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 שלישי, 01.12.2020, בשעה 23:59.
- את המטלה יש להגיש **בזוגות בלבד**.
- יש להגיש מטלות מוקלדות בלבד. פתרונות בכתב יד לא ייבדקו.
- התשובות צריכות להיות כתובות בשפה העברית.
- 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 (טל סויסה) בלבד.
- במהלך התרגיל ייתכן שנעלה עדכונים, תיקונים והבהרות לדף FAQ ייעודי באתר ולמסמך הנ"ל. העדכונים הינם **מחייבים**, ועליכם להתעדכן עד מועד הגשת התרגיל.
- שימו לב, התרגיל מהווה כ- 30% 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 במקומות בהם התבקשתם לכך. 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 בין היתר 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 בפיתון 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 שמור ה...?" / "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. 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. לכן 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.
- לנוחיותכם, בקוד שסופק לכם הוכנסו type-annotations (ציון של טיפוסים של שדות/פרמטרים). זאת במטרה להקל עליכם בהתמצאות בקוד. אנחנו מצפים מכם להשכיל ולהשתמש ב- IDE (ממליצים על PyCharm) שיוכל לסייע לכם להתמצא בקוד ביתר קלות, יציע לכם השלמת שדות, ויזהה עבורכם שגיאות בצורה סטטית – כל אלו יחסכו לכם הרבה זמן. ה- type-annotations עוזרים ל- IDEs לעזור לכם – נצלו את זה.

- 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 בפיתון 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 וזה כמובן יכול להשפיע באופן ניכר על זמני הריצה. 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 כ- 3 דקות. 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 ככל הנראה תוביל לכישלון מיידי 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 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.
- 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>`.

הוראות עבור שאלות הוכח/הפרך על מרחב MDA בתרגיל

- הניחו שכל הנקודות במפת הכבישים הן נקודות ב- $\mathbb{R}^2$ והמרחק בין זוג נק' הוא המרחק האוקלידי.
- הניחו כי כביש במפת הכבישים הוא בהכרח דו-כיווני. הכביש הינו קו ישר במישור.
- חידוד: עבור מפת כבישים כלשהי נתונה, לא בהכרח קיים כביש בין כל זוג נק' שם. בפרט, עבור זוג נק' כלשהו במפת הכבישים זו, אורך המסלול הקצר ביותר ביניהן אינו בהכרח אורך של קו ישר המחבר ביניהן.
- הפרכות:
  - הפרכה אפשרית אך ורק בעזרת פירוט של **דוגמא נגדית**. תשובות הפרכה ללא מתן דוגמא נגדית קונקרטית ושלמה לא יזוכו בנקודות. בפרט, לא יתקבלו תשובות שינסו לתאר רעיון איך אפשר לבנות דוגמא נגדית או להפריך את הטענה באמצעות שימוש בטענות.
  - דוגמא נגדית היא למעשה **הגדרה מלאה של מפת כבישים + מרחב MDA** (כולל למשל מספר מטושים בכל מעבדה, מספר דיירים בכל דירה, קיבולת האמבולנס, מספר מטושים התחלתי באמבולנס וכו') - ללא נתונים אלו לא נוכל לבדוק לכם את התשובה. אין צורך לספק איור של מרחב MDA כולל האופרטורים שלו. יש לספק איור של מפת הכבישים תוך ציון הפרטים של כל נקודה בה (האם זו דירה/מעבדה/נק' התחלה ואת כל הנתונים שלה). לשם הפשטות, אפשר לבנות מפת כבישים בה כל הנקודות הן נקודות רלוונטיות למרחב MDA עם כבישים ישירים בניהם.
  - אם ישנם קבועים נוספים בשאלה (כמו אפסילון) – אל תשכחו לספק אותם. זכרו שבעת הבדיקה לא נוכל לנסות לנחש נתונים חסרים כדי להשתכנע שהדוגמא שלכם פועלת. תפקידכם הוא לשכנע אותנו בכך.
  - עבור הפרכת קבילות היוריסטיקה: יש להראות כי במרחב הדוגמא שלכם קיים מצב s עבורו מתקיים $h(s) > h^*(s)$. את החישוב של שני ערכים אלו יש להציג באופן מלא ופורמלי. בפרט, אין צורך לספק סימולציה של ריצה של אלג' חיפוש מיודע עבור הפרכת קבילות היוריסטיקה.
  - עבור הפרכות של טענות המתייחסות לאלג' חיפוש מסוים: יש לצרף לדוגמא הנגדית **טבלת מעקב** אחר ריצת האלג' המדובר. זאת הדרך היחידה שתשכנע אותנו (ואתכם) שהדוגמא שלכם אכן מהווה סתירה לטענה. לא נוכל לבדוק את הדוגמא שלכם ללא המעקב. בטבלת המעקב מופיעים שלבי הריצה של האלג' ובכל אחד מפורטים מבני הנתונים של האלג' (open, close) וכל הצמתים המופיעים בהם כולל ערכי f, g, h שלהם.

בטבלה יצוין מיהו הצומת הנשלף מ- open בתחילת כל שלב בריצה. הטבלה תהיה קריאה ברורה ומסודרת ויהיה קל לעקוב אחרי ריצת האלג' הרלוונטי על הדוגמא שלכם מתוך התבוננות בה.

  - שימו לב להבדל בין נקודה על מפת הכבישים לבין מצב במרחב החיפוש עליו האלגוריתם רץ. עבור נקודה מסוימת במפה יכולים להיות מספר מצבים שונים במרחב החיפוש שבהם המיקום הנוכחי הוא אותה הנקודה. הקפידו ליצור הבדל בטבלת המעקב בין **נקודות שמייצגות מצבים שונים**. זה יעזור לכם ולנו להשתכנע שהמעקב תקין.
  - מפת הכבישים צריכה להיות קשירה ובעיית ה- MDA המתוארת צריכה להיות פתירה עבורה אלא אם נאמר אחרת.
  - על המרחב להיות קטן ככל הניתן (מספר **מינימלי** של נקודות במפת הכבישים). טיפ: קודם כל נסו למצוא דוגמא שעובדת ולאחר מכן צמצמו אותה. שימו לב: לא יענו שאלות בסגנון "מה הגודל של הדוגמא המינימלית שקיימת?" / "האם הדוגמא שלי מספיק קטנה או שכדאי לי לחשוב עוד איך לצמצם אותה?"
  - שימו לב: יש להראות שאכן ניתן **למקם את הנקודות במישור** באופן שתיארתם תוך שמירה על המרחקים שציינתם על גבי הקשתות (קיום אי-שוויון המשולש). אם זה לא טריוויאלי ציינו את הקואורדינטות של הנקודות (x,y) במישור.
- הוכחות:
  - הוכחה צריכה להיות פורמלית ומסודרת ואורכה יהיה לכל היותר 7 שורות.
  - הסבר רעיוני/אינטואיטיבי לא יזוכה בנקודות כלל.

