תרגיל בית 1

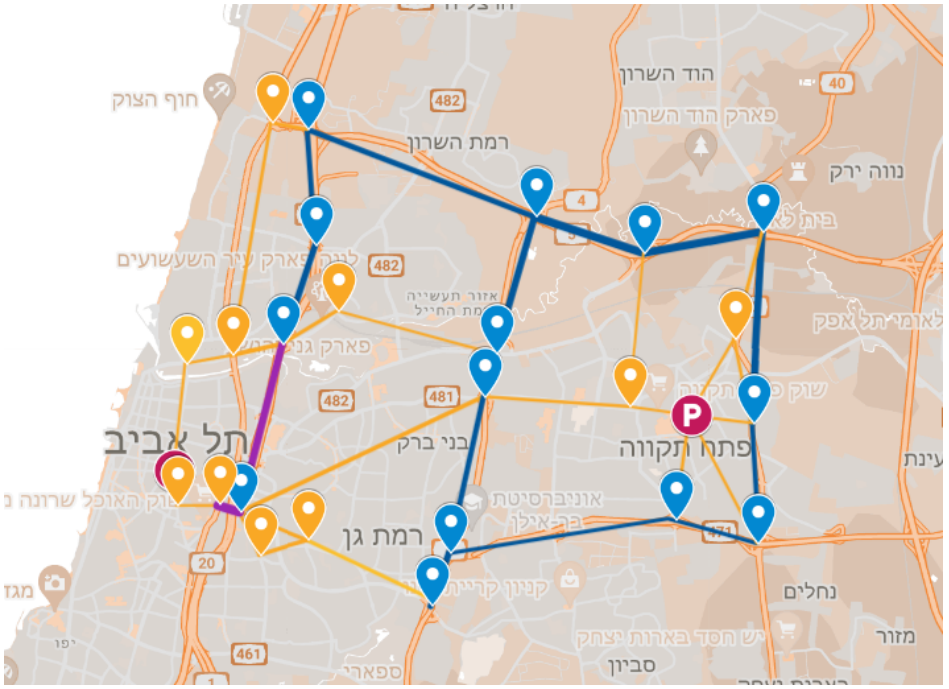
עבודה זו מסכמת את הנושאים שנלמדו בשני פרקים בכיתה:

- ☐ ***איך עובד Waze***
- ☐ ***עוד אלגוריתמים בתורת הגרפים:***
  - ☐ ***אלגוריתם Autocomplete ,***
  - ☐ ***קידוד בשיטת האפמן - Huffman Code,***
  - ☐ ***משפט 4 הצבעים***

לפני הגשת תרגילי הבית, יש לעיין ב**הנחיות להגשת תרגילי הבית.** מסמך שפורסם באתר הקורס. שאלות ובקשות בנוגע להגשה יש לפנות לעוזר ההוראה.

איך Waze עובד?

1. בתרגיל זה נכתוב קוד פייתון למציאת מסלול קצר ביותר לנסיעה מסניף הדואר הראשי בפתח תקווה לככר רבין במרכז תל אביב. לרשותכם מפה "בסיסית" עליה סומנו מספר צמתים וכבישים בהם ניתן לנסוע. יש להשתמש בתרגיל זה בנתונים הבסיסיים של המפה בלבד. אין לנווט מחוץ לכבישים המסומנים במפה כאן.



1.1. צרי גרף שיתאר את המפה והכבישים שניתן לנסוע בהם. את הערכים של הקשתות (הכבישים) נמלא במרחק הפיזי בין הנקודות בקצות הכביש. לנוחותם, הנה רשימת האורכים של כל הכבישים:

קו (כביש)	21	23	24	25	26	28	29	30	31	32	35	36	37
אורך (ק"מ)	2.7	0.6	2.1	1.0	5.0	2.4	2.3	2.8	4.8	2.2	0.9	2.0	0.7

קו (כביש)	38	39	40	41	43	46	47	48	49	51	52	53	54
אורך (ק"מ)	4.7	2.9	0.9	0.9	5.3	1.0	1.4	0.9	2.2	1.7	1.1	2.4	3.0

קו (כביש)	56	57	60	61	62	63	64	66	67	69	70	71
אורך (ק"מ)	3.2	1.3	0.8	0.4	3.4	3.2	2.9	2.1	4.5	2.2	1.6	1.7

צרי את הגרף בצורה של אוסף קשתות. **כל קשת** מוגדרת על ידי:

- ☐ נקודת תחילת הקשת (שם הצומת בו היא מתחילה)
- ☐ נקודת סיום הקשת (שם הצומת בו היא מסתיימת)
- ☐ צבע הקשת: צהוב, כחול או סגול (לרשום את הצבע)
- ☐ אורך הקשת לפי המרחק האוירי שלה כמוסבר לעיל
- ☐ מהירות נסיעה כשאין עומסים (30 קמ"ש לכבישים צהובים. 60 קמ"ש לכבישים כחולים וסגולים)
- ☐ מהירות נסיעה כשיש עומסים (20 קמ"ש לכבישים צהובים. 10 קמ"ש לכבישים כחולים. 5 קמ"ש לכבישים סגולים).

