# תרגיל בית 1: אוטובוס הקסמים

**מטרות התרגיל**

- נתמודד עם בעיות פרקטיות ותיאורטיות של חיפוש במרחבי מצבים גדולים.
- נתרגל את הנלמד בהרצאות ובתירגולים.
- נתנסה בתכנות ב-python לפיתרון בעיות פרקטיות.

**הנחיות כלליות**

- **תאריך הגשה:** יום ב', 04.12.2017, בשעה 23:55.
- 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 .
- בקשות דחיה **מוצדקות** יש לשלוח לאמיר דנ"כ 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:
  - **הדו"ח הסופי.** מעבר לתשובות הנכונות, אתם נבחנים גם על הצגת הנתונים והתוצאות בצורה קריאה ומסודרת. בפרט, יש לכתוב כותרות מתאימות לגרפים ולצירים בגרפים (כולל יחידות מידה היכן שצריך).
  - **הקוד המוגש.** יש להקפיד על הגשת קוד מסודרת בהתאם להנחיות. יש לכתוב הערות במקומות חשובים בקוד כדי שיהיה קריא וקל לבדיקה.
- גרסאות python איתן אתם נדרשים לעבוד הן 3.5 או 3.6. גם קבצי המקור שקיבלתם מתאימים לגירסאות אלה.
- אלא אם נכתב אחרת, אין לשנות את פונקציות מוכנות שקיבלתם, או את החתימה של פונקציות שהתבקשתם לממש. אם יש בעיה נקודתית, ניתן לשלוח מייל לתיבה הקורסית.
- לצורך ההרצות תצטרכו להתקין חבילות של python כמו numpy, scipy, matplotlib וכו'.
- בתרגילי הבית בקורס הרצת הניסויים עשוייה לקחת זמן רב, ולכן מומלץ מאוד להימנע מדחיית העבודה על התרגיל לרגע האחרון. לא תינתנה דחיות על רקע זה.

הבהרות ועדכונים שנוספים אחרי הפרסום הראשוני יסומנו בצהוב.

# פרק ראשון – אוטובוס הקסמים (90 נק')

## חלק א' – מבוא והנחיות

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 ולהריץ את תוצאותיהם. אתם נדרשים לבצע ניתוח מעמיק ומפורט של התוצאות, כפי שיוסבר בהמשך.

### מוטיבציה

אוטובוס הקסמים הוא מיזם לתחבורה ציבורית שיתופית במסגרתו מסתובב אוטובוס קסום ברחבי המדינה, אוסף אנשים לפי הזמנות הנתונות מראש ומוריד אותם במחוז חפצם. כמעט כמו מונית ספיישל, אך עם אופי יעיל וחסכוני המאפשר לכל הלקוחות לחלוק את הנסיעה – למשל, ניתן לאסוף מתל אביב לקוחה שרוצה להגיע לחיפה, לנסוע לחדרה ולאסוף שני לקוחות שרוצים להגיע לדלית אל-כרמל, להוריד אותם בדלית אל-כרמל ומשם לנסוע לחיפה ולהוריד את הלקוחה הראשונה.

מה קסום באוטובוס אתם שואלים? מספר המקומות על האוטובוס אינו מוגבל!

המטרה שלנו היא לכתוב תוכנית שתקבל הזמנות מלקוחות ותתכנן תוכנית נסיעה שתיקח את כל הלקוחות מנקודת האיסוף לנקודת העצירה שלהם, תוך הבאת המרחק הכולל של הנסיעה למינימום שנצליח.

אז... לחגור חגורות!

### פורמאליזם

נתונה רשת כבישים בצורת גרף $G_M = (V, E)$ שבו כל צומת (vertex) מייצג צומת דרכים (junction), והקשתות (edges) מייצגות דרך המקשרת בין צמתי דרכים (links).

כמו כן נתונה נקודה מוצא על הגרף $v_0 \in V$, וכן נתונות $k \in \mathbb{N}$ הזמנות: $Ord = \{(s_1, t_1), \dots, (s_k, t_k)\}$, כאשר הזמנה $i$ היא זוג סדור של מקור ויעד $s_i, t_i \in V$.

לצורך פשטות, ניתן להניח שאין הזמנות שמתחילות או נגמרות בנקודת המוצא $v_0$.

אנו רוצים להחזיר מסלול נסיעה חוקי $P$ הממלא את כל דרישות הלקוחות:

1. $P$ מתחיל בנקודת המוצא $v_0$.
2. $P$ עובר בכל המקורות ובכל היעדים ($s_i, t_i$ לכל $i$).
3. לכל הזמנה $i$, $P$ עובר במקור $s_i$ לפחות פעם אחת לפני הפעם האחרונה בה הוא עובר ביעד $t_i$.

את ביצועי התוכנית נמדוד לפי מרחק הנסיעה הכולל, כפי שיפורט בהמשך.

### הבנת הבעיה

אנו מחפשים למעשה פרמוטציה מהצורה $v_0, s_i, \dots, s_j, \dots, t_j, \dots, t_i, \dots$, שמקיימת את הדרישות הכתובות לעיל. בהינתן פרמוטציה שכזו, נרצה למלא בין כל שתי נקודות עוקבות בו את המסלול הקל ביותר ביניהן, אותו ניתן למצוא בעזרת אלגוריתם A*, שכפי שראינו בהרצאות, שהוא אלגוריתם לחיפוש מיודע.

ניתן להראות שישנן $\frac{(2k)!}{2^k}$ פרמוטציות כאלה, לכן כבר עבור כמות הזמנות לא גדולה במיוחד, לא נוכל לפתור את הבעיה בעזרת חיפוש brute force.