אנחנו קשובים לפניות שלכם במהלך התרגיל ומעדכנים את המסמך הזה בהתאם. גרסאות עדכניות של המסמך יועלו לאתר. הבהרות ועדכונים שנוספים אחרי הפרסום הראשוני יסומנו כאן בצהוב. בנוסף, לכל עדכון יהיה מספר גרסה כדי שתוכלו לעקוב. ייתכן שתפורסמנה גרסאות רבות – אל תיבהלו מכך. השינויים בכל גרסה יכולים להיות קטנים.

חלק א' – מבוא והנחיות (3 נק' יבש + 3 נק' בונוס)

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 בקרב האוכלוסיה. 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 מקררים מיוחדים שבהם ניתן לשמור את כל הבדיקות שנלקחו עד כה. המקום במקררים מוגבל, וכאשר כולם מתמלאים מוטי צריך לעבור באחת מהמעבדות האזוריות כדי להעביר להם את הבדיקות ולפנות מקום במקררים לבדיקות הבאות (מוטי גם דואג לכבות את המקררים הריקים כדי לחסוך בצריכת הדלק). 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 כל המקררים שלו ריקים (כלומר אין לו בדיקות להעביר למעבדה). 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 נתוני 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 $AmbulanceTestsCapacity > 0$ בדיקות שהוא יכול לאחסן במקררים.

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

נתונות $m \in \mathbb{N}$ מעבדות $Labs = \{l_1, \dots, l_m\}$. 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 $\left|\{v_0\} \cup \{d_i.loc\}_{i\in[k]} \cup \{l_i.loc\}_{i\in[m]}\right| \equiv k+m+1$.

הבנת קושי הבעיה

בשלב זה אנחנו רוצים לקבל קצת אינטואיציה לגבי רמת הקושי של הבעיה. המטרה היא להשתכנע שאנחנו לא מסוגלים לפתור את הבעיה בעזרת חיפוש brute-force (בגלל מגבלת משאבים). כלומר ביצוע מעבר ממצה על כל המסלולים האפשריים שהאמבולנס יכול לנסוע בהם בזה אחר זה, לכל אחד מהם לבדוק האם הוא חוקי, לחשב את עלותו ולבסוף להחזיר את המסלול האופטימלי שנבחן (ע"פ פונק' עלות כלשהי שנקבעה מראש, כמו למשל מרחק כולל של המסלול).

המטרה שלנו כעת היא לשערך את מספר המסלולים האפשריים ולהשתכנע שהוא גדול מאוד (גם עבור קלטי בעיה k,m קטנים יחסית) ושלא סביר לבצע חיפוש brute-force כמתואר.

מציאת ביטוי מתמטי פשוט שמתאר את מספר המסלולים החוקיים תחת כל אילוצי הבעיה המקוריים כפי שתוארו יכולה להיות בעיה קומבינטורית מאתגרת. לכן לצורך פשטות, נניח לרגע (לטובת חלק זה בלבד) חלק מאילוצי הבעיה (קיבולת מקרר, מטושים, חוקיות מעבר במעבדות). נתרכז אך ורק בכל המסלולים מהסוג הבא (ונניח שכולם חוקיים): עוברים בכל דירה פעם אחת בדיוק, חייבים לסיים במעבדה (אחרי הביקור האחרון בדירה עוברים במעבדה אחת בדיוק), לא חייבים לבקר בכל המעבדות (ייתכן שקיימות מעבדות שלא נעבור בהן), אין הגבלה למספר הפעמים שעוברים במעבדה מסוימת, בתחילת המסלול (לפני

הביקור הראשון בדירה כלשהי) ובין זוג ביקורים רצוף בדירות ניתן לבקר ב- 0 או 1 מעבדות – לא ייתכנו 2 ביקורים רצופים במעבדות.

1. יבש (1 נק'): מצאו ביטוי מתמטי עבור מספר המסלולים האפשריים תחת ההנחות הללו. על הביטוי להיות תלוי בפרמטרים k,m בלבד. אנו מצפים לביטוי מתמטי סגור וללא סכומים (סיגמא). הביטוי צריך להיות כתוב בשפה מתמטית פורמלית ללא שימוש בסימוני עזר וללא תיאורים מילוליים. לא יינתנו נקודות עבור תשובות שאינן מדויקות. תנו כותרת קצרה עבור כל אחד מהמרכיבים הכפליים בביטוי.
2. יבש <u>בונוס</u> (3 נק'): הנחות חדשות: נתרכז אך ורק בכל המסלולים מהסוג הבא (ונניח שכולם חוקיים): עוברים בכל דירה פעם אחת בדיוק, חייבים לסיים במעבדה (אחרי הביקור האחרון בדירה עוברים במעבדה אחת בדיוק), לא חייבים לבקר בכל המעבדות (ייתכן שקיימות מעבדות שלא נעבור בהן), ניתן לבקר במעבדה רק אם (א) טרם עברנו בה או (ב) ביקרנו בדירה ממש לפני הביקור במעבדה זו, בתחילת המסלול (לפני הביקור הראשון בדירה כלשהי) ובין זוג ביקורים רצוף בדירות ניתן לבקר ב- 0 או יותר מעבדות – ייתכן רצף של ביקורים במעבדות בלבד בתנאי שמתקיימים כל התנאים הקודמים. מצאו ביטוי מתמטי עבור מספר המסלולים האפשריים תחת ההנחות הללו. על הביטוי להיות תלוי בפרמטרים k,m בלבד. הביטוי יכול להכיל סכומים (סיגמא) וביטויים קומבינטוריים מוכרים אחרים. הביטוי צריך להיות כתוב בשפה מתמטית פורמלית ללא שימוש בסימוני עזר וללא תיאורים מילוליים. לא יינתנו נקודות עבור תשובות שאינן מדויקות. תנו כותרת קצרה עבור כל אחד מהמרכיבים בביטוי (כפליים/חיבוריים/משתני סכום וכו'). הסבירו בקצרה (עד 3 שורות).
3. יבש (2 נק'): מלאו את הטבלה הבאה. השתמשו בנוסחה שמצאתם בסעיף 1. הזינו את מספר הפרמוטציות האפשריות עבור ערכי k,m המופיעים בטבלה. נניח שמחשב יחיד יכול לבחון $\frac{2^{30}}{100(k+m)}$ מסלולים בשנייה (הסבר הנחה: המחשב יכול לעשות 2^{30} פעולות בסיסיות בשנייה וצריך $100(k+m)$ פעולות בסיסיות ע"מ לבחון מסלול בודד). מלאו בעמודה האחרונה כמה זמן ייקח למחשב כזה לבדוק כל אחד מהמסלולים (לפי היחידות המפורטות).