***הערה**: כדי לא לסבך את התרגיל, הקשתות לא מכוונות כלומר, נסיעה בין הנקודות של קשת לוקחת אותו זמן לא חשוב מה כיוון הנסיעה. במציאות ובמפות של וייז יש חשיבות רבה לכיוון הנסיעה. למשל: בבוקר יש עומס רב בכבישים בכיוון הכניסה לתל אביב אבל בכיוון ההפוך (יציאה מתל אביב), אין עומסים. אחה"צ התמונה הפוכה ויש עומסים בכיוון היציאה מהעיר ואין עומסים בכיוון הכניסה לעיר. בתרגיל זה לא נממש את כיווניות הכבישים.*

1.2 מצאי את המסלול הקצר ביותר להגעה מ-**סניף הדואר הראשי בפתח תקווה** אל **ככר רבין בתל אביב** (שתי הנקודות מסומנות בסימן "P" במפה) בשעות בהן אין עומסים בכבישים. לשם כך עליך לבצע חיפוש בשיטת דיקסטרה על הגרף ולמצוא את המסלול המהיר ביותר תוך שימוש בערכים של נסיעה ללא עומסים בכביש.

*שימי לב: אם הערך "מהירות נסיעה כשאין עומסים" = 30 קמ"ש והערך **אורך הקשת** הוא 3 ק"מ אז זמן הנסיעה בקטע הכביש הזה יהיה 0.1 שעה (0.1=3/30). הקפידו לעבוד כל הזמן עם יחידות של שעה כדי לא להתבלבל בחישובים.*

יש להגיש:

1. מסלול הנסיעה הטוב ביותר (אפשר לסמן אותו על המפה באופן גרפי או לרשום את רשימת הצמתים דרכם נוסעים, החל מסניף הדואר ועד ככר רבין).
2. זמן הנסיעה המשוער (*ETA: Estimated Time of Arrival*)

1.3 יש לחזור על התרגיל ולמצוא את המסלול המהיר ביותר כשיש עומסים בכבישים.

יש להגיש: 1. מסלול הנסיעה הטוב ביותר. 2. זמן הנסיעה המשוער.

1.4 לסיום תרגיל זה, נניח שקרתה תאונה ואחד הכבישים במסלול שמצאתם בסעיף 1.3 נחסם בגלל התאונה. חיזרי על סעיף 1.3 עם הנתון שקטע הכביש המסויים הזה נחסם, ותנו לקוד שלכם למצוא מסלול חלופי עם זמן נסיעה מינימלי.

***הערה**: אפשר להריץ בדיוק את הקוד מסעיף קודם אם רק משנים ערך בודד בגרף, כדי שייצג את העובדה שקרתה תאונה ואי אפשר לנסוע בכביש הזה יותר. חישבו מה הדרך המעשית הפשוטה ביותר לעשות זאת.*

יש להגיש: 1. מסלול הנסיעה הטוב ביותר. 2. זמן הנסיעה המשוער.

2. בכיתה ראינו מימוש של אלגוריתם דייקסטרה על מפה (גרף) של OSM. עם פונקציה בשם: `dijkstra0`.
עייני בפונקציה ועני על הסעיפים הבאים:

2.1. הוסיפי לקוד הפונקציה `dijkstra0` שורה בודדת שתראה **כמה צמתים ביקר האלגוריתם בגרף, עד שמצא את המסלול האופטימלי.**

2.2. הוסיפי לקוד שורה שתציג את אורך המסלול שנבחר לנסיעה.

