תרגיל בית 1:

Blind Search

מטרות התרגיל

- התנסות בְּייצוג בעיות "מהעולם האמיתי" כמרחבי מצבים.
- התנסות בהפעלת אלגוריתמי חיפוש **לא מיודעים** לפתרון בעיות.
- מימוש וריאציה לאלגוריתמים קיימים בהתאם לדרישות הבעיה.

הערות

- תאריך הגשה: 27.11.16.
- את המטלה יש להגיש **בזוגות בלבד!**
- שאלות בנוגע לתרגיל יש לשלוח לניצן: csnussa@gmail.com
- בקשות **מוצדקות** לדחייה יש לשלוח לנעם, המתרגלת האחראית: noamrvd@cs.technion.ac.il
- קִראו היטב את ההסברים וההוראות במסמך זה, מטרתם לסייע לכם בהבנת הדרישות של התרגיל.
- הִתְעַדְכְּנו ברשימת ה-FAQ באתר הקורס בתדירות גבוהה, לפני פנייה בשאלות דרך המייל ולפני הגשת התרגיל. ההערות שתתפרסמנה באתר הקורס מחייבות את כלל הסטודנטים בקורס!
- הקוד שלכם ייבדק על-ידי $Unit-Tests$, לכן עליכם לעקוב בתשומת לב רבה אחר הוראות ההגשה המצורפות במהלך התרגיל ובסופו לפני הגשתו.
- בתרגיל זה, כמו גם בתרגילים הבאים בקורס, הרצת הניסויים עשוייה לקחת זמן רב ולכן מומלץ מאוד להמנע מדחיית העבודה על התרגיל לרגע האחרון. **לא תינתנה דחיות על רקע זה.**

חלק א'- מבוא והנחיות

במטלה זו נעסוק בהפעלת אלגוריתמי חיפוש **לא מיודעים** על בעיות ניווט. מומלץ לחזור על שקפי ההרצאות והתרגולים הרלוונטיים לפני תחילת העבודה על התרגיל.

במהלך התרגיל תתבקשו להריץ מספר ניסויים ולנתח את תוצאותיהם. אנא בצעו **ניתוח מעמיק ומפורט** של התוצאות וצרפו אותו לדו"ח כפי שיוסבר בהמשך התרגיל.

1. מוטיבציה:

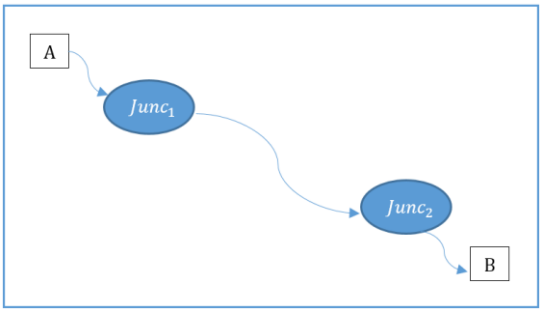
תוכנת $Waze$ מוצאת את המסלול בעל <u>זמן הנסיעה הצפוי הקצר ביותר</u>, כאשר זמן הנסיעה בכל קטע משוערך על ידי המהירות הממוצעת העכשווית באותו קטע.

תוכנת $\boldsymbol{BetterWaze}$ אותה תכתבו תנסה לקצר את **זמן החישוב** הדרוש למציאת מסלול בין שני מיקומים (במחיר אפשרי של בחירת מסלול לא אופטימלי). לצורך כך נשאב השראה מאופן פעולתם של נהגי המוניות (שיתואר בהמשך). כמו כן, על מנת להקל על עבודתכם, נניח בתרגיל זה כי מסלול אופטימלי הינו מסלול בעל **<u>מרחק נסיעה</u>** קצר ביותר (בניגוד ל**<u>זמן נסיעה</u>** קצר ביותר).

כידוע, במערכת הכבישים בערים ישנם צמתים מרכזיים דרכם ניתן (וכדאי) לעבור במהלך נסיעה בין מקומות שונים בעיר. נהגי מוניות, עקב נסיונם הרב בנסיעה עירונית, מכירים היטב את הצמתים המרכזיים, ויודעים מהי דרך הנסיעה המועדפת עליהם בין כל זוג צמתים מרכזיים קרובים כנ"ל (לצורך העניין, כל בוקר נהגי המוניות משננים את רשימת הצמתים המרכזיים והמסלולים בין זוגות צמתים קרובים, כך שבמהלך יום עבודה הם יודעים לנווט בין הצמתים האלו מזכרונם בלבד).

נהג מונית מסיע נוסע מנקודה A לנקודה B בעיר. שתי הנקודות A ו-B הן מקומות לא מרכזיים ושכוחי אל שהנהג לא מכיר, אך כאמור הוא מכיר היטב את כל הצמתים <u>המרכזיים</u> בעיר. על הנהג להחליט **בְּמהירות** באיזה מסלול נסיעה עליו לִסוע (אחרת הנוסע ירד מהמונית בכעס).

לשם כך הנהג:



1. מברר מהו הצומת המרכזי האופטימלי $Junc_1$ עבור נסיעה מהנקודה A (כלומר הצומת המרכזי ש**<u>מרחק הנסיעה</u>** מ-A אליו הוא הקצר ביותר). את $Junc_1$ הנהג מכיר היטב.
2. מברר מהי הדרך הטובה ביותר לִסוע מהנקודה A לצומת המרכזי $Junc_1$ (כלומר הדרך ש**<u>מרחק הנסיעה</u>** בה הוא הקצר ביותר).
3. מברר מהו הצומת המרכזי האופטימלי $Junc_2$ עבור נסיעה אל הנקודה B (כלומר הצומת המרכזי ש**<u>מרחק הנסיעה</u>** ממנו אל B הוא הקצר ביותר). גם את $Junc_2$ הנהג מכיר היטב.

4. מחליט, על סמך נסיונו הרב בנהיגה בין צמתים מרכזיים, מהי הדרך הטובה ביותר לִסוע מ-$Junc_1$ אל $Junc_2$.
5. מברר מהי הדרך הטובה ביותר לִסוע מהצומת המרכזי $Junc_2$ אל הנקודה B (כלומר הדרך ש**מרחק הנסיעה** בה הוא מינימלי).
6. נוסע מהנקודה A אל הנקודה B דרך הצמתים $Junc_1$ ו-$Junc_2$ לפי המסלולים שנמצאו בשלבים (2), (4) ו-(5).