k	m	$\#possiblePaths$	Estimated calculation time
7	2	~00.00 × 10^00	~00.0 [secs]
7	3	~00.00 × 10^00	~00.0 [mins]
8	3	~00.00 × 10^00	~00.0 [hours]
8	4	~00.00 × 10^00	~00.0 [hours]
9	3	~00.00 × 10^00	~00.0 [days]
10	3	~00.00 × 10^00	~00.0 [months]
11	3	~00.00 × 10^00	~00.0 [years]
12	3	~00.00 × 10^00	~00.0 [thousand years]
12	4	~00.00 × 10^00	~00.0 [thousand years]
13	4	~00.00 × 10^00	~00.0 [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:

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

(הערה: שימו לב להבדל בין $\mathcal{S}_{map}$ שזהו המרחב לבין S_{map} שזוהי קב' המצבים)

- **קבוצת המצבים:**
נרצה לייצג מצב כך שיחזיק את כל המידע שנחוץ לנו עליו במהלך החיפוש במרחב. במקרה המדובר מספיק לשמור את הצומת ברשת הכבישים.
$$S_{map} \triangleq \{s | s.coordinates \in V_{map}\}$$
הסבר: לכל מצב s ב- S_{map} יש שדה בודד בשם $coordinates$ שיכול להיות כל נקודה על רשת הכבישים.
- **קבוצת האופרטורים:**
ניתן לעבור ממצב אחד לעוקבו בתנאי שיש כביש מהצומת המיוצג ע"י המצב הראשון לצומת המיוצג ע"י המצב העוקב.

$$O_{map} \triangleq \left\{ o_{map}^{(s_1,s_2)} \middle| s_1, s_2 \in S_{map} \wedge (s_1.coordinates, s_2.coordinates) \in E_{map} \right\}$$

- **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:
$$cost_{map}^{dist}\left(o_{map}^{(s_1,s_2)}\right) = roadLength\big((s_1.coordinates, s_2.coordinates)\big)$$
- **המצב ההתחלתי:**
$$I_{map} \triangleq (coordinates = v_{src})$$
- **מצבי המטרה:**
$$G_{map} \triangleq \{(coordinates = v_{dst})\}$$

חלק ג' – הגדרת מרחב החיפוש של בעיית מד"א (6 נק' יבש)

בהינתן רשת הכבישים, נקודת המוצא, רשימת המעבדות, נתוני האמבולנס ורשימת הדירות המדווחות, נגדיר מרחב חיפוש עבור בעיית מד"א:

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

(הערה: שימו לב להבדל בין $\mathcal{S}_{MDA}$ שזהו המרחב לבין S_{MDA} שזוהי קב' המצבים)

- **קבוצת המצבים:**

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

- **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 במקררי 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 במקררי האמבולנס עבור אחסון כל הבדיקות מדירה זו, כלומר מתקיים הפרדיקט הבא:

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

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

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

וכן תחום הפעולה המתקבל עבור האופרטור o_{d_i}:

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

<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 (המקררים נותרים ריקים), 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 הפרדיקט הבא:

$$CanVisit(s, l_i) \triangleq \begin{bmatrix} s.Taken \neq \emptyset \\ \vee \\ l_i \notin s.VisitedLabs \end{bmatrix}$$

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

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

הערה: $\mathbb{I}_{[pred]}$ הוא אינדיקטור המקיים $\mathbb{I}_{[pred]} = \begin{cases} 1, & pred\ is\ true \\ 0, & otherwise \end{cases}$

וכן תחום הפעולה המתקבל עבור האופרטור o_{l_i}:

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

<u>לבסוף, קבוצת כל האופרטורים הינה</u>:

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

- **עלות אופרטור:**

במטלה נגדיר 3 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. עלות כספית של הנסיעה:

מורכב ממספר גורמים חיבוריים: (א) עלות הדלק שהאמבולנס צורך במהלך הנסיעה; (ב) עלות קליטת בדיקות במעבדה (בכל ביקור במעבדה בה מעבירים בדיקות למעבדה יש לשלם למעבדה עמלה); (ג) עלות ביקור חוזר במעבדה (בכל ביקור במעבדה, שאינו הביקור הראשון באותה המעבדה יש לשלם עמלה).

$$\begin{aligned} cost_{MDA}^{monetary}(s, o) \triangleq\ & gasPrice \\ & \cdot \big(driveGasConsumption + fridgesGasConsumption(s)\big) \\ & \cdot cost_{MDA}^{dist}(s, o) + \mathbb{I}_{[o(s).curLoc\ is\ a\ Laboratory]} \\ & \cdot \big(\mathbb{I}_{[s.Taken \neq \emptyset]} \cdot labTestTransferCost + \mathbb{I}_{[o(s).curLoc \in s.VisitedLabs]} \\ & \cdot labRevisitCost\big) \end{aligned}$$

