מבוא לבינה מלאכותית

תרגיל בית 1

מגישים:

אביב כספי 311136691

יקיר יהודה 205710528

תרגיל 1:

עלינו למצוא ביטוי מתמטי למספר הסידורים החוקיים העונים על האילוצים:

נתחיל מכך שיש לנו 2k צמתים בהם עלינו לבקר, כלומר עלינו למצוא את מספר הסידורים החוקיים ל2k צמתים אלה.

נניח תחילה כי אין לנו אף אילוץ למעט הופעת כל הצמתים בסידור, כלומר יש לנו !(2k) סידורים אפשריים.

כעת נוסיף אילוץ יחיד כי צומת ההורדה של חבילה מס' 1 תופיע אחרי צומת האיסוף של חבילה זו.
כעת נשים לב כי כל הסידורים החוקיים שהיו לנו, מתחלקים לשתי קבוצות שוות, אחת בה צומת ההורדה נמצאת אחרי צומת האיסוף והשנייה בה צומת ההורדה נמצאת לפני צומת האיסוף. (לכל סידור יש סידור זהה בו סדר שתי הצמתים הללו הפוך ולכן גודל הקבוצות שווה)
כלומר כעת יש לנו $\frac{(2k)!}{2}$ סידורים חוקיים.

כעת נוסיף אילוץ נוסף וזהה לאילוץ הקודם אך כעת עבור חבילה 2.
בצורה דומה, מספר הסידוריים החוקיים הקודם שלנו מתחלק לשתי קבוצות שוות, כך שאחת הקבוצות מקיימת את האילוץ והשנייה לא.
ולכן שוב נקבל כי מספר הסידורים החוקיים שלנו מתחלק ב2 כלומר $\frac{(2k)!}{2^2}$ סידורים חוקיים.

נשים לב כעת כי הוספה של אילוץ מסוג זה מחלקת לנו את מספר הסידורים החוקיים ב-2 לפי ההסבר מעלה. ולכן אם נסתכל על המקרה שלנו, בו יש k אילוצים כאלה, (עבור כל חבילה צריך שיתקיים כי צומת ההורדה מופיעה אחרי צומת האיסוף), מספר הסידוריים החוקיים שנקבל הוא :

$\frac{(2k)!}{2^k}$ **סידורים חוקיים.**

תרגיל 2:

k	#possiblePaths	$\log_2(\#possiblePaths)$	Calculation time
5	113400	16.79	< 1 sec
8	$8.1729 \cdot 10^{10}$	36.25	76.11 sec
9	$1.25046 \cdot 10^{13}$	43.5	3.2349 hours
10	$2.37588 \cdot 10^{15}$	51.08	0.07016 years
11	$5.48828 \cdot 10^{17}$	58.93	16.208 years
15	$8.09487 \cdot 10^{27}$	92.71	239058 million years

תרגיל 3:

ערך מקסימלי לדרגת יציאה: k

ערך מינימלי לדרגת יציאה : 0

עבור המצב ההתחלתי, בו המשאית ריקה ולא הרמנו אף חבילה.
יש לנו k אופרטורים שניתן להפעיל וכל אחד יביא אותנו למצב אחר (ניתן להרים כל חבילה שהי ולא ניתן להוריד אף חבילה).

בכל מצב אחר, מספר האופרטורים המקסימלי שנוכל להפעיל במצב אידיאלי (בו יש מקום במשאית) הינו מספר החבילות שעדיין לא הרמנו + מספר החבילות שנמצאות במשאית.
כלומר נוכל לעבור לצומת בה לא היינו עדיין ולהרים חבילה, או לעבור לצומת שלא היינו עדיין ולהוריד חבילה שנמצאת במשאית. לא יתכנו אופרטורים נוספים.