שימו לב כי שלבים (2) ו-(5) נותנים לנהג **תת-מסלול** אופטימלי בין נקודות המוצא והיעד **שלהם**, בעוד שלב (4) מתבצע על סמך הערכה כללית של הנהג עצמו (ולכן לא בהכרח מחזיר מסלול אופטימלי). בנוסף, ייתכן כי המסלול האופטימלי בין הנקודה A לנקודה B כלל לא עובר בצמתים $Junc_1$ ו-$Junc_2$. כלומר, ייתכן כי בסופו של דבר המסלול הכולל שהנהג יבחר לא יהיה אופטימלי (= בעל מרחק נסיעה מינימלי), אך אופן קבלת ההחלטות של הנהג נועד לצמצם את **זמן התכנון** של מסלול הנסיעה.

2. רעיון אלגוריתמי: שיפור אלגוריתמי של חיפוש במרחב חיפוש:

בתרגיל זה ננסה לשפר אלגוריתמֵי חיפוש במרחבי חיפוש שנלמדו בכתה. האופן שבו נעשה זאת יהיה ע"י בניית **מרחב חיפוש אבסטרקטי** מעל מרחב החיפוש הקיים, ובניית אלגוריתם חיפוש שיעשה שימוש במרחב החיפוש שנבנה.

2.1 מרחב חיפוש:

נזכיר כי **מרחב חיפוש** הוא רביעיה: $Space = (S, O, I, G)$, כאשר:

- S היא קבוצת המצבים במרחב.
- O היא קבוצת פעולות/אופרטורים ממצב למצב עוקב, $o_i : S \to S \cup \{\phi\}$ $\quad O = \{o_1, \dots, o_k\}$.
- $I \in S$ הוא המצב ההתחלתי.
- $G \subseteq S$ היא קבוצת המצבים הסופיים.

2.2 מרחב חיפוש אבסטרקטי:

בהינתן מרחב חיפוש $Space = (S, O, I, G)$, נסמן ב-$Ops = O^*$ את קבוצת סדרות האופרטורים (מאורך כלשהו) במרחב החיפוש $Space$. כלומר:

$$Ops = \bigcup_{i=0}^{\infty} O^i$$

כאשר O^i הוא קבוצת כל סדרות האופרטורים מאורך i, כלומר:

$$O^i = \{(o_1^i, o_2^i, \dots, o_i^i) \mid \forall j \in [i]: \ o_j^i \in O\}$$

כמו כן, בהינתן מרחב חיפוש $Space = (S, O, I, G)$, נגדיר עבורו **<u>מרחב חיפוש אבסטרקטי</u>** $Space' = (S', O', I', G')$ באופן הבא:

- $S' \subseteq S$ היא קבוצת המצבים במרחב $Space'$.
- $O' \subseteq Ops$ היא קבוצת פעולות/אופרטורים ממצב למצב עוקב, $o'_i : S' \to S' \cup \{\phi\}$ $O' = \{o'_1, \dots, o'_k\}$ המקיימת: לכל $o' = (o'_1, o'_2, \dots, o'_i) \in O'$ ולכל מצב $s' \in S'$ כך ש-$s' \in Domain(o')$, המסלול המושרה ע"י הפעלת o' על s' מתאר מסלול $s_1 \to s_2 \to \cdots \to s_i$ במרחב החיפוש המקורי כאשר $s_1 = s'$, $s_i \in S'$ ולכל $2 \le j \le i$ מתקיים: $s_j \notin S'$ (כלומר מסלול שאינו עובר דרך מצבים ב-S שיש עבורם מצב מתאים מ-S').
- $I' \in S'$ הוא המצב ההתחלתי.
- $G' \subseteq S'$ היא קבוצת המצבים הסופיים.

כלומר מרחב החיפוש האבסטרקטי $Space'$ מכיל תת-קבוצה של קבוצת המצבים של המרחב המקורי $Space$, והאופרטורים בו מייצגים סדרות של אופרטורים במרחב החיפוש המקורי (במילים אחרות: האופרטורים במרחב החיפוש האבסטרקטי עוברים בין המצבים ב-S' באמצעות הפעלה של סדרות אופרטורים במרחב החיפוש המקורי). שימו לב כי גם אם קיים מסלול (סדרת אופרטורים) המוביל מהמצב $s_1 \in S$ אל המצב $s_2 \in S$ במרחב החיפוש המקורי, אין כל הכרח כי יהיה אופרטור $o' \in O'$ בין המצבים s'_1 ו-s'_2 המתאימים להם במרחב החיפוש האבסטרקטי.

שימו לב כי מרחב החיפוש האבסטרקטי שבנינו אף הוא <u>מרחב מצבים בעצמו</u>, ולכן ניתן, באותו האופן, לבנות מרחב מצבים אבסטרקטי נוסף $Space''$ מעל מרחב המצבים האבסטרקטי $Space'$ שבנינו, ובדרך זו ליצור מרחב רב-שכבתי בו כל שכבה היא מרחב חיפוש אבסטרקטי מעל מרחב החיפוש המיוצג בשכבה שמתחתיה.

2.3 בניית מרחב חיפוש אבסטרקטי:

בהינתן מרחב חיפוש בעל $|S| = N$ מצבים, ובהינתן שני ערכים $k, m \in (0,1]$ (ניתן לחשוב על k ו-m כעל "אחוזים"), נציג תיאור **כללי** לבניית מרחב החיפוש האבסטרקטי:

1. נבחר $kN = N'$ מצבים מרכזיים ב-S להיות המצבים במרחב החיפוש $Space'$ (לפי מדד מרכזיות כלשהו).
2. נחבר כל מצב ב-S' ל-mN' מצבים אחרים כלשהם ב-S', כאשר כזכור, כל קשת (אופרטור) במרחב האבסטרקטי מייצגת סדרת קשתות (אופרטורים) במרחב החיפוש המקורי.

לפי גישה זו, k מייצג את אחוז המצבים מבין כלל המצבים במרחב החיפוש המקורי שהם מצבים מרכזיים, ו-m מייצג את אחוז המצבים מבין מצבי המרחב האבסטרקטי אליהם יחובר כל מצב במרחב האבסטרקטי.