הפונק' $fridgesGasConsumption(s)$ קובעת את צריכת הדלק למטר נסיעה הנדרשת עבור פעילות מקררי הדגימות (מחושב לפי מספר הדגימות המאוחסנות על האמבולנס במצב s). היא מוגדרת באופן הבא:

$$fridgesGasConsumption(s) \triangleq \sum_{i=0}^{\#activeFridges(s)-1} fridgeGasConsumption[i]$$

כל אחד מהמקררים יכול להכיל לכל היותר $fridgeCapacity$ דגימות. לכן מספר המקררים הדלוקים הינו:

$$\#activeFridges(s) \triangleq \left\lceil \frac{\sum_{d\in s.Taken} d.roommates}{fridgeCapacity} \right\rceil$$

כאשר $gasPrice$, $driveGasConsumption$, $labTestTransferCost$, $labRevisitCost$, $fridgeGasConsumption[\,]$, $fridgeCapacity$ הם קבועים של הבעיה.

3. 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 משלושת 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- $cost_{MDA}^{monetary}$ וב- $cost_{MDA}^{test\ travel}$.
  - שימו לב: בהינתן אופרטור $o \in O_{MDA}$ ומצב $s \in Domain(o)$, על מנת לחשב את $cost_{MDA}^{dist}(s, o)$ או את $cost_{MDA}^{monetary}$ 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

לטובת הסעיפים בחלק זה הנח שלאו דווקא קיים פתרון ישיג במרחב.

4. יבש (1 נק'): מהם ערכי הקיצון (המקסימלי והמינימלי) האפשריים של דרגת היציאה במרחב החיפוש? ספקו ביטוי מתמטי כפונק' של הפרמטרים k, m של השאלה בלבד. נמקו בקצרה (שורה אחת לכל מקרה).
5. יבש (1 נק'): האם ייתכנו מעגלים במרחב החיפוש שלנו? אם כן תנו דוגמה למעגל כזה, אחרת נמקו. (עד 5 שורות).
6. יבש (1 נק'): כמה מצבים יש במרחב זה (כפי שהוגדר)? האם כולם ישיגים (ציינו כן/לא)? נמקו (עד 3 שורות).
7. יבש (1 נק'): האם ייתכנו בורות ישיגים מהמצב ההתחלתי שאינם מצבי מטרה במרחב החיפוש? אם כן – איך זה ייתכן? אם לא – למה? (נימוק לכל היותר שורה אחת)
8. יבש (1 נק'): מהו טווח האורכים האפשריים של מסלולים במרחב ממצב התחלתי אל מצב סופי? (אורך מסלול = מס' הקשתות) (לכל היותר 7 שורות סה"כ).
9. יבש (1 נק'): הגדירו פורמלית ובצורה ישירה את פונקציית העוקב $Succ_{MDA}: S \rightarrow \mathcal{P}(S)$ המתאימה לבעיה זו (ללא שימוש בקבוצת האופרטורים O).

   שימו לב, אנו מצפים לביטוי מהצורה: $Succ_{MDA}(s) = \{(?, ?, \ldots) | ?\} \cup \{(?, ?, \ldots) | ?\}$

חלק ד' – מתחילים לתכנת

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 בבדיקות.

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 כללים 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 ממשפחת 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 ממשת 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, גם בקוד 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. הבהרה: אין לבצע טעינות נוספות של הקלטים. אנו עשינו זאת בשבילכם בכל המקומות הנדרשים.

תרגילים

10. רטוב: 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. וודאו שאתם מבינים את החלקים השונים. שימו לב שמחלקה זו יורשת מהמחלקה GraphProblem (שתוארה מקודם) וממששת את המתודות האבסטרקטיות הנדרשות.
    f. עתה, לאחר תיקון קוד המחלקה MapProblem, הריצו בשנית את main.py.

חלק ה' – אלגוריתם *A (2 נק' יבש)

עתה נתחיל במימוש *Weighted A.

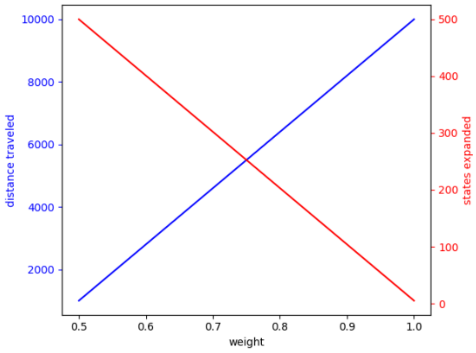
עיינו בקובץ framework/graph_search/astar.py. שם מופיע מימוש חלקי למחלקה AStar. שימו לב: המחלקה AStar יורשת מהמחלקה האבסטרקטית BestFirstSearch (הסברנו עליה בחלק ד'). זהו את החלק בהצהרת המחלקה AStar בו הירושה מוגדרת. המחלקה AStar צריכה לממש את המתודות האבסטרקטיות שמוגדרות ע"י BestFirstSearch. הכותרות של 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

11. 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.
12. 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.
13. רטוב: 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).
    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.
14. יבש (1 נק'): כתוב בדו"ח את מס' פיתוחי המצבים ה**יחסי** שחסכנו בריצה בסעיף קודם לעומת הריצה העיוורת (ההפרש חלקי מס' הפיתוחים בריצה **בלי** ההיוריסטיקה).
15. רטוב: 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.
16. יבש (1 נק'): צרפו לדו"ח את הגרף שנוצר בריצה מהסעיף הקודם. הסבירו את הגרף שהתקבל. ציינו אילו איזורים בגרף הם יותר כדאיים ואילו פחות – ציינו למה (עד 2 שורות). 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? (תשובה עד 4 שורות). על התרשים להראות כמו בדוגמה הזו (צורת העקומות עצמן עשויה להשתנות כמובן):



חלק ו' – מימוש בעיית מד"א (15 נק' יבש)

כעת נרצה לממש את המחלקה שמייצגת את מרחב המצבים של בעיית מד"א. בבעיה זו נרצה למצוא סדר אופטימאלי למעבר של האמבולנס בדירות המדווחות (לצורך לקיחת בדיקות) והעברת הבדיקות למעבדות 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 במפת הכבישים) עבורו הערך ההיוריסטי על אחד המצבים לפחות גדול ממש מעלות הפתרון האופטימלי למטרה.