נשים לב כי עבור כל חבילה שעדיין לא הורדנו, נוכל להפעיל רק אופרטור יחיד, או להרים אותה אם עדיין לא הרמנו, או להוריד אותה. ובגלל שיש רק k חבילות במקרה ה"טוב" ביותר יהיו לנו k אופרטורים שנוכל להפעיל. (הכוונה במקרה הטוב ביותר, היא למקרה בו עדיין לא הורדנו אף חבילה ולכן לכל חבילה ניתן להפעיל אופרטור כלשהו)
לכן דרגת יציאה מקסימלית הינה k.

על פי הגדרת האופרטורים, נוכל להפעיל אופרטור להעמסה של חבילה, רק אם לא העמסנו את החבילה בעבר ואם יש מקום במשאית.
ונוכל להפעיל אופרטור הורדה של חבילה, רק אם החבילה נמצאת במשאית.

כלומר אם נסתכל על מצב סופי בו מתקיים כי המשאית ריקה והורדנו את כל החבילות שקיימות, נשים לב כי לא נוכל להפעיל אף אופרטור מפני שביקרנו בכל הצמתים ואין לנו אף חבילה להוריד או להרים.
לכן דרגת היציאה של מצב סופי הינה 0 ולכן מינימלית.

תרגיל 4:

לא ייתכנו מעגלים במרחב המצבים שלנו.

על פי הגדרת האופרטורים האפשריים, ניתן לעבור לנק' איסוף כלשהי, אם לא אספנו את החבילה בעבר (כלומר, אם לא ביקרנו בנק' זו) וניתן לעבור לנק' הורדה כלשהי, רק אם החבילה המתאימה נמצאת במשאית (אחרי שעוברים את נק' ההורדה בהכרח החבילה הורדה מהמשאית, כלומר לא ניתן לבקר בנק' זו שוב).
כלומר אם הגדרת המצבים שלנו היא $(curLoc, Loaded, Dropped)$ אזי הערך של curLoc אינו יכול לחזור על עצמו במסלול כלשהו, מפני שאז נבקר באותה נק' פעמיים ולפי הגדרת האופרטורים, דבר זה אינו ייתכן. ולכן לא ייתכנו מעגלים במרחב המצבים.

תרגיל 5

מספר המצבים במרחב הינו :

$$1+\sum_{d=0}^{k}\binom{k}{d}\sum_{p=0}^{k-d}\binom{k-d}{p}(d+p)=1+2\cdot 3^{k-1}k$$

הסבר לחישוב :

נסתכל על מבנה המצבים שלנו : $(curLoc, Loaded, Dropped)$

כעת, נבדוק את כל אפשרויות לכל אחד מהחלקים במבנה המצב.
עבור Dropped יכולים להיות k+1 גדלים אפשריים לקבוצה זו (כאשר לא הורדנו אף חבילה, וכל מצב אחר עד אשר הורדנו את כל החבילות). עבור גודל נתון של Dropped d ישנם $\binom{k}{d}$ אפשרויות בחירה של איזה חבילות נמצאות בdropped .
עבור מספר זה של Dropped לLoaded ייתכנו k-d+1 גדלים אפשריים לקבוצה זו (כאשר אין אף חבילה על המשאית, וכל מצב אחר עד אשר הרמנו את כל החבילות שנותרו k-d). עבור גודל נתון של Loaded p ישנם $\binom{k-d}{p}$ אפשרויות בחירה של איזה חבילה נמצאת בloaded (נבחר p חבילות מתוך החבילות שנותרו k-d .
עבור מספר זה של Dropped ו Loaded ייתכנו d+p אפשרויות לערך של curLoc, מפני שעל פי הגדרת המצבים, curLoc חייב להיות מיקום איסוף של חבילה בLoaded או מיקום הורדה של חבילה בDropped .

לכן נקבל כי מספר המצבים האפשריים הוא : $\sum_{d=0}^{k}\binom{k}{d}\sum_{p=0}^{k-d}\binom{k-d}{p}(d+p)$. נשים לב כי הוספנו 1 למספר בהתחלה וזה נובע בגלל המצב ההתחלתי אשר אינו נכלל בחישוב הקודם.