*הערה: הבעיה בה אנו עוסקים מזכירה את בעיית הסוכן הנוסע האסימטרית (שהיא בעיה קשה חישובית ממחלקת NP-hard) ואף מכלילה אותה. נזכיר כי בבעיית הסוכן הנוסע האסימטרית רוצים למצוא בגרף מכוון נתון מהלך המילטוני במשקל מינימלי.*

במהלך המטלה ניגש לבעיה מזוויות שונות וננסה למצוא פתרונות טובים בעזרת כלים שלמדנו בקורס.

### תרגיל

1. הזינו טבלה של מספר הפרמוטציות האפשריות עבור ערכי $k$ (מספר הזמנות) מ-1 עד 10. היעזרו בנוסחה דלעיל.

## חלק ב' – הגדרת מרחב החיפוש במפה

בהינתן רשת הכבישים, נקודת מקור $u \in V$ ונקודת יעד $v \in V$, נגדיר מרחב חיפוש עבור מציאת מסלול ביניהן:

$$Map = (S, O, I, G)$$

מרחב החיפוש שיווצר יהיה קצת טריוויאלי וכמעט זהה לגרף של רשת הכבישים, אך השמירה על כלליות תאפשר לנו לממש את האלגוריתמים באופן כללי יותר.

- **קבוצת המצבים:**
נרצה לייצג מצב כך שיחזיק את כל המידע שנחוץ לנו על נקודת זמן בפיתרון, ושניתן להמשיך ממנה הלאה (אך את הפיתרון נמצא למשל על ידי מעבר על ההורים בעץ החיפוש). במקרה המדובר מספיק לשמור את הצומת בגרף בו אנו נמצאים.
$$S = \{(v) | v \in V\}$$
- **קבוצת האופרטורים:**
ניתן לעבור ממצב אחד למשנהו בתנאי שיש כביש מהצומת המייצג את המצב הראשון לצומת המייצג את המצב השני.
~~כמו כן נדרש שלפחות המצב של הזמנה אחת משתנה בעקבות הפעלת האופרטור.~~ שורה נמחקה.
$$O = \{(S_1, S_2) | (S_1.v, S_2.v) \in E\}$$
  - **עלות אופרטור:**
במטלה נגדיר המחיר של מעבר מצומת דרכים אחד לצומת דרכים אחר ע"י האורך של הכביש ביניהם.
$$cost\big((S_1, S_2)\big) = length\big((S_1.v, S_2.v)\big)$$
- **המצב ההתחלתי:**
$$I = (u)$$

## חלק ג' – הגדרת מרחב החיפוש של מסלולי נסיעת האוטובוס (15 נקודות)

בהינתן רשת הכבישים, נקודת המוצא ורשימת ההזמנות, נגדיר מרחב חיפוש עבור אוטובוס הקסמים:

$$Bus = (S, O, I, G)$$

- **קבוצת המצבים:**

נרצה לייצג מצב כך שיחזיק את כל המידע שנחוץ לנו על נקודת זמן בפיתרון, ושניתן להמשיך ממנה הלאה (אך את הפיתרון נמצא למשל על ידי מעבר על ההורים בעץ החיפוש).

$$S = \left\{ \left( \underbrace{v}_{\text{המיקום הנוכחי}}, \underbrace{W}_{\text{הזמנות ממתינות}}, \underbrace{B}_{\text{הזמנות באוטובוס}}, \underbrace{F}_{\text{הזמנות גמורות}} \right) \middle| \begin{array}{c} v \in V \\ W, B, F \subseteq [k] \\ W \cap B \cap F = \emptyset \\ W \cup B \cup F = [k] \end{array} \right\}$$

הערה: מספיק לשמור במצב רק שתיים משלוש הרשימות, ואת השלישית ניתן להסיק משתי הרשימות השמורות ומרשימת ההזמנות ההתחלתית $Ord$. לצורך נוחות הדיון נשמור את שלוש הרשימות בכל מצב.

- **קבוצת האופרטורים:**

ניתן לעבור ממצב אחד למשנהו בתנאי שכל מי שהמתין לאיסוף ונמצא ב-$v_2$ עולה על האוטובוס – משמע יוצא מהרשימה $W_1$ ונכנס לרשימה $B_2$, וכמו כן מי שהיה על האוטובוס והיעד שלו היה $v_2$ יורד מהאוטובוס ונכנס לרשימה $F_2$.

כמו כן נדרוש שלפחות המצב של הזמנה אחת משתנה בעקבות הפעלת האופרטור.

$$O = \left\{ \left( (v_1, W_1, B_1, F_1), (v_2, W_2, B_2, F_2) \right) \middle| \begin{array}{c} W_1 \setminus W_2 = B_2 \setminus B_1 = \{i | i \in W_1 \wedge s_i = v_2\} \\ F_2 \setminus F_1 = \{i | i \in B_1 \wedge t_i = v_2\} \\ W_1 \neq W_2 \ \vee \ B_1 \neq B_2 \end{array} \right\}$$

  - **עלות אופרטור:**

המחיר של מעבר בין מצב אחד למשנהו הוא משקל המסלול הקל ביותר בין $v_1$ ל-$v_2$, שכפי שכתבנו ניתן לחישוב ע"י $A^*$ (למשל).

$$cost\left( \left( (v_1, W_1, B_1, F_1), (v_2, W_2, B_2, F_2) \right) \right) = d_{A^*}(v_1, v_2)$$

- **המצב ההתחלתי:**

$$I = (v_0, Ord, \emptyset, \emptyset)$$

### תרגילים

2. מהם ערכי הקיצון (המקסימלי והמינימלי) האפשריים של מקדם הסיעוף במרחב החיפוש? נמקו בקצרה.
3. האם ייתכנו מעגלים במרחב המצבים שלנו? אם כן תנו דוגמה למעגל כזה, אחרת נמקו.
4. הגדירו פורמאלית ומילולית את $G$ – קבוצת מצבי המטרה.
5. בהנחה שאין שתי הזמנות שמתחילות או נגמרות במיקומים זהים, כמה מצבי מטרה יש בגרף החיפוש? יש לכתוב ביטוי מדוייק כפונקציה של הקלט ולנמק.