17. 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 מאחרת 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.
18. רטוב: השלימו את הקוד ב- main.py תחת ההערה הרלוונטית לסעיף זה. הריצו את הקוד הנ"ל (הרצת UniformCost על בעיית מד"א עם הקלט הקטן). המטרה היא לוודא שהמימוש שלכם מהסעיף הקודם באמת רץ בהצלחה.
19. שאלה יבש (2 נק'): בחלק ב' הגדרנו את מרחב מפת הכבישים, ובחלק ג' הגדרנו את מרחב מד"א. נסתכל על זוג של מצב s ועוקב כלשהו שלו $o(s)$ במרחב מד"א. שימו לב כי ייתכן שלא קיימת קשת ישירה בין זוג הצמתים $s.curLoc$ ל- $o(s).curLoc$ במרחב מפות הכבישים. עם זאת, קיים מסלול כלשהו ביניהם במרחב המפה. לכן, כל פעם שצריך לחשב את עלות האופרטור $cost_{MDA}^{dist}(s,o)$ (כפי שהוגדר לעיל בחלק ג'), למעשה פותרים "בעיית ביניים" במרחב מפות הכבישים $\langle S_{map}, O_{map}, I_{map} = s.curLoc, G_{map} = \{o(s).curLoc\}\rangle$.

    עקרונית, היה אפשר לשלב את שני המרחבים המדוברים למרחב-על אחד. במרחב-העל הנ"ל עבור מצב s כלשהו, המיקום שלו $s.curLoc$ היה יכול להיות כל נקודה ברשת הכבישים. בנוסף, היינו מוסיפים אופרטורים שמאפשרים מעבר למצב עוקב שבו רק המיקום של האמבולנס היה משתנה (לאחת מהנקודות $Succ_{map}(s.curLoc)$) על רשת הכבישים).

    שאלה: מה יכול להיות חסרון בגישה שכזאת מבחינת יעילות הפתרון? על תשובתכם להתייחס לטכניקה ספציפית שהשתמשנו בה במימוש. תשובה עד 3 שורות.
20. שאלה יבש (3 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 בקוד מבטיח שלא יהיה ניתן לשנות בטעות את האובייקט ו/או את המבנים שהוא מחזיק?

    (iii) האם ייתכן באלג' *A שנפגוש מצב בשנית גם לאחר שפיתחנו אותו? (אם כן – ציין את השורה

באלג׳ A* מההרצאה שבה זה קורה).
(iv) 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״. על הבאג להיגרם מכך שבפיתון משתנה מחזיק בפועל מצביע לאובייקט ולא העתק שלו. יש להשתמש בתשובה מהסעיף הקודם כדי לענות על סעיף זה. הבאג צריך להיות שגיאה תמימה של מתכנת שלא רגיל לשפות תכנות בהן מתבצע copy-by-reference. תשובה ל- (iv) עד 5 שורות.

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

21. 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.
22. רטוב: השלימו את הקוד ב- main.py תחת ההערה הרלוונטית לסעיף זה. הריצו את הקוד הנ״ל (הרצת AStar על בעיית מד״א עם ההיוריסטיקה שמומשה בסעיף הקודם). המטרה היא לוודא שהקוד שרשמתם בסעיף הקודם באמת רץ בהצלחה ולבחון את התוצאות המתקבלות.
23. יבש (4 נק׳): הוכח/הפרך: ההיוריסטיקה MDAMaxAirDistHeuristic הינה קבילה (עבור פונק׳ המחיר $cost_{MDA}^{dist}$). ראה בעמוד השני במסמך את ההערות המתייחסות לשאלות הוכח/הפרך.
24. 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.
25. רטוב: השלימו את הקוד ב- main.py תחת ההערה הרלוונטית לסעיף זה. הריצו את הקוד הנ״ל (הרצת Astar על בעיית מד״א עם ההיוריסטיקה שמומשה בסעיף הקודם). המטרה היא לוודא שהקוד שרשמתם בסעיף הקודם באמת רץ בהצלחה ולבחון את התוצאות המתקבלות.
26. יבש (4 נק׳): הוכח/הפרך: ההיוריסטיקה MDASumAirDistHeuristic הינה קבילה (עבור פונק׳ המחיר $cost_{MDA}^{dist}$). ראה בעמוד השני במסמך את ההערות המתייחסות לשאלות הוכח/הפרך.
27. 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.
28. רטוב: השלימו את הקוד ב- main.py תחת ההערה הרלוונטית לסעיף זה. הריצו את הקוד הנ״ל (הרצת AStar על בעיית מד״א עם ההיוריסטיקה שמומשה בסעיף הקודם). המטרה היא לוודא שהקוד שרשמתם בסעיף הקודם באמת רץ בהצלחה ולבחון את התוצאות המתקבלות.
29. יבש (4 נק׳): הוכח/הפרך: ההיוריסטיקה MDAMSTAirDistHeuristic הינה קבילה (עבור פונק׳ המחיר $cost_{MDA}^{dist}$). ראה בעמוד השני במסמך את ההערות המתייחסות לשאלות הוכח/הפרך.
30. רטוב + יבש (1 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. ציינו אילו איזורים בגרף הם יותר כדאיים ואילו פחות.

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 (21 נק' יבש)

במד"א רוצים לאפטם את עלות יום העבודה של האמבולנס. העלות כוללת כמה מרכיבים: (א) הוצאות הדלק עבור נסיעת האמבולנס; (ב) הוצאות דלק עבור הפעלת הגנרטורים באמבולנס שאחראים לספק חשמל למקררים של הדגימות; (ג) עמלה שגובה המעבדה עבור כל ביקור במעבדה בו מבוצעת פריקה של דגימות מהאמבולנס; ו- (ד) עמלה שגובה המעבדה על כניסה חוזרת של האמבולנס לשטחי המעבדה (המעבדה רוצה לעודד מינימום תנועה בשטחה ולכן גובה עמלה החל מהביקור השני של האמבולנס בשטחה). למעשה, כבר מימשתם בחלקים קודמים את פונק' העלות $cost_{MDA}^{monetary}$ המחשבת עלות זו.