כל המצבים שתיארנו במרחב זה בו מתקיים $TruckCapacity = \infty$ הינם ישיגים.
זה נובע מכך שעבור מצב אקראי כל שהו מתוך המצבים שתוארו נוכל לשחזר מסלול המגיע למצב זה מהמצב ההתחלתי.

לדוג' עבור מצב כלשהו (c,L,D) כאשר בה"כ נניח כי c הינה נק' איסוף של חבילה ב L. נוכל להגיע למצב זה מהמצב ההתחלתי בכך שתחילה נאסוף את כל החבילות שנמצאות ב D , לאחר מכן נוריד את כל החבילות בנק' ההורדה שלהן, כעת מתקיים כי אנחנו במצב (a,0,D) כאשר a נק' כלשהי וLoaded הינה קבוצה ריקה. לאחר מכן נאסוף את כל החבילות שנמצאות ב L כך שהחבילה האחרונה שנאסוף היא החבילה בנק' c .
כלומר בסיום המסלול נגיע למצב (c,L,D) ולכן כל מצב המתאים להגדרות שהוגדרו בתרגיל ושספרנו מעלה, ישיג מצומת ההתחלה.

תרגיל 6

לא ייתכנו בורות ישיגים מהמצב ההתחלתי שאינם מצבי מטרה במרחב המצבים.

נניח בשלילה כי קיים בור כזה והוא מהצורה (c,L,D) .
אם $L \neq \emptyset$, אזי קיימת חבילה במשאית, ולכן לפי הגדרת התרגיל נוכל להפעיל את אופרטור הפריקה של חבילה זו, ולכן נוכל לצאת מהבור בסתירה להנחה. לכן מתקיים $L = \emptyset$.
אם $|D| \neq k$, אזי קיימת חבילה שעדיין לא הורדנו, ולכן לפי הגדרת התרגיל נוכל להפעיל את אופרטור האיסוף של חבילה זו (אנחנו מניחים כי יש מקום במשאית לחבילה החסרה כי המשאית ריקה, כלומר מניחים כי אין חבילה גדולה מגודל המשאית), ולכן נוכל לצאת מהבור בסתירה להנחה. כלומר מתקיים $|D| = k$.
כלומר אנחנו נמצאים במצב בו סיפקנו את כל החבילות שיש ולכן אנחנו נמצאים במצב מטרה בסתירה להנחה. (אנו מניחים כי c הינה נק' הורדה של אחת החבילות, מפני שכל נק' אחרת לא קיימת לפי הגדרת המצבים שלנו).

כלומר לא ייתכנו בורות ישיגים מהמצב ההתחלתי שאינם מצבי מטרה.

תרגיל 7

כל מסלול ממצב ההתחלה למצב סופי הינו מאורך 2k (מספר הקשתות במסלול).

מצב סופי מוגדר כך (c,L,D) כאשר מתקיים $|D| = k , L = \emptyset$ וגם c נק' הורדה של אחת החבילות.
עבור כל חבילה שהורדנו, היה עלינו לעבור בשתי נק', אחת נק' איסוף והשנייה הורדה של אותה חבילה. בהנחה כי כל הנק' זרות, נקבל כי עלינו לעבור ב2k נק' על מנת להגיע למצב בו הורדנו את כל החבילות. בנוסף לכך התחלנו ממצב התחלתי ולכן מסלול לנק' סיום חייב להכיל לפחות 2k+1 צמתים ולכן 2k קשתות (אורך מינימלי הינו 2k).
בנוסף לפי הגדרת התרגיל , לא ייתכן כי נעבור בנק' בהן היינו, ולא ייתכן כי נעבור בנק' שאינה אחת מ2k+1 הנק' הנתונות, ולכן מסלול לנק' הסיום חייב לעבור בלא יותר מ2k+1 צמתים כלומר 2k קשתות (אורך מקסימלי הינו 2k) ולכן נקבל כי אורך כל מסלול ממצב ההתחלה למצב סופי הינו באורך 2k.