שימו לב כי התיאור הנ"ל הוא כללי וניתן לגזור ממנו דרכים שונות לבניית מרחב החיפוש האבסטרקטי הנבדלות זו מזו:

- באופן בו הן מגדירות את המרכזיות של מצב במרחב חיפוש בשלב (1)

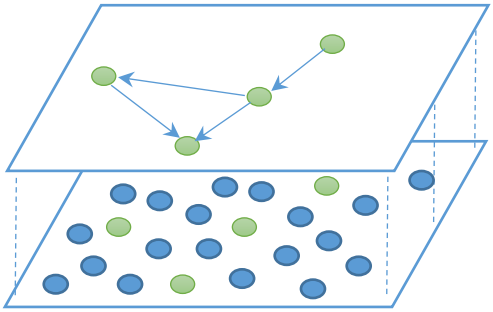
- באופן בו הן בוחרות את mN' המצבים שיחוברו לכל מצב בשלב (2)

- ובאופן בו הן בוחרות את הסדרה של האופרטורים שתקשר בין כל זוג מצבים כנ"ל שנבחרו בשלב (2) (אם יש יותר מסדרה אחת כזו).

כמו כן שימו לב כי קיימות דרכים נוספות לבניית מרחבי חיפוש אבסטרקטי מעל מרחב חיפוש קיים.

2.4 איור להמחשה:

האיור מורכב משתי שכבות. השכבה התחתונה היא מרחב המצבים המקורי $Space$ המכיל מצבים רבים כאשר חלקם מצבים מרכזיים (מסומנים בירוק), והאופרטורים בין המצבים הושמטו לשם בהירות. השכבה העליונה היא מרחב המצבים האבסטרקטי $Space'$ שמצביו הם הצמתים המרכזיים במרחב המצבים המקורי והאופרטורים בו מתארים סדרת אופרטורים במרחב המצבים המקורי.



2.5 מבנה אלגוריתם חיפוש העושה שימוש במרחב חיפוש אבסטרקטי:

לאחר שבנינו את מרחב החיפוש האבסטרקטי, נרצה לבנות אלגוריתם חיפוש **כללי** העושה בו שימוש. אלגוריתם כזה מקבל מצב מוצא A ומצב יעד B, צריך להחזיר מסלול בינהם ופועל באופן הבא:

1. מוצא מצב מרכזי s_1 הקרוב למצב המוצא A.
2. מחפש מסלול מ-A אל s_1 במרחב החיפוש המקורי $Space$.
3. מוצא מצב מרכזי s_2 הקרוב למצב היעד B.
4. מחפש מסלול מ-s_1 אל s_2 במרחב החיפוש האבסטרקטי $Space'$.
5. מחפש מסלול מ-s_2 אל B במרחב החיפוש המקורי $Space$.
6. מחזיר את המסלול מ-A אל B המורכב מתתי-המסלולים שנמצאו בשלבים (2), (4) ו-(5).

שימו לב כי אלגוריתם החיפוש שתואר לעיל הוא **כללי** וניתן לגזור ממנו אלגוריתמי חיפוש שונים הנבדלים זה מזה באופן בו הם בוחרים את המצבים המרכזיים s_1 ו-s_2 בשלבים (1) ו-(3), ובאופן בו הם מחפשים מסלולים בשלבים (2), (4) ו-(5).

כמו כן שימו לב כי כאשר נרצה <u>לממש</u> אלגוריתם חיפוש כנ"ל, נצטרך לבנות את מרחב החיפוש האבסטרקטי **<u>לפני</u>** הרצת אלגוריתם החיפוש עצמו $(pre-proccessing\ time)$, ורק לאחר הבנייה נוכל להריץ את אלגוריתם החיפוש עצמו $(runtime)$. בניית מרחב החיפוש האבסטרקטי מתרחשת פעם <u>יחידה</u> והיא מאפשרת לאחר מכן הרצות רבות של אלגוריתמי חיפוש הנעזרים במרחב זה. דבר זה מאפשר לנו להקדיש זמן חישוב רב לבנייה של מרחב החיפוש האבסטרקטי.

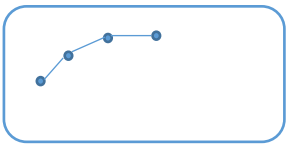
חלק ב'- הגדרת מרחב החיפוש (16 נק')

בחלק הקודם הוצג תיאור כללי של מרחבי חיפוש. כעת נרצה להגדיר מרחב חיפוש עבור התוכנה ***BetterWaze*** אותה תכתבו. בהינתן מפה המתארת רשת כבישים, נגדיר מרחב חיפוש $Roads = (S, O, I, G)$ המתאר בעיית חיפוש ברשת כבישים, כאשר:

- S (קבוצת המצבים במרחב) תוגדר להיות צמתי המפה.

$$S = \{p = (p_x, p_y) \in \mathbb{R}^2 |\ p\ is\ a\ node\ in\ the\ given\ map\}$$

שימו לב שצמתים $(nodes)$ במפה אינם בהכרח $junctions$, ובפרט, תוואי עקום של כביש מתואר ע"י מספר צמתים במפה.



- O (קבוצת האופרטורים ממצב למצב עוקב) תוגדר על בסיס כבישי המפה. הגדרה ישירה של קבוצת האופרטורים במרחב חיפוש זה היא פחות טבעית, ולכן נגדיר את <u>קבוצת המצבים העוקבים</u> של מצב באמצעות פונקציית העוקב $(successor)$:

$$\forall p \in S: \quad Succ(p) = \{p' \in S |\ There\ exists\ a\ road\ from\ p\ to\ p'\}$$

- $I \in S$ (המצב ההתחלתי) ו-$G \subseteq S$ (קבוצת המצבים הסופיים) יוגדרו על סמך בעיית החיפוש הספציפית אותה נרצה לפתור במרחב החיפוש.

ספציפית, מרחב החיפוש (גרף המצבים) יוגדר ע"י קובץ נתונים המייצג את רשת הכבישים של איזור תל-אביב (ברדיוס של מספר קילומטרים מסביב למגדלי עזריאלי). אנו ביצענו הורדה של המפה של איזור תל-אביב מהאתר www.openstreetmap.org והמרה לפורמט שיהיה נוח לעבודה עם $Python$ ושיכלול רק את המידע הרלוונטי מתוך שלל הנתונים המקוריים, זהו הקובץ tlv.csv.

