbb{N} \wedge s_0 = I_{MDA} \wedge \forall_{i<t}\, s_i \notin G \wedge s_t \in G_{MDA} \wedge \forall_{i\in\{1,\dots,t\}} \exists_{o\in O_{MDA}} o(s_{i-1}) = s_i \right\}$$

(זהו אוסף **כל המסלולים האפשריים** מהמצב ההתחלתי ועד מצב סופי במרחב S_{MDA})

$$C^*_{dist} \triangleq \min\left\{cost_{MDA}^{dist}(p) \middle| p\in P_{MDA}^{I\to G}\right\}$$

$$DistEpsOptimal \triangleq \left\{p\in P_{MDA}^{I\to G} \middle| cost_{MDA}^{dist}(p) \le (1+\varepsilon)\cdot C^*_{dist}\right\}$$

$$\widetilde{C^*} \triangleq \min\left\{cost_{MDA}^{test\ travel}(p) \middle| p\in DistEpsOptimal\right\}$$

$$OptimalPaths \triangleq \left\{p\in DistEpsOptimal \middle| cost_{MDA}^{test\ travel}(p) = \widetilde{C^*}\right\}$$

הקבוצה $OptimalPaths$ מכילה בדיוק את כל המסלולים שעונים על "הקריטריון המשולב" שהוצג מעלה.

לצורך הסעיף היבש הבא, נגדיר את הפעולה הכללית $\mathcal{P}(\mathcal{S})$ שמקבלת מרחב $\mathcal{S} = \langle S, O, I, G \rangle$ ומגדירה מרחב חדש $\mathcal{P}(\mathcal{S}) \triangleq \langle S^P, O^P, I^P, G^P \rangle$ באופן הבא:

- $S^P \triangleq \left\{\langle s_0, \dots, s_t \rangle \middle| t \in \mathbb{N} \wedge s_0 = I \wedge \forall_{i<t} s_i \notin G \wedge \forall_{i \in \{1,\dots,t\}} \exists_{o \in O} o(s_{i-1}) = s_i \right\}$
- $O^P \triangleq \left\{o_i^P \middle| o_i \in O\right\}$ כאשר $\forall_{p=\langle s_0,\dots,s_t\rangle \in S^P, o_i \in O}\, o_i^P(p) \triangleq \begin{cases} \langle s_0, \dots, s_t, o_i(s_t) \rangle & ; \quad o_i(s_t) \neq \emptyset \\ \emptyset & ; \quad otherwise \end{cases}$
- $I^P \triangleq \langle I \rangle$
- $G^P \triangleq \{\langle s_0, \dots, s_t \rangle \in S^P | s_t \in G\}$

אלג' $\mathcal{A}_1$ מריץ UCS על המרחב $\mathcal{P}(\mathcal{S}_{MDA})$ עם פונק' העלות הבאה:

$$cost\left(\underbrace{\langle s_0, \dots, s_t \rangle}_{p_1}, \underbrace{\langle s_0, \dots, s_t, s_{t+1} \rangle}_{p_2}\right) \triangleq \begin{cases} cost_{MDA}^{test\ travel}(p_2) & ; \quad cost_{MDA}^{dist}(p_2) \leq (1+\varepsilon) \cdot C_{dist}^* \\ \infty & ; \quad otherwise \end{cases}$$

בסעיפים היבשים בחלק זה, הנח שכל הנקודות במפת הכבישים הן נקודות ב- $\mathbb{R}^2$ והמרחק בין זוג נק' הוא המרחק האוקלידי. אם אתם מספקים דוגמא נגדית, היא צריכה להיות קטנה ככל הניתן.

32. יבש (3 נק'): הוכח/הפרך: אם קיים פתרון במרחב, אלג' $\mathcal{A}_1$ בהכרח מחזיר פתרון.
33. יבש (3 נק'): הוכח/הפרך: אם אלג' $\mathcal{A}_1$ מחזיר פתרון אז הפתרון המוחזר בהכרח אופטימלי.

עתה נציע את אלג' $\mathcal{A}_2$ שפועל באופן הבא:

i. הרץ A^* (עם היוריסטיקה קבילה) על המרחב $\mathcal{S}_{MDA}$ עם פונק' העלות $cost_{MDA}^{dist}$.
ii. שמור את עלות הפתרון המוחזר במשתנה C_{dist}^*.
iii. הרץ A^* (עם היוריסטיקה קבילה) על המרחב $\mathcal{S}_{MDA}$ עם פונק' העלות $cost_{MDA}^{test\ travel}$. במהלך הריצה, סכום בצמתי עץ החיפוש גם את העלות $cost_{MDA}^{dist}$ בשדה נפרד. במהלך הריצה, מיד לאחר יצירת צומת חיפוש חדש, הוסף את הבדיקה הבאה: אם העלות *dist* שלו גדולה מ- $(1+\varepsilon) \cdot C_{dist}^*$, מחק את הצומת הזה ואל תוסיף אותו ל- *open*.