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 $cost_{MDA}^{test\ travel}$ (שהוגדרה בחלק ג') מתארת את המדד הנ"ל.

בחלק זה נשלב את שתי העלויות הנוספות בפתרון בעיית מד"א.

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 מד"א. איך זה נעשה? ה- constructor של המחלקה ***MDAProblem*** מקבל פרמטר בשם *optimization_objective* מטיפוס ***MDAOptimizationObjective*** (זהו *enum* שערכיו האפשריים הם *Distance*, *Monetary*, *TestsTravelDistance*). העברת הערך בעת יצירת בעיה מגדיר וריאנט של הבעיה (קובע את פונק' העלות להיות אחת מ- $cost_{MDA}^{dist}$, $cost_{MDA}^{monetary}$, $cost_{MDA}^{test\ travel}$). בסעיפים הקודמים כאשר יצרנו בעיית מד"א העברנו לפרמטר *optimization_objective* את הערך *MDAOptimizationObjective.Distance* ובכך הורנו למחלקה *MDAProblem* להשתמש בפונק' העלות $cost_{MDA}^{dist}$. ערך זה נשמר באובייקט הבעיה תחת השדה *optimization_objective*. עתה, נוכל להשתמש בערך *MDAOptimizationObjective.Monetary* או בערך *MDAOptimizationObjective.TestsTravelDistance*.

31. יבש (2 נק'): סמן בכל אחד מהתאים כן/לא. האם כל אחת מההיוריסטיקות הנקובות הינה קבילה ביחס לפונק' המחיר הנקובה? (אין צורך בנימוק).

	MDAMaxAirDistHeuristic	MDASumAirDistHeuristic	MDAMSTAirDistHeuristic
$cost_{MDA}^{test\ travel}$			
$cost_{MDA}^{monetary}$			

32. רטוב + יבש (0.5 נק' יבש): כעת נפתור את הבעיה עם פונק' העלות $cost_{MDA}^{monetary}$. השלימו בקובץ main.py את הקוד תחת ההערה הרלוונטית לסעיף זה. השוו כאן בדו"ח את התוצאות עם תוצאות מסעיפים קודמים של פתרון בעיה זו עם מדד המרחק כ- optimization objective. הראו בדו"ח איך רואים בתוצאות שהפתרון המתקבל אכן ממזער את המדד הרלוונטי בהתאם לפונק' העלות שהופעלה (אין צורך לצרף את כל הפלט עם המסלול, רק את העלויות).
33. 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.
34. יבש (2 נק'): הוכח/הפרך: ההיוריסטיקה *MDATestsTravelTimeToNearestLabHeuristic* הינה קבילה עבור פונק' המחיר $cost_{MDA}^{test\ travel}$. ראה בעמוד השני במסמך את ההערות המתייחסות לשאלות הוכח/הפרך.
35. רטוב + יבש (0.5 נק' יבש): כעת נפתור את הבעיה עם פונק' העלות $cost_{MDA}^{test\ travel}$. השלימו בקובץ main.py את הקוד תחת ההערה הרלוונטית לסעיף זה. השוו כאן בדו"ח את התוצאות עם תוצאות מסעיפים קודמים של פתרון בעיה זו עם מדד המרחק כ- optimization objective. הראו בדו"ח איך רואים בתוצאות שהפתרון המתקבל אכן ממזער את המדד הרלוונטי בהתאם לפונק' העלות שהופעלה (אין צורך לצרף את כל הפלט עם המסלול, רק את העלויות).

שילוב בין 2 מדדים

ראינו שניתן להתעניין במספר מדדים שונים עבור איכות פתרון. ראינו שכל מדד מביא פתרון אחר לבעיה המאפטם אותו. לפעמים בחיים אנחנו מעוניינים למצוא פתרון שמתחשב במספר מדדים שונים. בחלק זה נציג הצעה לשילוב בין 2 מדדים ונבחן 2 אפשרויות לממש שילוב שכזה.

נציג הצעה לשילוב בין 2 המדדים $cost_{MDA}^{dist}$ ו- $cost_{MDA}^{test\ travel}$: נניח שעלות פתרון הממזער 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:

$$P_{MDA}^{I\to G} \triangleq \left\{ \langle s_0 \xrightarrow{o_0} s_1 \xrightarrow{o_1} \dots \xrightarrow{o_{t-1}} s_t \rangle \middle| \begin{matrix} t\in\mathbb{N} \wedge s_0 = I_{MDA} \wedge \forall_{i<t}\, s_i \notin G \wedge s_t \in G_{MDA} \wedge \\ \forall_{i\in\{0,\dots,t-1\}} o_i \in O_{MDA} \wedge \forall_{i\in\{1,\dots,t\}} o_{i-1}(s_{i-1}) = s_i \end{matrix} \right\}$$

(זהו אוסף **כל המסלולים האפשריים** מהמצב ההתחלתי ועד מצב סופי במרחב $\mathcal{S}_{MDA}$)

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

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

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

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

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

לצורך תיאור האלג' הבא, נגדיר את הפעולה הכללית $\mathcal{P}(\mathcal{S})$ שמקבלת מרחב $\mathcal{S} = \langle S, O, I, G\rangle$ ומגדירה את "מרחב המסלולים" התואם $\mathcal{P}(\mathcal{S}) \triangleq \langle S^P, O^P, I^P, G^P\rangle$ באופן הבא:

- $S^P \triangleq \left\{ \langle s_0 \xrightarrow{o_0} s_1 \xrightarrow{o_1} \dots \xrightarrow{o_{t-1}} s_t \rangle \middle| \begin{matrix} t\in\mathbb{N} \wedge s_0 = I \wedge \forall_{i<t} s_i \notin G \wedge \\ \forall_{i\in\{0,\dots,t-1\}} o_i \in O \wedge \forall_{i\in\{1,\dots,t\}} o_{i-1}(s_{i-1}) = s_i \end{matrix} \right\}$
- $O^P \triangleq \{o^P | o\in O\}$ כאשר:

$$\forall_{p=\langle s_0 \xrightarrow{o_0} s_1 \xrightarrow{o_1} \dots \xrightarrow{o_{t-1}} s_t\rangle \in S^P, o\in O}\ o^P(p) \triangleq \begin{cases} \langle s_0 \xrightarrow{o_0} s_1 \xrightarrow{o_1} \dots \xrightarrow{o_{t-1}} s_t \xrightarrow{o} o(s_t)\rangle & ;\quad o(s_t) \neq \emptyset \\ \emptyset & ;\quad otherwise \end{cases}$$