שאלה a - היכרות עם קבצי הקוד והנתונים המסופקים (16 נק'):

1. פִּתחו את הקבצים db/tlv.csv ותארו את המבנה שלהם בִּמְדויק: מה מייצגת כל שורה ומה הפרמטרים בה. לצורך כך, עיינו בקוד של השגרה load_map_from_csv בקובץ הקוד ways/graph.py המסופק.
2. מַמשו את תוכן שגרת העזר map_statistics בקובץ stats.py המחשבת פרמטרים המאפיינים את המפה הנתונה, כאשר $Link\ distance$ מחושב במטרים, וממוצעים צריכים להינתן כטיפוס $float$ (כלומר אין לעגל ממוצע למספר שלם). יש לאפשר להריץ את הקוד עבור סעיף זה דרך שורת הפקודה:

```
$ python stats.py
```

הפלט צריך להיות תוצאת הדפסה פשוטה של המילון אותו מחזירה הפונקציה.

צרפו את התשובות לשני הסעיפים הנ"ל לדו"ח היבש של התרגיל (כלומר על הדו"ח היבש של התרגיל להכיל את התיאור של מבנה הקבצים מסעיף 1 ואת פלט שורת הפקודה מסעיף 2).

חלק ג'- הרצת הניסוי וניתוח תוצאותיו (84 נק')

לאחר שתיארנו בחלק המבוא את המבנה הכללי של אלגוריתם חיפוש העושה שימוש במרחב חיפוש אבסטרקטי, ובחלק ב' את מרחב החיפוש עִמוֹ נעבוד, נציג כאן את תיאור התרגיל עצמו, בו כאמור תממשו את התוכנה $\boldsymbol{BetterWaze}$.

חלק זה (חלק ג') מורכב משישה שלבים. לנוחיותכם, כל שלב מופיע בעמוד חדש.

רשת הכבישים הנתונה מייצגת את מרחב המצבים ("המקורי") עליו נרצה לבצע את החיפוש. נסמן ב-N את מספר הצמתים ($nodes$) בגרף הנתון (הגרף המיוצג ע"י רשת הכבישים הנתונה), וכן נתון $0 < \mathrm{k} \leq 1$ (למשל: $\mathrm{k} = \frac{1}{100}$) שכאמור ניתן לחשוב עליו כעל "אחוזים".

1. מציאת kN צמתים מרכזיים (מרכזים):

על מנת לבנות את מרחב החיפוש האבסטרקטי כפי שתואר בחלק המבוא של התרגיל, עלינו ראשית למצוא kN צמתים מרכזיים בגרף הנתון (אותם צמתים שאת המסלול בינהם נהג המונית מסיפור המוטיבציה משנן מראש). שימו לב כי k הוא פרמטר של האלגוריתם והוא נתון לבחירתנו. בהמשך התרגיל נבחן מהו ערכו האופטימלי עבור בעיה זו.

בחירת kN הצמתים ($nodes$) המרכזיים מתוך N צמתי הגרף הנתון תתבצע באופן הבא:

1. בְּנו מאגר של 500,000 (חמש מאות אלפים) מסלולים ע"י הליכה רנדומית בגרף הנתון עבור $\rho = 0.99$ (הליכה רנדומית בגרף מוגדרת בהמשך). שימו לב כי זמן הריצה קצר משעה (אחרת עליכם לייעל את הקוד שלכם).
2. סִפְרוּ את מספר הפעמים הכולל שצומת בגרף ($node$, לא בהכרח $junction$) מופיע במסלולים שבמאגר שמצאתם (מספר הפעמים שמסלול כלשהו עבר בצומת זה). מספר זה יְיַצג את המרכזיות של צומת. צומת יחשב מרכזי יותר ככל שמסלולים רבים יותר עוברים בו.
3. מִצְאוּ את kN הצמתים ($nodes$) המרכזיים ביותר.

הערות:

(*) שלב זה מתבצע כחלק מבניית מרחב המצבים האבסטרקטי ב-$preProccessing\ time$ ולכן ניתן להקדיש לו זמן חישוב רב.

(*) <u>הליכה רנדומית בגרף</u>- תהליך המתחיל בצומת אקראי כלשהו מהגרף, ומתקדם בין צמתי הגרף. בסיום התהליך מתקבלת קבוצת מסלולים אקראיים בגרף. התהליך פועל באופן הבא:

1. בחרו צומת אקראי (בהתפלגות אחידה) מהגרף. צומת זה יהווה את תחילת המסלול.
2. בהסתברות ρ בחרו להמשיך את יצירת המסלול הנוכחי, ובהסתברות $(1-\rho)$ בחרו לסיימה, כאשר:

א. אם בחרתם להמשיך את יצירת המסלול הנוכחי, הצומת הבא במסלול ייבחר באקראי (בהתפלגות אחידה) מבין צמתי הגרף **השכנים** של הצומת הנוכחי, כלומר הצמתים אליהם ניתן להגיע מהצומת הנוכחי ע"י הפעלה יחידה של אופרטור יחיד.

ב. אם בחרתם לסיים את יצירת המסלול הנוכחי וברצונכם ליצור מסלול נוסף: חזרו לשלב 1.

שאלה b - מציאת מרכזיות הצמתים (8 נק'):

עליכם להגיש קובץ $.csv$. בשם $centrality.csv$ שיכיל את כל הצמתים (ID) ולכל צומת את מספר המעברים בצומת זה (מרכזיות הצומת). על קובץ זה להיות ממוין בסדר <u>יורד</u> של מרכזיות הצמתים, כלומר: השורה ה-i בקובץ תכיל את ה-ID של הצומת ה-i בסדר מרכזיות יורד (נסמנה ID_i) ואת מרכזיותו $(centrality_i)$:

$$line_i: \quad ID_i, centrality_i$$

2. יצירת מרחב חיפוש אבסטרקטי (גרף מרכזים):

כעת, כאשר יש בידנו את קבוצת $kN = N'$ הצמתים המרכזיים, נרצה לבנות מהם מרחב חיפוש אבסטרקטי (גרף מרכזים) באופן שתואר בחלק המבוא של התרגיל.

נסמן ב-$Centers$ את קבוצת N' הצמתים המרכזיים הנ"ל, וכן נתון $m = 0.1$.

בניית הגרף האבסטרקטי תתבצע באופן הבא:

1. לכל צומת $v \in Centers$:

מצאו את mN' המרכזים הקרובים* ביותר ל-v (לא כולל v עצמו) ואת המסלולים האופטימליים אליהם, תוך שימוש ב-$Uniform\ Cost$ (UC) על גבי מרחב החיפוש המקורי.