2.3. הוסיפי לקוד **הדפסה של המסלול שנבחר לנסיעה** (בסיום תהליך חיפוש המסלול הטוב ביותר). צריך להדפיס רשימת שורות שכל שורה כוללת את הערכים הבאים:
(1) הצומת ממנה יוצאים (לכל צומת יש ATTRIBUTES שאחד מהם הוא שם הצומת. אם יש שם - להדפיס אותו. אם אין לצומת שם - להדפיס את מספר הצומת (ה-ID שלה).
(2) קטע הכביש בו נסע מהצומת הזו. כנ"ל - אם יש שם, לבחור בו. אם אין - מספר מזהה של הקשת המייצגת את קטע הכביש בו ניסע בגרף.
(3) משקל הקשת
(4) סך הכל המרחק המצטבר מנקודת היציאה (סך כל המשקלים של הקשתות מנק. היציאה עד כה)

2.4. כתבי פונקציה `dijkstra1` שמחפשת מסלול קצר ביותר עד שהצומת הבאה לבדיקה במחסנית ה-Priority Queue מכילה ערך (מרחק נסיעה עד כה) גדול יותר מהערך ששמור נכון לרגע בצומת של נקודת הסיום. שימי לב, גישה זו ממשיכה לחפש ולדייק מסלולים וזמני נסיעה גם אחרי שהגענו לנק. הסיום. יש להגיש: (1) **הקוד של הפונקציה**. (2) **פלט של המסלול שנבחר**, כמו שעשית בשאלה 2.
(3) **ציון מספר הצמתים** סה"כ שנבדקו במהלך החישוב, כפי שעשית בשאלה 2.

2.5. השווי את ביצועי הפונקציה `dijkstra0` והפונקציה `dijkstra1` באופן הבא:
השתמשי בקטע הקוד בו קראנו לפונקציות מספריית OSMNX. במקום השורות הבאות בקוד:

```
place        = 'Tel Aviv, Israel'  # Select a Map
start_lnglat = (34.794, 32.070)     # Google as start point
end_lnglat   = (34.802, 32.115)     # Tel Aviv University as end point
```

כיתבי לולאה שרצה N פעמים (צריך לבחור N בין 100 ל-1000, כך שהקוד ירוץ כדקה, לא הרבה יותר). בכל פעם, מגרילים נק. התחלה וסיום בנקודות אקראיות במפה של תל אביב (רמז: יש למצוא את נקודות `lat` , `lng` המקסימום והמינימום במפה של תל אביב ולהגריל נקודות אקראיות "בתוך" המפה). אח"כ אפשר להשתמש בדיוק בקוד מהכיתה:

```
graph = ox.graph_from_place(place, network_type = 'drive')
# find the nearest node to the start/end locations
orig_node = ox.nearest_nodes(graph, *start_lnglat)
dest_node = ox.nearest_nodes(graph, *end_lnglat)
```

בשלב זה יש לקרוא לפונקציה `dijkstra0` והפונקציה `dijkstra1` במקום לפונקציה `shortest_path`.

איספי את כל N התוצאות משתי הפונקציות והציגי את התשובה הממוצעת של כל פונקציה. האם התוצאות זהות? אם לא - מי יעילה יותר? האם אפשר להסביר את התוצאות שקיבלת?

3. האלגוריתם של WAZE טועה לפעמים (די הרבה...) בשיערוך זמן ההגעה ליעד (ETA, Estimated time of arrival) מהסיבה הבאה: ETA מחושב ביחד עם המסלול הטוב ביותר לפני תחילת הנסיעה. אם במהלך הנסיעה נוצרים עומסים ופקקים בדרך שנבחרה, או ההיפך, משתחררים פקקים שהיו בזמן החישוב אבל עד

שהגענו לאותה נקודה, הפקק התפוגג, נקבל הפרשים של דקות רבות, לפעמים גם חצי שעה ויותר, בין הזמן המשוער לזמן הנסיעה בפועל.

א. מדוע יש קושי לחשב נכון את ETA בשלב חישוב המסלול? ל-WAZE יש סטטיסטיקה של העומסים בכל קטע כביש ובכל שעה ודקה לאורך היממה. מדוע אי אפשר לחשב בדיוק רב אם יהיה פקק כשנגיע לנקודה או שעד אז הוא ישתחרר (וגם להיפך, אם יווצר פקק בהמשך הדרך אם בתחילת הנסיעה אין שם עומסים כלל)?

ב. הצע/י שיטה מעשית, אפשר גם שיטה לא מושלמת, לתת הערכה מדויקת ככל שניתן (מעשית...) לזמן ההגעה ליעד שתתחשב בנתוני העומסים המשתנים עם הזמן במפה. הנח/י שיש לכם את כל המידע כל זמן הנסיעה בכל מקטע כביש במפה, בכל שעה של היום.

