Sagiva13@gmail.com

Sagiv Antebi

318159282

AI – EX1

1.1.תשובה:
כל שורה בקובץ מכילה:
עמודה 1: אינדקס צומת
עמודה 2: קו רוחב צומת
עמודה 3: קו אורך צומת
עמודה -4: לינקים של הצומת
לדוגמא: עבור שורה מספר 1:

1 29.44148 34.8417302 2@114@2 0@32@2

<u>נקרא זאת כך:</u>

מצומת מספר 1, שממוקמת בקו רוחב של 29.44148 וקו אורך של 34.8417302, ניתן להגיע לצומת 2 במרחק של 114, וסוג כביש המהיר הוא 2. כמו כן ניתן להגיע לצומת 0 במרחק של 32, וסוג הכביש הוא 2.

<u>הסבר תשובה:</u> load_map_from_csv קוראת לפונקציה make_junction_ על כל שורה בטבלת csv שלנו.
לאחר מכן make_junction_ קוראת לפונקציה make_link_ שבעצם מפצלת את הקלט שלנו מrawו לארבעה פרמטרים:
1. האינדקס
2.קו רוחב (float)
3.קו אורך (float)
4.שאר השורה

כאשר עבור הפרמטר הרביעי היא עוברת על כל עמודה ומפצלת את הכתוב (ע"י פיצול לפי התו '@'), ומכניס למערך link_params.
וכעת ניצור Link כשערכיו יהיו:
Link – אינדקס הi שקיבלנו
source – אינדקס הi שקיבלנו
target – מספר int שמייצג את היעד
distance – מרחק
highway_type – סוג כביש מהיר
link_params – טאפל של שני פלאוטים – שבעצם זה יהיה ה
ולבסוף tupple של Link_traffic_params – כך שהראשון מייצג תדירות cos והשני תדירות sin.

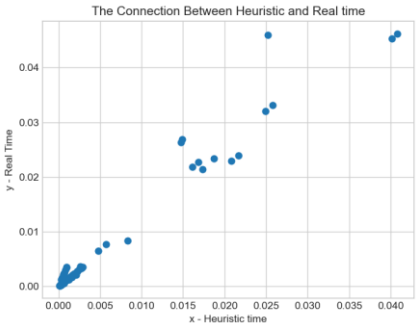
לאחר מכן מחזירים את הLink בחזרה לפונקציה make_junction_ שבונה junction מהLink שיתקבל ושאר הפרמטרים שקיבלה. נבנה כך רשימה של לינקים.
ולבסוף נקבל אובייקט מסוג צומת – אשר ערכיו הם:
1. האינדקס
2.קו רוחב (float)
3.קו אורך (float)
4.ה Link שיצרנו
לבסוף לאחר שיצרנו את הצמתים מקובץ הcsv שלנו נעבור ונבנה מפה שמפה בין מספר i לבין צומת שבה נאתחל את כל הצמתים שמקושרים לצומת הספציפית הזו.

8. תשובה:
הפונקציה שבחרתי הינה חישוב המרחק לפי compute_distance חלקי 110 – המהירות המקסימלית שניתן בכל הכבישים הנתונים.נשים לב שכמו כן החישוב קו הוא אווירי בין הsource (הראשוני) - לבין הtarget – לפי הproblem.
הקוד:

```
def heuristic_function(lat1, lon1, lat2, lon2):
    return compute_distance(lat1, lon1, lat2, lon2) / 110
```

טענה: הפונקציה ההיוריסטית שבחרתי אכן קבילה.
הוכחה: ניזכר בהגדרה: יוריסטיקה קבילה h מעריכה "אופטימית" את מחיר המסלול מכל מצב s למצב מטרה .
כלומר $\forall s \in S : 0 \leq h(s) \leq h^*(s)$
כמו כן בגלל שחישבנו מרחק אווירי בין שתי נקודות,ההתחלה והסיום,ולאחר מכן חילקנו במהירות הגבוהה ביותר, בוודאי שהוא הזמן הקצר ביותר.
נניח בשלילה שקיים מסלול שניתן לקחת באותם דרכים וזמנו יהיה קטן משלנו,
אזי הדרך חייבת תהיות קו אווירי (משום שזה לפי חישוב מתמטי הדרך הקצר הביותר), אך משום שזה אותו מסלול שלנו אזי עליו לסוע מעל 110 קמ"ש, אך זו סתירה להנחה שלא קיימת מהירות גבוהה מ110.
ולכן לכל פונקציה אחרת שנבחר היא חייבת להיות גדולה או שווה לשיערוך של הפונקציה ההיוריסטית שלנו. מ.ש.ל.

9. מסקנה ברורה מהגרף היא שכל הנקודות נמצאות במצב בו הערך בציר הY יותר גדול מהערך בציר הX.
כלומר בגלל שהפונקציה ההיוריסטית שלנו תמיד מחשבת את המצב הטוב ביותר – בו נוכל לסוע במהירות הגבוהה ביותר באותו המרחק אזי כמובן שהזמן המשוער בציר הX יהיה קטן מאשר הזמן האמיתי המוצג בציר הY.



10. לא, המסלול לא יהיה אופטימלי. משום שבחישובנו אנו מחשבים לפי המהירות המקסימלית של סך כל הכבישים.
כלומר, בגלל שהעומס בכביש יגרום למהירות נמוכה יותר מהמהירות המקסימלית האפשרית בכביש, יתכן שקיים מסלול אחר שבו הזמן הממוצע יקח פחות זמן בהתחשב בנתונים החדשים.

13. בחרתי להריץ עם 10 הבעיות הבאות:

19,21 18,25 17,21 16,18 15,19 14,16 11,13 10,13 5,7 4,6

זמני ריצה ממוצעים:

```
AVG RUN TIME:
UCS: 0:00:00.125980
A-STAR: 0:00:00.026894
IDA-STAR: 0:00:00.741215
```

כמו שציפינו האלגוריתם של A-STAR רץ בזמן הקצר ביותר.
A-STAR רץ בזמן הקצר ביותר משום שהוא מקל מבחינת פונקציית המשקל ובוחר בצורה טובה יותר מUCS.
עם זאת, נבחין כי זמני הריצה של UCS וA-STAR דומים מאור וזה בגלל שהאלגוריתם זהה, פרט לשינוי המשקולות של תור העדיפויות.
נבחין כי זמן הריצה של IDA-STAR הארוך ביותר, וזאת מכיוון שבאלגוריתם זה אנחנו חוסכים במקום על חשבון זמן ריצה, זה נובע מיצירת העץ בכל פעם מחדש של ריצת האלגוריתם DFS_Counter.