34. רטוב + יבש (1.5 נק'): בשלב זה נממש ונריץ את $\mathcal{A}_2$. השלימו בקובץ main.py את הקוד תחת ההערה הרלוונטית לסעיף זה. צרפו את התוצאות שקיבלתם לדו"ח. השוו בטבלה לתוצאות הריצה מסעיפים קודמים (על אותה הבעיה עם שתי פונק' עלות השונות) והראו מספרית שהפתרון המתקבל בסעיף זה אכן מקיים איזון בין שני המדדים. חשבו וצרפו לדו"ח את הערך $\frac{C_{dist}^*}{DistCost(ReturnedSolution)} - 1$.
35. יבש (3 נק'): הוכח/הפרך: אם קיים פתרון במרחב, אלג' $\mathcal{A}_2$ בהכרח מחזיר פתרון.
טיפ: כדי לקבל קצת יותר אינטואיציה, אתם יכולים להריץ את הדוגמא מסעיף קודם עם ערכי ε שונים.
36. יבש (3 נק'): הוכח/הפרך: אם אלג' $\mathcal{A}_2$ מחזיר פתרון אז הפתרון המוחזר בהכרח אופטימלי.
37. יבש (2 נק'): ציין והסבר בקצרה יתרון של $\mathcal{A}_2$ ע"פ $\mathcal{A}_1$ במובנים של זמני ריצה.

חלק ח' – מימוש האלג׳ $A^*\varepsilon$ והרצתו (1.5 נק׳ יבש)

38. רטוב: ממשו את החלקים החסרים של אלג׳ $A^*\varepsilon$ בקובץ framework/graph_search/astar_epsilon.py ע"פ ההנחיות המופיעות שם.
39. רטוב + יבש (1.5 נק׳): מימשנו היוריסטיקה קבילה (MST) והיוריסטיקה לא קבילה אך מיודעת יותר (Sum). הבעיה היא שאין לנו אף הבטחה על איכות הפתרון שמניב *A עם היוריסטיקה שאינה קבילה. נרצה לנצל את הבטחת איכות הפתרון של $A^*\varepsilon$ כדי לעשות שימוש מועיל בהיוריסטיקה שאינה קבילה במטרה לחסוך במספר הפיתוחים מבלי לפגוע באופן דרסטי באיכות הפתרון. השלימו בקובץ main.py את הקוד תחת ההערה הרלוונטית לסעיף זה. צרפו את התוצאות שקיבלתם לדו"ח (אל תצרפו את המסלולים עצמם). האם חסכנו בפיתוחים? אם כן, בכמה? הסבירו למה בכלל ציפינו מראש ש- $A^*\varepsilon$ יוכל לחסוך במס׳ הפיתוחים בתצורה שבה הרצנו אותו.

חלק ט' – מימוש האלג׳ Anytime A* והרצתו (1.5 נק׳ יבש)

בסעיף זה נממש ווריאציה של אלג׳ Anytime A*. האלג׳ יפעל בצורה הבאה: נריץ את אלג׳ wA* על הבעיה על ערכי w שונים. בכל הרצה של wA* נגביל אותו למס׳ פיתוחים קבוע מראש (המחלקה BestFirstSearch והאלג׳ היורשים ממנה יודעים לקבל ב- constructor שלהם פרמטר אופציונלי בשם max_nr_states_to_expand שעוצר את החיפוש לאחר חריגה ממספר פיתוחים זה). נבצע "חיפוש בינארי" על ערכי $w \in [0.5,0.9]$ ונחפש את הפתרון הכי טוב מבין הפתרונות המוגבל במס׳ הפיתוחים כאמור (ושאנו מצליחים למצוא במסגרת שיטה זו). כמו בכל חיפוש בינארי, נתחזק גבול תחתון ועליון במהלך החיפוש. הגבול העליון יאותחל להיות 0.9 והתחתון יהיה 0.5. לאורך החיפוש תישמר האינווריאנטה הבאה: לא נמצא פתרון עבור ערכי w הקטנים או שווים לגבול התחתון (במסגרת הגבלת מס׳ פיתוחים), אך כן נמצא פתרון כזה עבור ערך w של הגבול העליון. בכל איטרציה של החיפוש נריץ את wA* על הבעיה עם ערך w ששווה למחצית הגבול התחתון והעליון ועם מגבלת מס׳ פיתוחים כאמור. נעדכן את הגבולות (בהתאם לקיום או העדר של פתרון) ע"מ לשמור על האינווריאנטה. בכך בכל איטרציה נצמצם את ההפרש בין הגבולות באופן אקספוננציאלי כיאה לחיפוש בינארי. בכל מקרה, נשמור את הפתרון הטוב ביותר שנמצא עד כה ואת הערך w שהוביל איליו. נמשיך כך עד שערכי הגבולות התחתון והעליון יתקרבו זה לזה מספיק.