תרגיל 8

גרף המצבים שלנו הינו גרף DAG , כלומר גרף ללא מעגלים.

נשים לב כי הגרף שלנו אינו עץ מפני שניתן להגיע לאותו מצב ממסלולים שונים, לדוג' על ידי איסוף החבילות 1,2,3 בסדר הבא $3 \to 2 \to 1$ או בסדר $3 \to 1 \to 2$.
בשני המצבים הסופיים הנק' הנוכחית היא צומת האיסוף של חבילה 3, ותכולת המשאית זהה וכך גם החבילות שהורדנו, כלומר ישנן מצבים אשר ניתן להגיע אליהן ממסלולים שונים ולכן הגרף אינו עץ.
בנוסף הוכחנו כי הגרף לא מכיל מעגלים, כלומר נקבל כי גרף המצבים שלנו הינו גרף DAG.

תרגיל 9

נגדיר את פונק' העוקב בצורה הבאה :

$$Succ(s) = \left\{ (c,L,D) \middle| \begin{array}{c} \exists p, L = s.Loaded \cup \{p\} \wedge \\ D = s.Dropped \wedge \\ c = p.pick \wedge \\ p \notin s.Loaded \cup s.Dropped \wedge \\ p.pkgs \le truck.MaxCapacity - \sum_{d \in s.Loaded} d.pkgs \end{array} \right\}$$
$$\cup \left\{ (c,L,D \middle| \begin{array}{c} \exists d, L = s.Loaded \ \setminus \{d\} \wedge \\ D = s.Dropped \cup \{d\} \wedge \\ c = d.drop \wedge \\ d \in s.Loaded \end{array} \right\}$$

הסבר : חילקנו את המצבים העוקבים לשתי קבוצות, הקבוצה הראשונה היא קבוצת המצבים העוקבים לאחר שאספנו חבילה חדשה, אשר לא אספנו או הורדנו לפני, יש מקום במשאית לחבילה והמיקום הנוכחי החדש הינו נק' האיסוף של החבילה.

הקבוצה השנייה הינה קבוצה המצבים העוקבים לאחר שהורדנו חבילה, אשר אספנו בעבר ולא הורדנו עדיין.

תרגיל 10

f. פלט התוכנית :

```
Solve the map problem.
StreetsMap(src: 54 dst: 549)          UniformCost
time:   1.07   #dev: 17354   |space|: 17514    total_g_cost:
7465.52560   |path|: 137   path: [   54  ==>     55  ==>     56  ==>
57  ==>     58  ==>     59  ==>     60  ==>  28893  ==>  14580  ==>
14590  ==>  14591  ==>  14592  ==>  14593  ==>  81892  ==>  25814
==>     81  ==>  26236  ==>  26234  ==>   1188  ==>  33068  ==>
33069  ==>  33070  ==>  15474  ==>  33071  ==>   5020  ==>  21699
==>  33072  ==>  33073  ==>  33074  ==>  16203  ==>   9847  ==>
9848  ==>   9849  ==>   9850  ==>   9851  ==>    335  ==>   9852  ==>
82906  ==>  82907  ==>  82908  ==>  82909  ==>  95454  ==>  96539
==>  72369  ==>  94627  ==>  38553  ==>  72367  ==>  29007  ==>
94632  ==>  96540  ==>   9269  ==>  82890  ==>  29049  ==>  29026
==>  82682  ==>  71897  ==>  83380  ==>  96541  ==>  82904  ==>
96542  ==>  96543  ==>  96544  ==>  96545  ==>  96546  ==>  96547
==>  82911  ==>  82928  ==>  24841  ==>  24842  ==>  24843  ==>
5215  ==>  24844  ==>   9274  ==>  24845  ==>  24846  ==>  24847  ==>
24848  ==>  24849  ==>  24850  ==>  24851  ==>  24852  ==>  24853
==>  24854  ==>  24855  ==>  24856  ==>  24857  ==>  24858  ==>
24859  ==>  24860  ==>  24861  ==>  24862  ==>  24863  ==>  24864
==>  24865  ==>  24866  ==>  82208  ==>  82209  ==>  82210  ==>
21518  ==>  21431  ==>  21432  ==>  21433  ==>  21434  ==>  21435
==>  21436  ==>  21437  ==>  21438  ==>  21439  ==>  21440  ==>
```

```
21441  ==>  21442  ==>  21443  ==>  21444  ==>  21445  ==>  21446
==>  21447  ==>  21448  ==>  21449  ==>  21450  ==>  21451  ==>
621  ==>  21452  ==>  21453  ==>  21454  ==>  21495  ==>  21496  ==>
539  ==>    540  ==>    541  ==>    542  ==>    543  ==>    544  ==>
545  ==>    546  ==>    547  ==>    548  ==>    549]
```