- $I^P \triangleq \langle I\rangle$
- $G^P \triangleq \left\{ \langle s_0 \xrightarrow{o_0} s_1 \xrightarrow{o_1} \dots \xrightarrow{o_{t-1}} s_t \rangle \in S^P \middle| s_t \in G \right\}$

אלג' $\mathcal{A}_1$ מבצע:

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

$$\forall_{\substack{p\in S_{MDA}^P,\ o\in O_{MDA} \\ s.t.\ p\in Domain(o^P)}} : cost\left(\underbrace{\langle s_0 \xrightarrow{o_0} s_1 \xrightarrow{o_1} \dots \xrightarrow{o_{t-1}} s_t \rangle}_{p}, o^P \right) \triangleq \begin{cases} cost_{MDA}^{test\ travel}(s_t, o) & ;\quad cost_{MDA}^{dist}(s_t,o) + \sum_{i=0}^{t-1} cost_{MDA}^{dist}(s_i,o_i) \le (1+\varepsilon)\cdot C_{dist}^* \\ \infty & ;\quad otherwise \end{cases}$$

הבהרה טכנית: אם בסיום ריצת $\mathcal{A}_1$ נמצא פתרון, מוחזר המצב הסופי (במרחב $\mathcal{P}(\mathcal{S})$), שהינו למעשה מסלול – סדרה של מצבים ואופרטורים במרחב המקורי $\mathcal{S}_{MDA}$ (באותה התצורה שמוחזר מסלול ע"י ריצת A^* על המרחב המקורי $\mathcal{S}_{MDA}$).

הבהרה טכנית: אם פונק' המחיר מקבלת ערך אינסופי על מצב מסוים, אז הפעלת האופרטור לא חוקית על המצב הנתון (והאלג' לא מוסיף ל- open).

36. יבש (3 נק'): הוכח/הפרך: אם קיים פתרון במרחב המקורי $\mathcal{S}_{MDA}$, אלג' $\mathcal{A}_1$ בהכרח מחזיר פתרון. ראה בעמוד השני במסמך את ההערות המתייחסות לשאלות הוכח/הפרך.
37. יבש (3 נק'): הוכח/הפרך: אם אלג' $\mathcal{A}_1$ מחזיר פתרון אז הפתרון המוחזר בהכרח אופטימלי ע"פ **הקריטריון המשולב** שהוגדר מעלה. ראה בעמוד השני במסמך את ההערות המתייחסות לשאלות הוכח/הפרך.

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 UCS 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*.

38. רטוב + יבש (0.5 נק'): בשלב זה נממש ונריץ וריאציה של $\mathcal{A}_2$ (השינוי הוא שבמימוש נשתמש ב- A^* עם היוריסטיקה קבילה במקום ב- UCS). השלימו בקובץ main.py את הקוד תחת ההערה הרלוונטית לסעיף זה. צרפו את התוצאות שקיבלתם לדו"ח (אין צורך במסלולים – מספיק עלויות הפתרון). השוו בטבלה לתוצאות הריצה מסעיפים קודמים (על אותה הבעיה עם שתי פונק' עלות השונות) ובדקו מספרית האם הפתרון המתקבל בסעיף זה אכן מקיים איזון בין שני המדדים. חשבו וצרפו לדו"ח את הערך $\frac{DistCost(ReturnedSolution)}{C_{dist}^{*}}-1$. האם אכן נשמר ערך ה- ε הנקוב?
39. יבש (4 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. ראה בעמוד השני במסמך את ההערות המתייחסות לשאלות הוכח/הפרך (הטיפ כאן ניתן רק ככלי עזר לפיתוח האינטואיציה. יש לספק הוכחה/הפרכה פורמלית ומלאה לפי ההוראות וללא התייחסות לתוצאות ריצה כזו או אחרת).
40. יבש (4 נק'): הוכח/הפרך: אם אלג' $\mathcal{A}_2$ מחזיר פתרון אז הפתרון המוחזר בהכרח אופטימלי ע"פ **הקריטריון המשולב** שהוגדר מעלה. ראה בעמוד השני במסמך את ההערות המתייחסות לשאלות הוכח/הפרך.
41. יבש (1.5 נק'): ציין והסבר בקצרה יתרון צפוי של $\mathcal{A}_2$ ע"פ $\mathcal{A}_1$ במובנים של זמני ריצה. התייחס בתשובתך ליחסי הגדלים בין שני המרחבים (עליהם שני האלג' רצים). תשובה עד 3 שורות.

חלק ח' – מימוש האלג' $A^*\varepsilon$ והרצתו (1 נק' יבש)

42. רטוב: ממשו את החלקים החסרים של אלג' $A^*\varepsilon$ בקובץ framework/graph_search/astar_epsilon.py ע"פ ההנחיות המופיעות שם.
43. רטוב: 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.
44. יבש (1 נק'): צרפו לדו"ח את התוצאות שקיבלתם בסעיף הקודם (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. לא מספיק לטעון ש- $A^*\varepsilon$ גמיש יותר בבחירה של הצומת הבא לפיתוח. נסו להסביר למה בעצם אנחנו מצפים שהגמישות הזאת של $A^*\varepsilon$ אכן תעזור לנו במקרה הזה לבחור מ- open צומת לפיתוח שיקדם אותנו מהר יותר למטרה. מה בעצם הוספנו לאלג' החיפוש? תשובה עד 2 שורות.

חלק ט' – מימוש האלג' Anytime A* והרצתו

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 אמיתית 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).

45. 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.
46. רטוב: השלימו בקובץ main.py את הקוד תחת ההערה הרלוונטית לסעיף זה.