שימו לב: בכיתה למדתם כלל אצבע לפיו "ככל ש- w קטן יותר כך הפתרון איכותי יותר ומס׳ הפיתוחים גדול יותר". הכלל הנ"ל מצביע על מגמה כללית, אך ציינו בחלקים הקודמים שכלל זה איננו נכון באופן גורף. לכן כשאנו מעדכנים את הגבול התחתון, אין למעשה הבטחה אמתית שעבור כל ערכי w שקטנים מהגבול החדש לא יימצא פתרון העונה על הדרישות. כלומר האלג׳ שלנו לא באמת מוצא ערך w מינימלי שמקיים את האמור, אלא הוא מנסה לקרב אותו ככל הניתן תוך הנחה על המגמה הכללית של הקשר בין w לבין מס׳ הפיתוחים (כלל האצבע).

הערה: ייתכן שהפתרון האופטימלי לאו דווקא הגיע מערך ה- w הקטן ביותר עבורו הרצנו wA* וקיבלנו פתרון. לכן אנו מעדכנים את המשתנה ששומר את הפתרון הטוב ביותר בזהירות (לאחר בדיקה לקיום שיפור באיכות הפתרון).

40. רטוב: השלימו את המימוש של אלג׳ AnytimeA* בקובץ framework/graph_search/anytime_astar.py ע"פ ההוראות המופיעות שם וע"פ ההערות שכתובות בראש המחלקה.
41. רטוב + יבש (1.5 נק׳): השלימו בקובץ main.py את הקוד תחת ההערה הרלוונטית לסעיף זה. צרפו את התוצאות שקיבלתם לדו"ח (אל תצרפו את המסלולים עצמם). הסבירו איך עזר לנו להריץ את הווריאציה הזו של Anytime A* במקרה זה. מה בעצם קיבלנו? שימו לב לגודל הבעיה אותה פתרנו. חזרו לחלק א׳ והיזכרו כמה זמן ייקח למחשב בודד לעבור על כל הסידורים האפשריים.

חלק י' – הגשת המטלה

- **יש לכתוב קוד ברור:**
  - קטעי קוד מסובכים או לא קריאים יש לתעד עם הערות.
  - לתת שמות משמעותיים למשתנים.
- **הדו"ח:**
  - יש לכתוב בדו"ח את תעודות הזהות של **שני** המגישים.
  - הדו"ח צריך להיות מוקלד במחשב ולא בכתב יד. הדו"ח צריך להיות מוגש בפורמט PDF (לא נקבל דוחות שהוגשו בפורמט וורד או אחרים).
  - יש לשמור על סדר וקריאות גם בתוך הדו"ח.
  - אלא אם נכתב אחרת, תשובות ללא נימוק לא יתקבלו.
  - יש לענות על השאלות לפי הסדר ומספרי הסעיפים שלהם.
- **ההגשה:**
  - יש להעלות לאתר קובץ zip בשם AI1_123456789_987654321.zip (עם תעודות הזהות שלכם במקום המספרים).
  - בתוך ה- zip צריכים להיות זה לצד זה:
    - הדו"ח הסופי בפורמט PDF בשם: AI1_123456789_987654321.pdf.
    - תיקיית הקוד ai_hw1 שקיבלתם בתחילת המטלה, עם כל השינויים הנדרשים.
    נא לא להכניס ל- zip את התיקייה db שבתיקייה שקיבלתם – אנא מחקו אותה משם.

שימו לב: הקוד שלכם ייבדק ע"י מערכת בדיקות אוטומטיות תחת מגבלות זמני ריצה. במידה וחלק מהבדיקות יכשלו (או לא יעצרו תוך זמן סביר), הניקוד עבורן יורד באופן אוטומטי. לא תינתן הזדמנות להגשות חוזרות. אנא דאגו לעקוב בהדיקות אחר הוראות ההגשה. שימו לב כי במהלך חלק מהבדיקות ייתכן שחלק מהקבצים שלכם יוחלפו במימושים שלנו. אם עקבתם אחר כל הדגשים שפורטו במסמך זה - עניין זה לא אמור להוות בעיה.

לא תתאפשרנה הגשות חוזרות, גם לא בגלל טעות טכנית קטנה ככל שתהיה. אחריותכם לוודא טרם ההגשה שהתרגיל רץ בסביבה שהגדרנו ושהקוד עומד בכל הדרישות שפירטנו.

אנא עברו בשנית על ההערות הטכניות שפורסמו בתחילת מסמך זה. וודאו שאתם עומדים בהן.

שימו לב: **העתקות תטופלנה בחומרה.** אנא הימנעו מאי-נעימויות.

מקווים שתהנו מהתרגיל!

בהצלחה!