.g

```python
@dataclass(frozen=True)
class MapState(GraphProblemState):
```

השורה הראשונה קובעת כי האובייקט לא יהיה ניתן לשינוי.

נרצה לקבוע כי משתנה זה יהיה לא ניתן לשינוי, מפני שאובייקט זה מגדיר לנו צומת בגרף המצבים, ואם נחזיר למשתמש צומת זו, והוא בטעות ישנה אותה, או שאנחנו בטעות נשנה אותה.
כל מרחב המצבים שלנו יכול להשתנות, ולפגוע בפתרון הבעיה.
נניח כי שינו את מספר הצומת המגדירה את הstate, כעת לא נוכל להמשיך להשתמש במצב זה כי הוא כבר לא מגדיר לנו את הצומת הנכונה ונקבל פתרון שאינו נכון.

תרגיל 13

```
Solve the map problem.
StreetsMap(src: 54 dst: 549)          UniformCost
time:   1.07   #dev: 17354   |space|: 17514    total_g_cost:
7465.52560   |path|: 137   path: [   54  ==>    55  ==>    56  ==>
57  ==>    58  ==>    59  ==>    60  ==>  28893  ==>  14580  ==>
14590  ==>  14591  ==>  14592  ==>  14593  ==>  81892  ==>  25814
==>    81  ==>  26236  ==>  26234  ==>   1188  ==>  33068  ==>
33069  ==>  33070  ==>  15474  ==>  33071  ==>   5020  ==>  21699
==>  33072  ==>  33073  ==>  33074  ==>  16203  ==>   9847  ==>
9848  ==>   9849  ==>   9850  ==>   9851  ==>    335  ==>   9852  ==>
82906  ==>  82907  ==>  82908  ==>  82909  ==>  95454  ==>  96539
==>  72369  ==>  94627  ==>  38553  ==>  72367  ==>  29007  ==>
94632  ==>  96540  ==>   9269  ==>  82890  ==>  29049  ==>  29026
==>  82682  ==>  71897  ==>  83380  ==>  96541  ==>  82904  ==>
96542  ==>  96543  ==>  96544  ==>  96545  ==>  96546  ==>  96547
==>  82911  ==>  82928  ==>  24841  ==>  24842  ==>  24843  ==>
5215  ==>  24844  ==>   9274  ==>  24845  ==>  24846  ==>  24847  ==>
24848  ==>  24849  ==>  24850  ==>  24851  ==>  24852  ==>  24853
==>  24854  ==>  24855  ==>  24856  ==>  24857  ==>  24858  ==>
24859  ==>  24860  ==>  24861  ==>  24862  ==>  24863  ==>  24864
==>  24865  ==>  24866  ==>  82208  ==>  82209  ==>  82210  ==>
21518  ==>  21431  ==>  21432  ==>  21433  ==>  21434  ==>  21435
==>  21436  ==>  21437  ==>  21438  ==>  21439  ==>  21440  ==>
21441  ==>  21442  ==>  21443  ==>  21444  ==>  21445  ==>  21446
==>  21447  ==>  21448  ==>  21449  ==>  21450  ==>  21451  ==>
621  ==>  21452  ==>  21453  ==>  21454  ==>  21495  ==>  21496  ==>
539  ==>    540  ==>    541  ==>    542  ==>    543  ==>    544  ==>
545  ==>    546  ==>    547  ==>    548  ==>    549]