$(*)$ הכוונה למרכזים שאורך המסלול הקצר ביותר מ-v אליהם הוא הקצר ביותר (ביחס לאורכי המסלולים מצמתים אחרים), ולא בהכרח למרכזים קרובים ביותר על פי מרחק אוירי.

נסמן ב-$Neighbours_v$ את קבוצת המרכזים הקרובים ביותר ל-v (לא כולל v עצמו), וב-PTN_v $(Paths\ To\ Neigbours)$ את קבוצת המסלולים האופטימליים במרחב החיפוש המקורי מ-v אליהם.

2. לכל צומת $v \in Centers$:

א. צרו צומת (מצב) אבסטרקטי v_{abs} במרחב החיפוש האבסטרקטי.

כמו כן נסמן ב-$AbsNeigbours_v$ את קבוצת המצבים **האבסטרקטיים** המתאימים ל-$Neighbours_v$, כלומר: $AbsNeigbours_v = \{v'_{abs} | \ v' \in Neighbours_v\}$

ב. לכל $n_v \in AbsNeighbours_v$:

צרו קשת מכוונת במרחב החיפוש האבסטרקטי מ-v_{abs} אל n_v, ושמרו על גביה את המידע על המסלול הרלוונטי מ-PTN_v.

הערות:

$(*)$ ייתכן כי תמצאו מרכזים "לא חברתיים" עבורם קיימים פחות מ-mN' מסלולים אל מרכזים שכנים. במצב זה דרגת היציאה של מרכז "לא חברתי" v_{abs} כזה במרחב החיפוש האבסטרקטי תהיה קטנה יותר מ-mN'.

$(*)$ שימו לב כי הגרף האבסטרקטי המתקבל מצומצם יותר ממרחב החיפוש המקורי ואף ייתכן כי הוא איננו קשיר.

שאלה c - בניית המרחב האבסטרקטי (8 נק'):

נסמן $K = \{0.0025, 0.005, 0.01, 0.05\}$. עליכם ליצור $dictionary$ של $Python$ הממפה צמתים מרכזיים אל רשימת שכניהם והמסלולים אליהם עבור $k = 0.005$. נסמן ב-$ID(v)$ את ה-ID של מרכז v, וב-$Junction(v)$ את האובייקט מטיפוס $Junction$ שמתאים ל-v. אובייקט זה יכיל את המסלולים (מטיפוס $AbstractLink$) מהצומת v אל הצמתים המרכזיים השכנים אליו והוא מוגדר בקובץ ways/graph.py. אזי המילון יהיה במבנה הבא:

$$dictionary:\ ID(v) \rightarrow Junction(v)$$

שמרו את ה-$dictionary$ באמצעות $pickle$ בקובץ בשם $abstractSpace.pkl$ וצרפו קובץ זה להגשה.

3. בניית DATA SET:

צרו $DataBase$ של 20 זוגות צמתים ($nodes$, לא בהכרח $junctions$) במרחב החיפוש המקורי.

עליכם לוודא כי לכל זוג צמתים (A, B) ב-$DataBase$ הנ"ל:

1. קיים מסלול מ-A אל B במרחב החיפוש המקורי.
2. אורך המסלול הקצר ביותר (במספר הפעלות האופרטורים) בין A ל-B הוא לפחות 200 אופרטורים [כלומר ישנם לפחות 199 צמתי ביניים ($nodes$, לא בהכרח $junctions$) במסלול הקצר ביותר מ-A אל B].

ניתן להבטיח את קיום שתי הדרישות הנ"ל אם בניית זוג במאגר תבוצע ע"י הרצת $Uniform\ Cost$ החל מצומת אקראי במרחב החיפוש המקורי.

על זוגות צמתים אלו תריצו את הניסויים משלב זה ואילך.

שאלה d - בניית $DataSet$ (8 נק'):

עליכם להגיש קובץ $.csv$ בשם $dataSet.csv$ שיכיל את עשרים זוגות הצמתים (ID_1, ID_2) אותם מצאתם.

כלומר קובץ זה יכיל עשרים שורות שונות זו מזו כך שהשורה ה-i במבנה:

$$ID_{i1}, ID_{i2}$$

וַדְאו כי שתי הדרישות שהוגדרו לעיל אכן מתקיימות!

4. בניית אלגוריתמי חיפוש:

כעת, כאשר יש בידנו את מרחב החיפוש האבסטרקטי, נרצה לבנות אלגוריתם חיפוש מסלול העושה בו שימוש ואלגוריתם חיפוש מסלול שלא עושה בו שימוש.

1. ממשו את אלגוריתם החיפוש שלא עושה שימוש במרחב המרכזים האבסטרקטי. אלגוריתם זה, בהינתן שני צמתים (צומת מוצא A וצומת יעד B), נעזר באלגוריתם החיפוש $Uniform\ Cost$ כדי למצוא את המסלול מ-A אל B על גבי מרחב החיפוש המקורי. המימוש צריך להופיע בקובץ main.py תחת הפונקציה "base".
2. ממשו את אלגוריתם החיפוש שעושה שימוש במרחב המרכזים האבסטרקטי (שייקרא $BetterWaze$). אלגוריתם זה, בהינתן שני צמתים (צומת מוצא A וצומת יעד B),
   a. נעזר באלגוריתם החיפוש $Uniform\ Cost$ כדי למצוא צומת מרכזי $Junc_1$ הקרוב ביותר* לצומת המוצא A (מבין הצמתים המרכזיים) ואת המסלול מ-A אל $Junc_1$ במרחב החיפוש המקורי.
   b. מוצא בעזרת חישוב מרחקים אוויריים את הצומת המרכזי $Junc_2$ שצומת היעד B קרוב אליו ביותר (לפי מרחק אוירי), ונעזר באלגוריתם החיפוש $Uniform\ Cost$ למציאת המסלול מ-$Junc_2$ אל B במרחב החיפוש המקורי.
   c. נעזר באלגוריתם החיפוש $Uniform\ Cost$ במרחב החיפוש האבסטרקטי כדי למצוא מסלול מ-$Junc_1$ אל $Junc_2$.
   d. אם שלושת השלבים a, b ו-c הושלמו בהצלחה- מחזיר את המסלול מ-A אל B המורכב מתתי-המסלולים שנמצאו בשלבים a, b ו-c.
   e. אם לא קיים מסלול כנ"ל (כלומר אחד השלבים a, b, c נכשל במציאת תת-מסלול מתאים), מבצע $Uniform$ Cost בין A ל-B לאחר ביצוע השלבים a, b ו-c.