**שימו לב:** ההנחה תקפה רק לסעיפים 5 ו-8 (ב-datasets שתקבלו יש חפיפה). הכוונה בהנחה היא שלכל שתי הזמנות $(s_i, t_i), (s_j, t_j)$ מתקיים ש-$s_i, t_i, s_j, t_j$ שונים <u>כולם</u> זה מזה.

6. האם ייתכנו בורות ישיגים מהמצב ההתחלתי שאינם מצבי מטרה במרחב המצבים?
7. הגדירו פורמלית את פונקציית העוקב $Succ: S \to 2^S$ (ללא שימוש בקבוצה $O$).

שימו לב, אנו מצפים לביטוי מהצורה:

$$Succ\left( (v_1, W_1, B_1, F_1) \right) = \left\{ (v_2, W_2, B_2, F_2) \middle| \begin{array}{c} W_2 = ?, B_2 = ?, F_2 = ? \\ v_2 \in ? \end{array} \right\}$$

8. בהנחה שאין שתי הזמנות שמתחילות או נגמרות במיקומים זהים, מהם ערכי הקיצון (המקסימלי והמינימלי) של העומק של מצב מטרה כלשהו במרחב החיפוש? נמקו.

## חלק ד' – מתחילים לתכנת (10 נק')

הורידו את *AI1.zip* מהאתר וטענו את התיקייה שבתוכו לסביבת העבודה המועדפת עליכם.

### מבנה מפת הדרכים

את מפת הדרכים טוענים בעזרת קריאה לפונקציה *ways.load_map_from_csv* עם פרמטר שמציין את הנתיב של קובץ הדאטה. חלק מהסקריפטים שקיבלתם כבר מכוונים לטעון את הקובץ לפי נתיב יחסי, ואמורים להמשיך ולעבוד גם בלי פעולה נוספת מצידכם.

פונקציית הטעינה מחזירה אובייקט מסוג *Roads*, שמכיל מתודת *junctions* שמחזירה את כל הצמתים שעל המפה, ולכל צומת יש רשימת *links* המכילה את כל הקשתות לשכניו, בצורת אובייקטים עם מאפיינים source ו-target.

בנוסף יש לכל צומת tuple בשם *coordinates* עם קווי הרוחב והאורך שלו בצורת *(lat,lon)*.

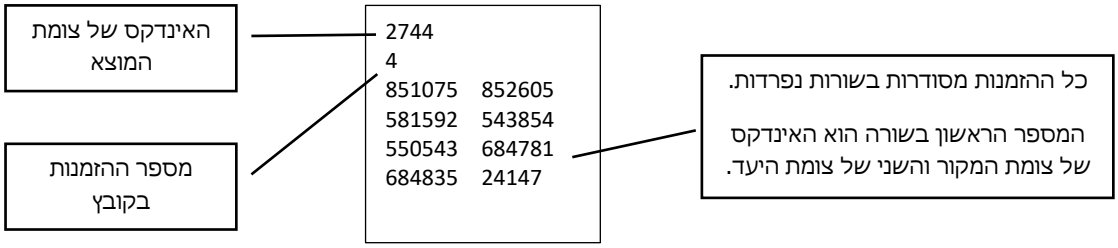
כמו כן ניתן להתייחס לאובייקט Roads בתור dictionary של צמתים.

אינכם אמורים לשנות את הקוד של המפה, אלא רק להשתמש בו.

### מבנה קבצי ההזמנות

את הבעיה נייצג במחלקה הנקראת Problem.

כמה קבצי קלט לדוגמה נתונים בתוך התיקייה *db* עם הסיומת *.in*. מבנה כל קובץ כזה הוא למשל:




הקוד שקורא את קבצי הקלט כבר מוכן ואין צורך לממש קריאות נוספות מקבצים.

### תרגילים

9. על מנת להתחיל להכיר את הקוד המסופק לכם, קראו והבינו את הסקריפט לדוגמה שנמצא ב-*scriptsAndExperiments/initial.py*.
הריצו את הקוד. זאת גם הזדמנות טובה לוודא שחבילות numpy, matplotlib מותקנות אצלכם כראוי.
שימו לב, בקובץ יש לכם שתי משימות (המסומנות ע"י הערות **TODO** כמו בעוד מקומות רבים לאורך המטלה).
   a. תקנו את הדפסת פרטי ההזמנה ואת חישוב החסם התחתון.
   מה הפלט המעודכן של שתי ההודעות?
   b. תקנו את הדפסת אורך המסלול לדוגמה (בקילומטרים). תוכלו לחשב אותו ע"י מתודה מוכנה של מחלקת *Path*. מה אורכו?
   c. תארו בקצרה מה פעולת הקוד והעתיקו את הפלט המתוקן לתשובתכם.
10. המחלקה *BusProblem* מתארת את מרחב המצבים של בעיית האוטובוס. המחלקה *BusState* מתארת מצב יחיד. קראו והבינו את פעולות המחלקות ואת המבנה ההיררכי שלהן (ממי הן יורשות, ואילו מחלקות נוספות יורשות ממחלקות הבסיס שלהן).
**השלימו את המתודה *BusProblem._getNewStateAtLoc***, לפי הפעולה המבוקשת של פונקציית העוקב שכתבתם בחלק הקודם.
כמו כן **השלימו את המתודה *BusState.isGoal*** שבודקת האם מצב נתון הוא מצב מטרה, לפי ההגדרה שכתבתם בחלק הקודם.
אלה שתי מתודות מאוד חשובות לתוכנית, לכן כתבנו עבורכם סקריפט שיעזור לוודא שהן עובדות כשורה (אך זו אינה בדיקה מקיפה):
**הריצו** את הקובץ *scriptsAndExperiments/problemVerify.py* וודאו שאתם מקבלים 3 מצבים עוקבים למצב הנתון בקובץ וששלושתם אינם מצבים סופיים.
**העתיקו לדו"ח** את המידע שנפלט עבור המצב העוקב שבו Junction idx: #851288.