חלק י' – שאלה תאורטית (12 נק' יבש)

סעיף (א) – 1 נק' יבש

כזכור, בכיתה הצגנו את אלגוריתם A^* שהינו שלם וקביל. לאחר מכן, הצגנו את אלג' IDA* שמטרתו הייתה לשפר מדד ביצועי כלשהו של אלג' A^*. ציין במילה **אחת** מהו אותו מדד ביצועי עבורו אלג' IDA* עדיף תמיד על פני אלג' A*. הסבר (עד 2 שורות).

סעיף (ב) – 5 נק' יבש

(i) (1 נק' יבש) באיזה מדד ביצועי אלג' IDA* עלול להיות משמעותית פחות טוב מאשר אלג' A^* במקרים רבים? תשובה עד 2 מילים.

(ii) (2 נק' יבש) למה מדד זה נפגע ב- IDA* (לעומת A^*)? תשובה עד שורה אחת.

(iii) (2 נק' יבש) האם מדד זה נפגע באותו האופן כמו שהוא נפגע ב- ID-DFS לעומת BFS? אם כן, למה? אם לא, מה ההבדל? תשובה עד 3 שורות.

בהמשך השאלה נבחן וריאציה לאלג' IDA* שמטרתה להתמודד עם הבעיה עליה נשאלתם בסעיף ב', תוך הקרבה של איכות הפתרון.

נתונים:

- נתון עבור מרחב חיפוש $\mathcal{S}=(S,I,O,G)$ בעל מקדם סיעוף חסום (כרגיל).
- נסמן ב- $\{(s,o(s))|s\in S,o\in O\}\triangleq E_{\mathcal{S}}$ את אוסף הקשתות בגרף המצבים של $\mathcal{S}$.
- נתונה עבור $\mathcal{S}$ פונק' עלות $w:E_{\mathcal{S}}\mapsto\mathbb{R}^+$ והיוריסטיקה $h:S\mapsto\mathbb{R}^+\cup\{0\}$ **קבילה**.
- נתון כי עלות פתרון אופטימלי במרחב $\mathcal{S}$ הינו $C^*_{\mathcal{S}}$.
- נתון קבוע $k\in\mathbb{N}^+$ עבורו מתקיים $\forall_{(s_1,s_2)\in E_{\mathcal{S}}}:w\big((s_1,s_2)\big)\geq 1/k$.

הגדרות נוספות:

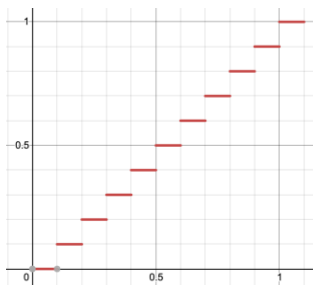
נגדיר את הקבוצה הבאה: $A_k\triangleq\left\{\frac{l}{k}\middle|l\in\mathbb{N}\right\}$. בנוסף נגדיר פונק' $Q_k:\mathbb{R}^+\cup\{0\}\mapsto A_k$ באופן הבא:

$$\forall_{x\in\mathbb{R}^+\cup\{0\}}:Q_k(x)\triangleq\frac{l}{k}\quad s.t.\quad l\in\mathbb{N}\wedge x\in\left[\frac{l}{k},\frac{l+1}{k}\right)$$

ניסוח שקול אחר:

$$\forall_{x\in\mathbb{R}}:Q_k(x)=\max\{a|a\in A_k\wedge a\leq x\}$$

לדוגמא, עבור $k=10$ מקבלים את הקב' $A_{10}=\{0,0.1,0.2,0.3,\dots,0.9,1.0,1.1,1.2,\dots\}$. להלן תרשים המתאר את הפונק' Q_{10} בקטע $x\in[0,1]$:



בהינתן אלג' חיפוש $\mathcal{A}$ ומרחב $\mathcal{S}$, נגדיר את $Cost(\mathcal{A}(\mathcal{S}))$ להיות עלות הפתרון שמוחזר ע"י האלג' $\mathcal{A}$ בריצתו על המרחב $\mathcal{S}$. כמו כן, נגדיר: $\varepsilon(\mathcal{A},\mathcal{S})\triangleq\left|C^*_{\mathcal{S}}-Cost(\mathcal{A}(\mathcal{S}))\right|$.

סעיף (ג) – 6 נק' יבש

אלג' $\mathcal{A}_1$ דומה ל- IDA* (הרגיל), עם השינויים הבאים:

(א) משנים את ערך ה- f-limit ההתחלתי להיות f-limit := $Q_k(h(I))$.

(ב) משנים את כלל העדכון של f-limit באופן הבא:

$$nextFLimit \coloneqq \max\left\{prevFLimit + \frac{1}{k}, Q_k(origNextFLimit)\right\}$$

ניסוח אלטרנטיבי שקול:

$$nextFLimit = \begin{cases} Q_k(origNextFLimit) & ; \quad Q_k(origNextFLimit) \neq prevFLimit \\ prevFLimit + \frac{1}{k} & ; \quad o.w. \end{cases}$$

כאשר $origNextFLimit$ הינו כלל העדכון של f-limit באלג' *IDA המקורי. כלומר זהו הערך שאלג' *IDA המקורי היה בוחר בתור ערך ה- f-limit הבא בתום האיטרציה האחרונה שבוצעה ע"י $\mathcal{A}_1$.

שימו לב, יש לספק ביטויים מתמטיים סגורים התלויים בקבועים המוגדרים בשאלה בלבד. בפרט, אין להגדיר קבועים אחרים שאינם מופיעים בגוף השאלה.

(i) (3 נק') כמה איטרציות לכל היותר יבצע $\mathcal{A}_1$ על $\mathcal{S}$? הסבר (לכל היותר 3 שורות).
(ii) (3 נק') ספק חסם עליון הדוק עבור $\varepsilon(\mathcal{A}_1, \mathcal{S})$. הסבר (לכל היותר 3 שורות).

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 הטכניון שפורסמו בתחילת מסמך זה. וודאו שאתם עומדים בהם.

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

מקווים שתהנו מהתרגיל!

בהצלחה!