המימוש של האלגוריתם $BetterWaze$ צריך להופיע בקובץ main.py תחת הפונקציה "betterWaze".

הערות:

(*) "צומת קרוב ביותר" הכוונה לצומת שמרחק הנסיעה הקצר ביותר אליו הוא מינימלי (ולא בהכרח קרוב ביותר לפי מרחק אוירי).

(*) שימו לב שעל מנת לקצר את זמן הריצה של שלב $(2.b)$, הצומת הקרוב ביותר הוגדר על סמך מרחק אווירי ולא על סמך מרחק נסיעה במסלול הקצר ביותר.

(*) וַדֵּאו כי ניתן יהיה להריץ את הקוד שלכם באמצעות הפקודות הבאות:

עבור האלגוריתם שלא עושה שימוש במרחב האבסטרקטי:

```
python main.py base <startID> <endID>
```

עבור האלגוריתם שכן עושה שימוש במרחב האבסטרקטי ($BetterWaze$):

```
python main.py bw <startID> <endID> <path to abstractSpace.pkl>
```

כאשר startID וְ-endID מציינים את ה-ID של צמתי ההתחלה והסיום מהם מורצים האלגוריתמים.

שאלה e - דיון על האלגוריתמים (24 נק'):

1. אם היינו רוצים בשלב $(2.b)$ למצוא את הצומת המרכזי $Junc_2$ שצומת היעד B קרוב אליו ביותר (לפי מדד של מרחק נסיעה), היה עלינו להריץ $Uniform\ Cost$ **מ**כל אחד מהמרכזים בגרף האבסטרקטי **אל** היעד B בשלב $(2.b)$ (כלומר לא היינו יכולים להשתמש ב-$Uniform\ Cost$ **מ**היעד B למציאת המרכז הקרוב אליו ביותר). מדוע זהו המצב? איזה תנאי צריך להתקיים על מרחב החיפוש על מנת שנוכל להשתמש ב-$Uniform\ Cost$ **מ**היעד B למציאת המרכז הקרוב אליו ביותר (לפי מדד של מרחק נסיעה)?
2. הציעו שיפור לאלגוריתם עבור המקרים בהם שלב b נכשל במציאת תת-מסלול מתאים. הסבירו מדוע הצעתכם משפרת את האלגוריתם (וביחס לאיזה מדד היא משפרת אותו).
3. הציעו שיפור לאלגוריתם עבור המקרים בהם שלב c נכשל במציאת תת-מסלול מתאים. הסבירו מדוע הצעתכם משפרת את האלגוריתם (וביחס לאיזה מדד היא משפרת אותו).

צרפו את הפתרון לשאלה זו לדו"ח היבש של התרגיל.

5. הרצת הניסוי:

נסמן ב-$DataSet$ את אוסף זוגות הצמתים שנבנו בשלב (3).

כזכור, $K = \{0.0025, 0.005, 0.01, 0.05\}$.

לכל $k \in K$ נסמן ב-$BetterWaze_k$ את אלגוריתם החיפוש $BetterWaze$ העושה שימוש בגרף אבסטרקטי בעל kN מרכזים שנבנה בשלב (2) מעל kN המרכזים שנמצאו בשלב (1).

א. הַריצו את אלגוריתם החיפוש $Uniform$ Cost (שלא עושה שימוש בגרף המרכזים האבסטרקטי) על כל זוג צמתים $(s,t) \in DataSet$. חַשבו את מספר פיתוחי הצמתים ומחיר הפתרון (מרחק נסיעה) עבור כל זוג כנ"ל.

ב. לכל ערך $k \in K$:

a. בְּנו את גרף המרכזים (מרחב החיפוש האבסטרקטי) המתאים $Centers_k$.

b. הריצו את אלגוריתם החיפוש $BetterWaze_k$ על כל זוג צמתים $(s,t) \in DataSet$. חשבו את מספר פיתוחי הצמתים $(*)$ ומחיר הפתרון (מרחק נסיעה) עבור כל זוג כנ"ל. ערכים אלו ישמשו אתכם מאוחר יותר לניתוח תוצאות הניסוי.

$(*)$ אם אין פתרון באמצעות שימוש בגרף האבסטרקטי, מספר פיתוחי הצמתים הוא **סכום** המספרים של פיתוחי הצמתים שבוצעו בנסיון להשתמש בגרף האבסטרקטי ולאחר מכן בביצוע של ה-$Uniform\ Cost$ עצמו.

שאלה f - הרצת הניסוי (8 נק'):

עליכם להגיש קובץ $.csv$. בשם $experiment.csv$ שיכיל את הנתונים שהתקבלו בשלבים א' וב' של הרצת הניסוי. מבנה הקובץ יכיל שורה לכל דוגמא (s,t) מה-$DataSet$ שבניתם בשלב (3), וכל שורה תכיל את צומת המוצא (s) וצומת היעד (t), ולכל אלגוריתם את מספר פיתוחי הצמתים ומחיר הפתרון שהתקבלו עבור הדוגמא (s,t). בהינתן אלגוריתם Alg נסמן ב-$N(Alg_{s,t})$ את מספר פיתוחי הצמתים שנדרשו לאלגוריתם Alg למציאת מסלול מ-s אל t, וב-$cost(Alg_{s,t})$ את מחיר הפתרון (מחיר המסלול) שהוחזר ע"י האלגוריתם Alg עבור מציאת מסלול מ-s אל t. לכן מבנה שורה בקובץ ה-csv. הנ"ל הוא:

$$s, t, N(UC_{s,t}), cost(UC_{s,t}), N\left(BetterWaze_{k1_{s,t}}\right), cost\left(BetterWaze_{k1_{s,t}}\right), N\left(BetterWaze_{k2_{s,t}}\right),$$
$$cost\left(BetterWaze_{k2_{s,t}}\right), \dots, N\left(BetterWaze_{k4_{s,t}}\right), cost\left(BetterWaze_{k4_{s,t}}\right)$$

כלומר כל שורה מורכבת משתי עמודות צמתים ושתי עמודות לכל אלגוריתם ← סה"כ 12 עמודות.