## חלק ה' – אלגוריתם A* (10 נק')

עתה נתחיל במימוש A*.

שימו לב שסיפקנו לכם stub (או שלד) של חלקים שונים בפעולתו. נשתדל במהלך המימוש לשמור עליו כללי ככל שניתן (ע"י שימוש במחלקות Problem האבסטרקטיות למשל), כדי שפעולת האלגוריתם תתאים גם למטרות אליהן נגיע בהמשך המטלה.

### תרגילים

11. השלימו את החלק החסר במתודה *costs.L2DistanceCost.compute* כך שתחשב את המרחק האווירי בין שני מצבים.

**שימו לב, לא ניתן לחשב את המרחק האווירי ישירות על ידי קווי רוחב ואורך**, אלא יש להשתמש בפונקציה מתאימה הנמצאת בקובץ *ways.tools*.

באופן דומה השלימו את החלק החסר במתודה *heuristics.L2DistanceHeuristic.estimate* כך שתחשב את המרחק האווירי בין מצב במרחב למצב המטרה במרחב (יש property כזה ל-*MapProblem*).

**השלימו את הקוד במחלקת *AStar*** (חפשו אחר ה-TODO). תוכלו להיעזר במצגות לשם כך.

שימו לב, הפונקציה *run* חייבת להחזיר את התוצאות הבאות בתוך tuple לפי הסדר:

$$\left(\underbrace{developed}_{\substack{\text{כמה מצבים}\\ \text{פותחו}\\ (\text{קריאה ל-}succ)}},\ \underbrace{h(I)}_{\substack{\text{הערך היוריסטי}\\ \text{של המצב}\\ \text{ההתחלתי}}},\ \underbrace{g(goal)}_{\substack{\text{המחיר איתו}\\ \text{הגענו}\\ \text{למצב המטרה}}},\ \underbrace{path}_{\substack{\text{סדרת המצבים}\\ \text{מהמצב ההתחלתי}\\ \text{למצב המטרה}}}\right)$$

**הבהרה:** *path* צריך להיות רשימת מצבים (שיורשים ממחלקת *state*), ולא רשימה של אינדקסים.

**שימו לב:** בתוך המתודה *AStar.run(...)* שאתם מתבקשים להשלים יש כבר שלד של מספר פעולות. כמו למשל יצירה של set של צמתים סגורים ו-dictionaries שימושיים:

- *parents* מ-*state* ל-*state* הקדמון שלו (בחיפוש, עשוי להתעדכן במהלכו).
- *g_score* מ-state לערך g שלו.
- *open_set* מ-state פתוח לערך f שלו.

עם זאת, זו רק המלצה/הצעה למימוש. אתם לא חייבים להשתמש בדיוק במבני הנתונים האלה. תוכן המתודה נתון לשיקולכם ולמימושכם, **אך אין לשנות את החתימות של המחלקה והמתודה**, וכן המתודה צריכה להשתמש ב-caching (ולשמור אליו כשהתוצאה מוכנה) והיא צריכה להחזיר בדיוק את מה שכתוב לעיל.

---

הריצו את *scriptsAndExperiments/astarVerify.py*. לאחר טעינת הגרף המקורי, כל לחיצה אמורה להציג את המסלול האופטימלי בין נקודת המקור והיעד של ההזמנה הנוכחית. היאזרו בסבלנות, ייתכן שלאחר כל לחיצה יידרש זמן חישוב לפני שהמסלול יופיע על המסך.

**צרפו לדו"ח** תמונה של התרשים הסופי שנוצר ומציג את כל המסלולים על המפה, וכן כתבו את סכום המסלולים (כתוב בסוף הסקריפט).

12. הסתכלו על הקוד ובחרו את הטענה המתאימה (נמקו בקצרה):

באופן כללי (לאו דווקא על ה-dataset הנתון) סכום המסלולים המופיע בסוף הסקריפט הוא...

a. חסם עליון למחיר פיתרון אופטימלי לבעיית האוטובוס על אותן ההזמנות.
b. חסם תחתון למחיר פיתרון אופטימלי לבעיית האוטובוס על אותן ההזמנות.
c. לא ניתן לקבוע באופן כללי.

### Caching

על מנת לחסוך חישובים חוזרים של אותם המסלולים, במיוחד בחלק הסטוכאסטי של המטלה בו נבצע הרצות רבות בזו אחר זו, נרצה להוסיף מנגנון מטמון למחלקה שלנו.

המתודות *_getFromCache, _storeInCache, _shouldCache* כבר מוכנות עבורכם במחלקת *AStar*.

כל שעליכם לעשות הוא להוסיף קריאה למתודת השמירה *_storeInCache* עם התוצאה במקום המתאים בקוד.

**שימו לב:** על מנת להפעיל את מנגנון המטמון יש ליצור אובייקט *AStar* עם ארגומנט *shouldCache=True*.

## חלק ו' – אלגוריתם חיפוש חמדני-דטרמיניסטי (10 נק')

נתחיל את ההתמודדות עם הבעיה בעזרת אלגוריתם חמדני פשוט ואינטואיטיבי.