אלגוריתם Auto-complete

בשאלות כאן, נסתמך על הטקסט הבא (טקסט מלא של ספר בשפה האנגלית) כדי לבנות את טבלאות השכיחויות האופיניות לשפה האנגלית.
שם הספר: BURNING DAYLIGHT by Jack London
לינק לנוסח המלא שלו באינטרנט (גישה חופשית): https://www.fulltextarchive.com/book/Burning-Daylight/

4. השתמשי בקוד שראינו בכיתה (ומצורף באתר הקורס לנוחותכם) והשלימי בו את הפונקציה Autocomplete שתציעה השלמה אוטומטית של מילים באופן הבא (אנו עושים לכם קצת "הנחות" במימוש, כדי שנתעסק בעיקר במהות ולא בענייני UI וכיו"ב):
   4.1. הקוד יקלוט (אפשר בפקודת input פשוטה) מילה מהמשתמש.
   4.2. הקוד יעשה סימולציה כאילו ההדפסה נעשית אות-אות, למשל: אם המשתמש הכניס את המילה Hello, הקוד יציג השלמת טקסט אחרי הקלדת "H" בלבד, ואח"כ השלמה (אולי שונה...) אחרי שהקלט המשתמש הוא "HE" ואח"כ שוב השלמה (אולי שוב שונה) אחרי שהקלט הוא "HEL" וכו'.
   4.3. בכל שלב לעיל, התכנה תציג הצעת השלמה למילה הכי סבירה מתוך חיפוש הסתברויות בעץ - Trie - שפיתחנו בכיתה.

שימו לב: *כדי לבחור מילים שהכי סביר שהמשתמש יקליד, צריך לחשב את ההסתברויות למילים השונות בעץ מתוך הנתונים שראינו בכיתה שהם* ***מניה*** *(counter) של מספר הפעמים שמגיעים לצומת הזאת בעץ. למשל: צריך להסתכל על כל הילדים (children) של node, לסכם את סך הפעמים שכל node היה במילון (הערך counter של אותו node), ואז, לחלק לכל node את ה-counter שלו בסכום כל ה-counter של כל הילדים המקבילים לו.*

טיפ: *פונקציות שיכולות לעזור לכם בעיבוד הנתונים בעץ הן:* node.children , node.child , node.char , node.counter

דוגמא: *לפלט אפשרי (לא אמיתי) אם המשתמש הכניס את המילה TRY:*

1) *T: To*
2) *TR: Tree*
3) *TRY: Try*

קידוד בעזרת אלגוריתם האפמן Huffman Coding

בשאלות כאן, נסתמך על הטקסט הבא (טקסט מלא של ספר בשפה האנגלית) כדי לבנות את טבלאות השכיחויות האופיניות לשפה האנגלית.

שם הספר: BURNING DAYLIGHT by Jack London

לינק לנוסח המלא שלו באינטרנט (גישה חופשית): https://www.fulltextarchive.com/book/Burning-Daylight/

5. השתמשי בטקסט איתו בנינו את העץ באלגוריתם Autocomplete כדי לחשב את השכיחויות של כל אותיות ה-ABC, כולל סימן רווח, בשפה האנגלית. להזכירכם, סיננו את הטקסט כך שהוא מכיל רק אותיות בפורמט lowercase ורווחים.

5.1. כיתבי את טבלת השכיחויות שיצאה לכם בדף ההגשה של שעורי הבית והכניסי אותה גם כמערך בקוד שאת כותבת. (יש לספור רק את האותיות a עד z ואת סימן הרווח).

5.2. כעת, כתבי קוד ליצירת עץ Huffman שייצג את האותיות וסימן הרווח מהסעיף הקודם.

5.3. קודדי את 1000 האותיות הראשונות בקובץ הטקסט שלנו (לינק לספר לעיל). כמה ביטים נדרשו לקידוד בשיטת Huffman? מה היעילות יחסית לשמירת האותיות בקוד ASCII שדורש 8 סיביות לכל סימן (סה"כ 8000 סיביות לשמירת 1000 סימנים)?

5.4. בחרי טקסט אחר כבסיס להכנת טבלת השכיחויות. אפשר לבחור ספר אחר ממבחר הספרים כאן: https://www.fulltextarchive.com/ או כל טקסט שתבחרי באינטרנט שאורכו לפחות 10,000 מילים בשפה האנגלית. הכיני טבלת שכיחויות כמו בסעיף לעיל והדפיסי את שתי הטבלאות, זו מהסעיף הקודם וזו כאן לפי סדר יורד של שכיחויות האותיות. האם יש הבדלים בין שני המקרים? מה אפשר ללמוד מכך על הרגישות לטקסט ממנו מכינים את טבלת השכיחויות?

--- סוף התרגיל ---