6. ניתוח תוצאות הניסוי:

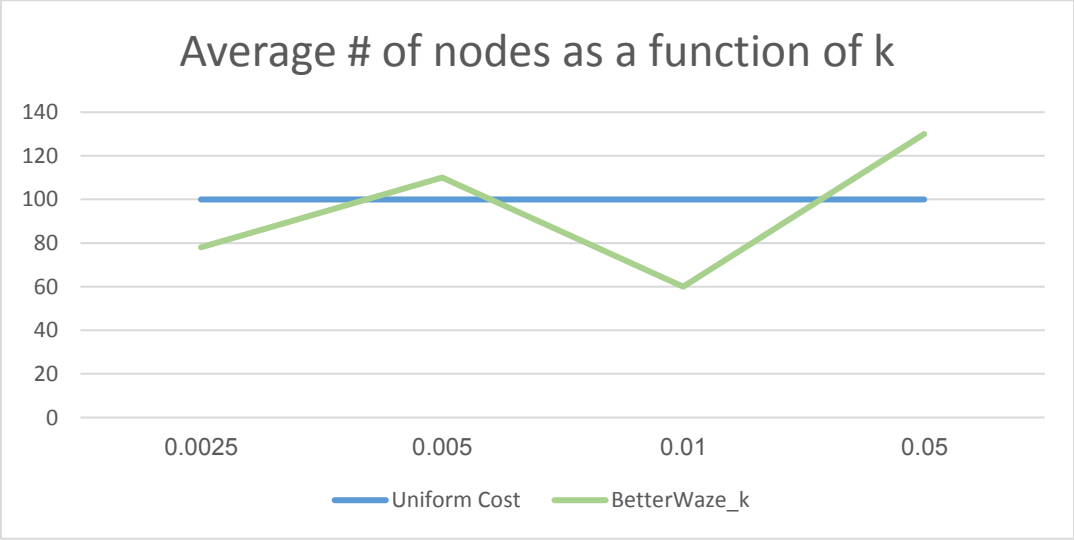
נזכיר כי אלגוריתם החיפוש $BetterWaze_k$ נועד לשפר את **יעילות האלגוריתם** על חשבון פגיעה אפשרית ב**איכות הפתרון** המוחזר על ידו. אנו נרצה לבחון את יחסי הגומלין בין שני המדדים האלו כתלות בערכו של הפרמטר k:

1. **יעילות האלגוריתם-** נמדדת על-ידי מספר הצמתים שפותחו באלגוריתם החיפוש כתלות ב-k. מתוך הנחה כי באלגוריתם העושה שימוש בגרף המרכזים האבסטרקטי נעשים פיתוחים מועטים יותר של צמתים (בזמן חיפוש המסלול. אנחנו מתעלמים מהזמן שנדרש ל-$PreProcessing$).
2. **איכות הפתרון-** נמדדת על-ידי יחס המחירים $\left(\frac{cost(betterWaze)}{cost(UC)}\right)$ כתלות ב-k. מתוך הנחה כי האלגוריתם העושה שימוש בגרף המרכזים האבסטרקטי מחזיר מסלולים בעלי מחיר גבוה יותר מהמסלולים המוחזרים ע"י $Uniform\ Cost$ ($Uniform\ Cost$ הוא אלגוריתם קביל ולכן מובטח כי הוא מחזיר מסלולים אופטימליים).

שאלה g - ניתוח תוצאות הניסוי (28 נק'):

ראשית נבחן את **יעילות האלגוריתמים**. לשם כך:

א. הציגו גרף המתאר את **ממוצע מספר הצמתים שפותחו** (על פני כל הבעיות ב-$DataSet$ שבניתם) בשני האלגוריתמים כתלות ב-k. שימו לב כי ביצועיו של אלגוריתם ה-$Uniform\ Cost$ **אינם תלויים** בערכו של k ולכן הוא יתואר בגרף באמצעות קו אופקי, בעוד שביצועיו של $BetterWaze_k$ עשויים להשתנות כתלות בערכו של k. לנוחיותכם מצורף גרף לדוגמא ($*$).



ב. עבור גרף זה צרפו גם את **טבלת הנתונים** שיצרו אותו, למשל $(*)$:

$BetterWaze\_k$	$Uniform\ Cost$	k
130	100	0.0025
60		0.005
110		0.01
78		0.05

עבור גרף זה עליכם לענות **בִּמְפורט** על הסעיפים הבאים:

ג. מהו הערך של הפרמטר k עבורו אלגוריתם ה-$BetterWaze_k$ המתקבל הוא היעיל ביותר?

ד. הַסבירו את מגמות השיפור והדעיכה ביעילות של $BetterWaze_k$ כתלות בערכו של הפרמטר k. במקרה ולא ניתן לחזות במגמה ברורה- ספקו הסבר אפשרי מדוע זהו המצב.

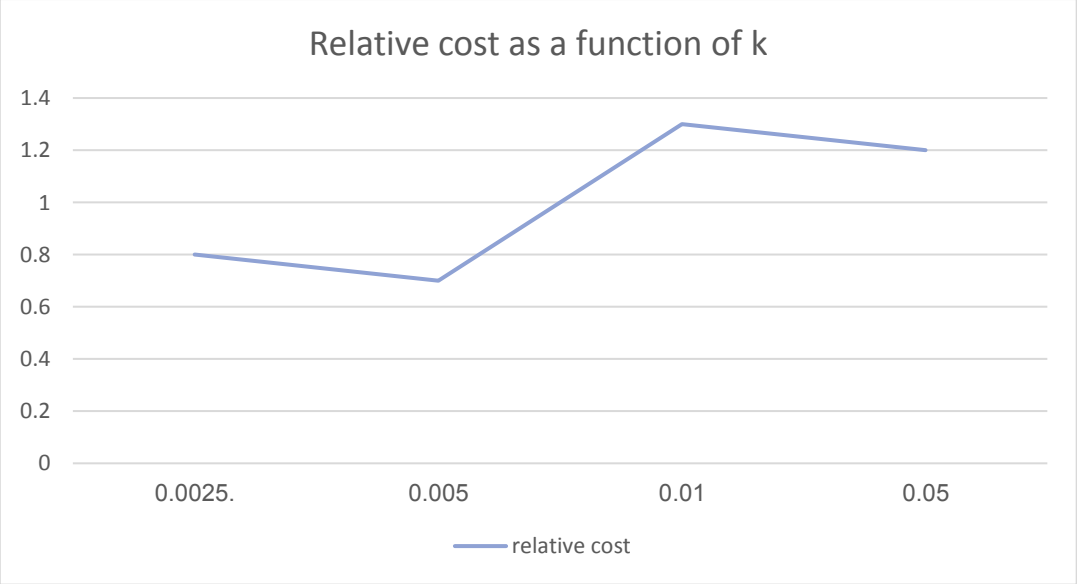
כעת נבחן את **איכות הפתרון של האלגוריתמים**. לשם כך:

א. הציגו גרף המתאר את **<u>ממוצע יחס המחירים</u>** של הפתרונות שמחזירים שני האלגוריתמים $\left[Avg\left(\frac{cost(betterWaze_k)}{cost(UC)}\right)\right]$ כתלות ב-k. שימו לב כי אנחנו מחשבים את **<u>הממוצע הגיאומטרי</u>** של יחס המחירים (לנוחיותכם, הנוסחא שלו מצורפת בהמשך). שימו לב כי הגרף מציג את הממוצע של <u>יחסי המחירים</u>, כלומר לכל k עליכם **ראשית** לחשב את <u>יחס המחירים</u> עבור הדוגמא ה-i ב-$DataSet$. אם נסמן דוגמא זו ב-$ex_i = (s_i, t_i)$, אזי היחס המתאים לה עבור k ספציפי יסומן ב- $relCost_{k,i} = \frac{cost(betterWaze_k(ex_i))}{cost(UC(ex_i))}$. כמו כן את גודל ה-$DataSet$ נסמן ב-$D = size(DataSet)$. רק לאחר שיש בידיכם את יחסי המחירים עליכם לחשב את **<u>הממוצע הגיאומטרי</u>** של יחס המחירים עבור k, כלומר לחשב את:

$$Avg_k = \sqrt[D]{\prod_{i=1}^{D} relCost_{k,i}}$$

הנקודות בגרף שיתקבל תהיינה מהצורה: (k, Avg_k).

לנוחיותכם מצורף גרף לדוגמא $(*)$.



ב. עבור גרף זה צרפו גם את **טבלת הנתונים** שיצרו אותו, למשל $(*)$:

Relative cost	***k***
1.2	0.0025
1.3	0.005
0.7	0.01
0.8	0.05

עבור גרף זה עליכם לענות **בִּמְפורט** על הסעיפים הבאים:

ג. מהו הערך של הפרמטר k עבורו היחס המתקבל הוא אופטימלי?

ד. הַסבירו את מגמות השיפור והדעיכה באיכות הפתרון המוחזר על-ידי $BetterWaze_k$ כתלות בערכו של הפרמטר k. במקרה ולא ניתן לחזות במגמה ברורה- ספקו הסבר אפשרי מדוע זהו המצב.

כלומר סך הכל, בשאלה זו (שאלה g) עליכם להציג שני גרפים ושתי טבלאות, ולענות על ארבעת הסעיפים המצורפים אליהם.

$(*)$ שימו לב כי הגרפים והטבלאות שהצגנו כאן נוצרו על סמך ערכים שרירותיים שקבענו לצורך **המחשה בלבד**. לא מובטח שהתוצאות שתקבלו בהרצת הניסויים שלכם תהיינה דומות לערכים המוצגים בגרף ובטבלה שלעיל.

חלק ד'- שאלת בונוס (עד 15 נק')

הציעו דרך טובה יותר לבחור את kN הצמתים המרכזיים שירכיבו את מרחב המצבים האבסטרקטי על מנת לקבל **אלגוריתם חיפוש מוצלח יותר**. הסבירו מדוע שיטה זו טובה יותר מהשיטה שהתבקשתם לבצע בתרגיל, והראו את ביצועיה על גבי הגרפים והטבלאות משאלה g.

בונוס יינתן עבור תשובות מנומקות ויצירתיות במיוחד.

הוראות ההגשה בעמוד הבא!

הוראות הגשה

- הגשת התרגיל תתבצע **אלקטרונית בלבד**.
- עליכם להגיש קובץ ארכיון יחיד בשם: AI1_<id1>_<id2>.zip (ללא הסוגריים המשולשים). קובץ זה יכיל:
  - קובץ בשם readme.txt בפורמט הבא:

```
name1 id1 email1
name2 id2 email2
```

  - קובץ בשם AI_HW1.PDF המכיל את דו"ח הניסויים שערכתם, תשובות לחלק היבש והערות לקוד שהגשתם (כולל תפקיד כל קובץ הנמצא בתיקייה שהגשתם).
  - קובץ בשם requirements.txt שיכיל כל חבילה חיצונית שאינה מותקנת כחלק מ-Anaconda Python. על אחריותכם לוודא כי ניתן להתקין כל חבילה כנ"ל באמצעות הרצת השורה: pip install -r requirements.txt
  - את הקובץ main.py שפורסם כחלק מקבצי הקוד של התרגיל לאחר שמימשתם את הפונקציות הרלוונטיות.
  - כל **קוד עזר** שכתבתם/השתמשתם בו לשם הרצת הניסויים או יצירת הגרפים.
- אין להעתיק את הקבצים המסופקים לכם אל תוך תיקיית ההגשה. הניחו כי קבצים אלו יהיו זמינים בעת בדיקת התרגיל.
- שימו לב שכל הפנייה למיקום קובץ/תיקייה כלשהם בקוד תהיה רלטיבית ($relative\ path$) ולא אבסולוטית, כך שהקוד יעבוד כפי שהוא על כל מחשב בכל מיקום שנבחר לתיקיית הפרוייקט. הקפידו לבדוק זאת לפני ההגשה!
- "המצאת" נתונים לצורך בניית הגרפים **אסורה** ותוביל לדיון בבית הדין המשמעתי של הטכניון.
- אתם רשאים לעשות שימוש בכל קוד שתמצאו ברשת, אך כל קוד חיצוני **מחייב הצהרה מפורשת** על המקור שלו בקובץ AI_HW1.PDF. אי-קיום דרישה זו מהווה עבירה משמעתית!
- הקפידו על קוד **<u>ברור, קריא ומתועד!</u>** עליכם לתעד כל חלק שאינו טריוויאלי בקוד שלכם. בפרט, אם התשמשתם בקוד שנמצא ברשת וביצעתם בו שינויים, עליכם לתעד זאת.

בהצלחה!!