נציג את הפסאודו-קוד של האלגוריתם שמוצא את סדר האיסוף:

```
curr ← initialState
pickingOrder ← [curr]
while not reach goal:
        nextState = argmin_{s∈succ(curr)} score(curr, s)
        add nextState to pickingOrder
        curr ←nextState
return pickingOrder
```

(האלגוריתם מחזיר פרמוטציה של המקורות והיעדים בהזמנות, ולא מסלול במפה)

פונקציית ה-$score$ יכולה להחזיר מדדים שונים. על מנת להחיש את החיפוש, **אנו נשתמש במרחק אווירי** (שהוא קל לחישוב), על אף שפונקציית המטרה שאנו מנסים להביא למינימום היא סכום האורכים האמיתיים של הכבישים. שימו לב, המרחק האווירי בין שני מצבים הוא פשוט המרחק האווירי בין המיקומים בהם הם נמצאים על מפת הדרכים.

הקבצים בתיקייה *busSolvers* מכילים כבר שלדים של מחלקות היראכיות שיעזרו לכם במימוש, כפי שיפורט בהמשך.

### תרגילים

13. השלימו את החלקים החסרים במחלקה *busSolvers.GreedyBestFirstSolver*.

**שימו לב:** הפונקצייה מקבלת ב-*init* פרמטר שנקרא *scorer* שיורש ממחלקת *Cost*. יש להשתמש בו במתודה כדי לקבל את העלות של המעבר לכל אחד מהיורשים (הוא נשמר ב-*self._scorer*).

הריצו את סקריפט המבחן *scriptsAndExperiments/greedy.py*.

צרו בדו"ח טבלה כדלהלן, ומלאו אותה בעזרת המידע המוצג בסוף ריצת הסקריפט.

| Input file | Solution cost (total distance in KM) |
|---|---|
| TLV_5 | |
| SDEROT_50 | |
| HAIFA_100 | |
| BEER_SHEVA_100 | |

14. תנו חסם עליון וחסם תחתון הדוקים ככל שתוכלו למספר הצמתים שיפותחו בריצת האלגוריתם על בעיית האוטובוס (כשפיתוח הוא קריאה לפונקציית העוקב של *BusProblem*).
15. מהי סיבוכיות המקום של האלגוריתם כפונקציה של הקלט לבעיה?

## חלק ז' – אלגוריתם חיפוש חמדני-סטוכאסטי (20 נק')

נמשיך להשתמש בגישה חמדנית, אך במקום לבחור באופן דטרמיניסטי את המצב הקרוב ביותר (לפי המרחק האווירי), נבחר באופן סטוכאסטי אחד מ-$N$ המצבים הקרובים ביותר.

באלגוריתם אותו נממש בחלק זה, נשתמש בערך הקבוע $N = 5$ (הקבוע כבר נמצא במחלקת הקבועים).

לצורך כך יש להגדיר פונקציית הסתברות על המצבים העוקבים למצב הנוכחי.

לשם נוחות הכתיבה, נגדיר וקטור עזר $x \in [0,1]^{|succ(curr)|}$ המכיל ניקוד חיובי (גם כאן נשתמש במרחק האווירי):

$$x^t = \left[score(curr, p_1) \ , \ \cdots \ , \ score(curr, p_{|pending|})\right]$$

$$\forall pnt_i \in pending: \Pr(pnt_i) = \begin{cases} \dfrac{(x_i)^{-\frac{1}{T}}}{\sum_{pnt_h \in best\ N\ points} (x_h)^{-\frac{1}{T}}}, & pnt_i \in best\ N\ points \\ 0, & pnt_i \notin best\ N\ points \end{cases}$$

לשם יציבות נומרית נשנה את הסקאלה של הניקוד ע"י הגדרת $\alpha = \min_j x_j$ ועדכון ההסתברויות:

$$\forall pnt_i \in pending: \Pr(pnt_i) = \begin{cases} \dfrac{\left(\frac{x_i}{\alpha}\right)^{-\frac{1}{T}}}{\sum_{pnt_h \in best\ N\ points} \left(\frac{x_h}{\alpha}\right)^{-\frac{1}{T}}}, & pnt_i \in best\ N\ points \\ 0, & pnt_i \notin best\ N\ points \end{cases}$$

קל לראות שמתקבלת התפלגות חוקית (כל הסתברות היא בין 0 ל-1, וסכום כל ההסתברויות הוא 1).

למשל, עבור הוקטור $x^T$ הנ"ל והקבועים $N = 5, T = 0.5$ נקבל וקטור ההסתברויות $P$:

$$x^t = [400, 450, 900, 390, 1000, 550] \ , \ P^t \cong [0.28, 0.22, 0.05, 0.3, 0, 0.15]$$

### הגישה הסטוכאסטית

בגישה זו נרצה להריץ את האלגוריתם המון פעמים, ולהחזיר את הפלט הטוב ביותר שקיבלנו במהלך הריצה. יתכן שמספר הפעמים יהיה קבוע מראש (כמו בתרגיל בהמשך), ויתכן שנריץ את האלגוריתם ככל שיאפשרו לנו, עד שגורם חיצוני (למשל המשתמש) ידרוש מאיתנו לעצור ולהחזיר את התוצאה הטובה ביותר שקיבלנו עד כה (אלגוריתם anytime).

### הטמפרטורה

ניתן להתייחס למשתנה $T$ כמו "טמפרטורה" במערכת. ככל שהוא גדל, אנחנו מכניסים יותר "רעש" ומאפשרים בחירות יותר "הרפתקניות", וככל שהוא קטן נעדיף בחירות יותר "בטוחות". נרחיב על גישה זו בהמשך הקורס.

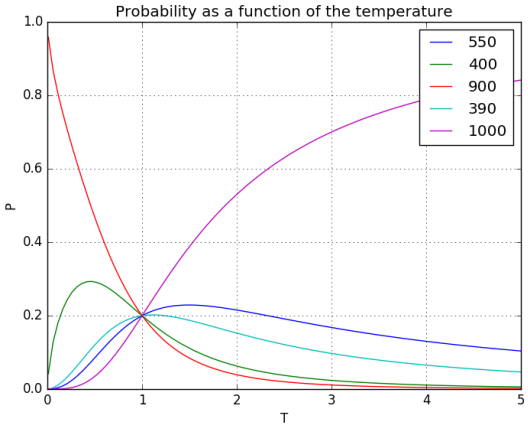
*הערה: בפונקציית ההסתברות שלעיל המשתנה $T$ לא מתאים במדוייק לפרשנות הפיזיקלית של טמפרטורה. יש פונקציות אחרות בהן זה כן מתקיים, כמו למשל פונקציית בולצמן.*

באלגוריתם אותו נממש בחלק זה, נתחיל בערך $T = 1$ ונקטין אותו פי 0.95 בכל שלב (משמע בכל פעם בו נחשב את $N$ האיברים הכי טובים). הקבועים כבר נמצאים במחלקת הקבועים.

## תרגילים

16. הוכיחו ששינוי הסקאלה לא משנה את ההתפלגות לוקטור $x$ נתון.
17. השלימו את הקוד בקובץ *scriptsAndExperiments/temperature.py*, כך שיתקבל הגרף הבא:

לכל אחד מהאיברים בוקטור הנתון בקובץ, ציירו בעקומה נפרדת את פונקציית ההסתברות שלו כפונקצייה של המשתנה $T$, עבור 100 ערכי $T$ מ-0.01 עד 5 (במרווחים שווים).
יש להוסיף מקרא לגרף.
על הפלט להיראות כמו בדוגמה הנלווית (העקומות עצמן הן להמחשה בלבד).




צרפו את הגרף לדו"ח.

18. על סמך התבוננות בגרף, או ניתוח אנליטי של פונקציית ההסתברות שהוגדרה, הסבירו איך יתנהג האלגוריתם בגבול $T \to 0$.
19. על סמך התבוננות בגרף, או ניתוח אנליטי של פונקציית ההסתברות שהוגדרה, הסבירו איך יתנהג האלגוריתם בגבול $T \to \infty$.
20. השלימו את החלקים החסרים במחלקה *busSolvers.GreedyStochasticSolver*.

בקובץ *scriptsAndExperiments/stochastic.py* כתוב קוד להרצת 150 חזרות של האלגוריתם הסטוכאסטי על קובץ הקלט *HAIFA_100.in*.

*** חשוב מאוד – הקוד הקיים משתמש במנגנון ה-Caching של מחלקת *AStar* ***

*** יש להקפיד להשתמש באותו אובייקט *AStar* לאורך כל הקוד כדי לנצל את ה-Caching ***

השלימו את *Part1* בסקריפט, כך שיוצג גרף של טיב הפיתרון כפונקציה של מספר האיטרציות.
המדד לטיב הפיתרון בו נשתמש הוא המרחק הכולל במסלול המוחזר ע"י האלגוריתם.
שימו לב, מכיוון שהאלגוריתם מחזיר בכל איטרציה את הפיתרון הכי טוב שהוא מצא עד כה, הפונקציה המוצגת אמורה להיות <u>מונוטונית יורדת</u>.
וכן יש להוסיף לגרף עקומה (קבועה) של טיב הפיתרון של האלגוריתם החמדני-דטרמיניסטי (שאינו תלוי במספר החזרות).
צרפו את הגרף לדו"ח.

*** בשלב כתיבת הקוד והדיבאג מומלץ לעבוד עם קבוע קטן מ-150 כדי לחסוך זמן ***

21. שימו לב שניתן להתייחס לתוצאה ה-$i$ של האלגוריתם הסטוכאסטי כמשתנה מקרי $X_i$. השלימו את Part2 בקובץ *scriptsAndExperiments/stochastic.py*, כך שיודפסו הממוצע המדגמי וסטיית התקן המדגמית של תוצאות אלה (המרחקים).

**נזכיר:** הממוצע המדגמי הוא $\bar{X} = \frac{1}{150}\sum_{i=1}^{150} X_i$ וסטיית התקן המדגמית היא $\tilde{\sigma} = \sqrt{\frac{1}{150}\sum_{i=1}^{150}(X_i - \bar{X})^2}$.

(ניתן לחשב זאת בקוד מפורש, או להשתמש בפונקציות ספריה מוכנות)

נסמן את תוצאת האלגוריתם החמדני-דטרמינסטי ב-$D$.

עתה נרצה **לבדוק את ההשערות** הבאות (אך ורק עבור קובץ ההזמנות הספציפי עליו הרצנו):

- השערת האפס: $H_0: \mu = D$
- השערה אלטרנטיבית: $H_1: \mu \neq D$

על פי משפט הגבול המרכזי, התפלגות הממוצע המדגמי של ההרצות מתקרבת להתפלגות $N\left(\mu, \frac{\sigma^2}{150}\right)$, כאשר $\mu$ ו-$\sigma$ הם התוחלת וסטיית התקן האמיתיות (והלא ידועות) של המשתנה המקרי.
עם זאת, בגלל שאיננו יודעים את השונות האמיתית ומסיבות סטטיסטיות שלא נלמד במסגרת הקורס, המבחן אותו נפעיל הוא מבחן $t$ (או באנגלית: t-test).
**מבולבלים?** זה בסדר. אנו הולכים להשתמש בפונקציית ספריה כדי לבדוק את ההשערה שלנו.
מה שחשוב להבין, זה שאנו מעוניינים לדעת ע"פ תוצאות אמפיריות, האם **התוחלת** של הרצה אחת של האלגוריתם החמדני-סטוכאסטי ל-dataset (קובץ הקלט) הנתון, שווה לתוצאה של האלגוריתם החמדני-דטרמיניסטי.

**ועכשיו למבחן** – קראו את הדוקומנטציה של הפונקציה scipy.stats.ttest_1samp .
כתבו את המבחן בהמשך *Part2* והדפיסו את ערך ה-P-value.
הריצו את הסקריפט הסופי (עם 150 חזרות).

**כתבו בדו"ח:**

- את הממוצע וסטיית התקן המדגמיים ואת ערך ה-P-value.
- ע"פ מה שלמדנו בתרגול הראשון, מה תהיה הכרעתכם בבדיקת ההשערות? האם נדחה את השערת האפס או לא?
- מתוך התבוננות בתשובה לנקודה הראשונה, נסו להסביר את תוצאת בדיקת ההשערות באופן אינטואיטיבי.

## חלק ח' – אלגוריתם מבוסס A* (20 נק')

בחלק זה נריץ אלגוריתם A* כדי לפתור את הבעיה עצמה שלשמה התכנסנו.

מתוך הגדרת מרחב המצבים והמימוש הכללי של A*, אין מניעה שנשתמש באלגוריתם זה כדי למצוא פיתרון אופטימלי במרחב המצבים של האוטובוס. כדי להבטיח שנקבל פיתרון אופטימלי, נפעיל את A* על מרחב המצבים של האוטובוס, עם A* על המפה (כפי שכבר הרצנו) בתור פונקציית מחיר.


נשים לב ש-A* עם היוריסטיקה אופטימית (קבילה) הוא אלגוריתם קביל, ולכן אמור למצוא את הפיתרון **האופטימלי**. מכיוון שהבעיה היא NP קשה, אנחנו לא יכולים לצפות שהאלג' יעבוד באופן כללי בזמן סביר, אפילו עם היוריסטיקות קבילות מאוד מתוחכמות (אחרת נוכיח ש-P=NP, נקבל פרס טיורינג ונלך הביתה). לכן נצמצם את השיח בחלק זה אך ורק ל-dataset הקטן ביותר שלנו – *TLV_5.in*.

### תרגילים

נזכיר כי ב-dataset יש 5 הזמנות.

22. כמה מסלולים יש בגרף המצבים מהמצב ההתחלתי למצב מטרה כלשהו? יש לכתוב מספר מפורש ולנמק בקצרה.
23. הריצו את הקובץ *scriptsAndExperiments/busAstar.py* מבלי לשנות אותו (שימו לב, ה-A* שמופעל על מרחב המפה עדיין משתמש במנגנון ה-Caching, אך אלה שמופעלים על מרחב האוטובוס לא משתמשים במנגנון זה).
מה תוצאת האלגוריתם החמדני-דטרמיניסטי (המרחק הכולל בק"מ)?
מה תוצאת האלגוריתם החמדני-סטוכאסטי (המרחק הכולל בק"מ)?
מה תוצאת A* (המרחק הכולל בק"מ) עם היוריסטיקת האפס? כמה צמתים פותחו?
24. לכל אחת מהיוריסטיקות הבאות יש לציין האם היא קבילה, ומאילו היוריסטיקות אחרות ברשימה היא מיודעת יותר. יש לנמק בקצרה.
נסמן עם $d_{L2}$ את פונקציית המרחק האווירי, ועם $d_{A^*}$ את פונקציית המרחק האמיתי (שניתן לממש באמצעות A*).

a. $h_a(s) = \arg\max_{(s_i,t_i)\in s.W} d_{L2}(s_i,t_i)$

b. $h_b(s) = \arg\max_{\substack{(s_i,t_i)\in s.W,\\ (s_j,t_j)\in s.W}} d_{L2}(s_i,t_j)$

c. $h_c(s) = arg\max_{(s_i,t_i)\in s.B} d_{L2}(s_i,t_i)$

d. $h_d(s) = \arg\max_{(s_i,t_i)\in s.W} d_{L2}(s.v,s_i)$

e. $h_e(s) = h_a(s) + h_d(s)$

f. $h_f(s) = \arg\max_{(s_i,t_i)\in s.W} d_{A^*}(s_i,t_i)$

g. $h_g(s) = \arg\max_{\substack{(s_i,t_i)\in s.W,\\ (s_j,t_j)\in s.W}} d_{A^*}(s_i,t_j)$

25. בקובץ *heuristics/TSPCustomHeuristic.py* ממשו את **אחת** מהיוריסטיקות המיודעות מהסעיף הקודם.
מחקו את פקודת ה-exit() הראשונה בקובץ *scriptsAndExperiments/busAstar.py* והריצו אותו שוב.
מה תוצאת A* (המרחק הכולל בק"מ) עם היוריסטיקה שמימשתם? מה הערך היוריסטי של המצב ההתחלתי? כמה צמתים פותחו?

## היוריסטיקה מתקדמת

נשתמש בהיוריסטיקה קבילה המבוססת על עבודה של Christofides מ-1976 מתחום אלגוריתמי הקירוב.

הרעיון הכללי: נשים לב שבמצב שאינו מצב מטרה, יש לנו רשימה של צמתי דרכים שעלינו לבקר עדיין. אם נקל את הבעיה ונניח שאנו יכולים לעבור צמתים אלה ישירות בעלות של המרחק האווירי ביניהם, נקבל קליקה לא מכוונת עם משקלים על הקשתות. ועכשיו עלינו למצוא את המרחק המינימלי שנדרש כדי לבקר בכל הצמתים האלה (ללא הגבלה על מספר הביקורים).

איך מוצאים את המרחק המינימלי לביקור בכל הצמתים של גרף לא מכוון? צודקים! עם עץ פורש מינימום!

ניתן להסיק שמשקל העפ"מ של הקליקה שתוארה הוא חסם תחתון להיוריסטיקה מושלמת מכל מצב, אך לצערנו בגרף מכוון עם אילוצים על סדר הביקור כמו שלנו, היוריסטיקה הזו עלולה להיות לא מאוד מיודעת.

בכל מקרה היא עדיפה מהיוריסטיקת האפס (שהופכת את A* ל-Uniform cost search).

## תרגיל

26. מחקו את פקודת ה-exit() השניה בקובץ *scriptsAndExperiments/busAstar.py* והריצו אותו שוב.
מה תוצאת A* (המרחק הכולל בק"מ) עם היוריסטיקה המתקדמת? מה הערך היוריסטי של המצב ההתחלתי? כמה צמתים פותחו?

**שימו לב:** ארבעת הסעיפים האחרונים יכולים לעזור לכם לוודא שהאלגוריתמים שלכם אכן עובדים כשורה. ודאו שהתוצאות שקיבלתם הגיוניות.

## חלק ט' – הגשת המטלה (5 נק')

מעבר למימוש ולדו"ח, ציונכם מורכב גם מהגשה תקינה של המטלה לפי הכללים הבאים:

- **יש לכתוב קוד ברור:**
  - לחלק למתודות היכן שנדרש.
  - קטעי קוד מסובכים או לא קריאים יש לתעד עם הערות.
  - לתת שמות משמעותיים למשתנים ולקבצים.
- **הדו"ח:**
  - יש לכתוב בדו"ח את תעודות הזהות של **שני** המגישים.
  - הדו"ח צריך להיות מוקלד במחשב ולא בכתב יד.
  - יש לשמור על סדר וקריאות גם בתוך הדו"ח.
  - יש לתת כותרות מתאימות לגרפים ולצירים (ומומלץ להשתמש ב-*plt.grid()*).
  - אלא אם נכתב אחרת, תשובות ללא נימוק לא יתקבלו.
  - יש לענות על השאלות לפי הסדר והמספרים שלהם.
- **ההגשה:**
  - יש להעלות לאתר קובץ zip בשם AI1_123456789_987654321.zip (עם תעודות הזהות שלכם במקום המספרים).
  - בתוך הזיפ צריכים להיות זה לצד זה:
    - הדו"ח הסופי בפורמט pdf, בשם: AI1_123456789_987654321.pdf.
    - תיקית הקוד AI1 שקיבלתם בתחילת המטלה, עם כל השינויים הנדרשים.

    **נא לא להכניס לזיפ את התיקייה db שבתיקייה שקיבלתם.**

חריגה מהכללים האלה עלולה לגרור הורדה של כל 5 הנקודות.

קוד לא ברור / לא עובד אף עלול להביא להורדה של נקודות נוספות בפרק בו הוא נכתב.

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

# פרק שני – שאלה תאורטית (10 נק')

יש לענות על פרק זה בסוף הדו"ח המוגש.

נתון מרחב מצבים ונתונה פונקציה היוריסטית חלקית $h$.

היוריסטיקה $h$ מעל אוסף מצבים $S$ תיקרא חלקית אם קיימת קבוצה $S' \subset S$ כך ש-$h$ אינה מוגדרת על $S'$.
ברשותכם פרדיקט $Applicable_h: S \to \{True, False\}$ המחזיר $True$ אמ"מ $h$ מוגדרת על $s$.

נניח שפונקציית המחיר חיובית וחסומה ע"י $\delta > 0$ וש-$h$ קבילה ואי-שלילית במצבים עליהם היא מוגדרת.

רוצים להפעיל A* על מרחב המצבים עם היוריסטיקה $h$, אך לא ניתן לעשות זאת מכיוון שהיא לא מוגדרת על כל המצבים. פתרון בסיסי לבעיה הוא שימוש בהיוריסטיקה החילופית הבאה:

$$h_0(h,s) = \begin{cases} h(s), & Applicable_h(s)\ is\ True \\ 0, & otherwise \end{cases}$$

א. הוכיחו: אם $h$ קבילה גם $h_0$ קבילה.

ב. נתון שמרחב המצבים הינו עץ. מצאו פתרון מיודע יותר מהפתרון הבסיסי הנתון אשר שומר על קבילות. כתבו את כל הפסאודו-קוד של האלגוריתם.

הבהרה:

- בהקשר הנוכחי, נאמר שפתרון מיודע יותר מפתרון אחר אם הוא מסתמך על היוריסטיקה מיודעת יותר.

ג. כעת לא נתון דבר על טופולוגיית מרחב המצבים. חזרו על סעיף ב'.

ד. כעת נתונה היוריסטיקה חלקית $h'$ הנותנת בדרך קסם את הערך המושלם עבור מצב ההתחלה בלבד, ואינה מוגדרת על יתר המצבים.

**הוכיחו/הפריכו:** קיים אלגוריתם קביל המשתמש ב-$h'$ וזמן ריצתו חסום מלעיל ע"י $A^*$ המשתמש בהיוריסטיקה $h_o(h',s)$ (הכוונה שאלגוריתם כזה מהווה שיפור ל-A* עם $h_0(h',s)$).