StreetsMap(src: 54 dst: 549)          A* (h=0, w=0.500)
time:   1.01   #dev: 17354   |space|: 17514    total_g_cost:
7465.52560   |path|: 137   path: [   54  ==>    55  ==>    56  ==>
57  ==>    58  ==>    59  ==>    60  ==>  28893  ==>  14580  ==>
14590  ==>  14591  ==>  14592  ==>  14593  ==>  81892  ==>  25814
==>    81  ==>  26236  ==>  26234  ==>   1188  ==>  33068  ==>
```

```
33069  ==>  33070  ==>  15474  ==>  33071  ==>   5020  ==>  21699
==>  33072  ==>  33073  ==>  33074  ==>  16203  ==>   9847  ==>
9848  ==>   9849  ==>   9850  ==>   9851  ==>    335  ==>   9852  ==>
82906  ==>  82907  ==>  82908  ==>  82909  ==>  95454  ==>  96539
==>  72369  ==>  94627  ==>  38553  ==>  72367  ==>  29007  ==>
94632  ==>  96540  ==>   9269  ==>  82890  ==>  29049  ==>  29026
==>  82682  ==>  71897  ==>  83380  ==>  96541  ==>  82904  ==>
96542  ==>  96543  ==>  96544  ==>  96545  ==>  96546  ==>  96547
==>  82911  ==>  82928  ==>  24841  ==>  24842  ==>  24843  ==>
5215  ==>  24844  ==>   9274  ==>  24845  ==>  24846  ==>  24847  ==>
24848  ==>  24849  ==>  24850  ==>  24851  ==>  24852  ==>  24853
==>  24854  ==>  24855  ==>  24856  ==>  24857  ==>  24858  ==>
24859  ==>  24860  ==>  24861  ==>  24862  ==>  24863  ==>  24864
==>  24865  ==>  24866  ==>  82208  ==>  82209  ==>  82210  ==>
21518  ==>  21431  ==>  21432  ==>  21433  ==>  21434  ==>  21435
==>  21436  ==>  21437  ==>  21438  ==>  21439  ==>  21440  ==>
21441  ==>  21442  ==>  21443  ==>  21444  ==>  21445  ==>  21446
==>  21447  ==>  21448  ==>  21449  ==>  21450  ==>  21451  ==>
621  ==>  21452  ==>  21453  ==>  21454  ==>  21495  ==>  21496  ==>
539  ==>    540  ==>    541  ==>    542  ==>    543  ==>    544  ==>
545  ==>    546  ==>    547  ==>    548  ==>    549]
StreetsMap(src: 54 dst: 549)          A* (h=AirDist, w=0.500)
time:   0.17   #dev: 2015    |space|: 2229     total_g_cost:
7465.52560   |path|: 137   path: [   54  ==>     55  ==>     56  ==>
57  ==>     58  ==>     59  ==>     60  ==>  28893  ==>  14580  ==>
14590  ==>  14591  ==>  14592  ==>  14593  ==>  81892  ==>  25814
==>     81  ==>  26236  ==>  26234  ==>   1188  ==>  33068  ==>
33069  ==>  33070  ==>  15474  ==>  33071  ==>   5020  ==>  21699
==>  33072  ==>  33073  ==>  33074  ==>  16203  ==>   9847  ==>
9848  ==>   9849  ==>   9850  ==>   9851  ==>    335  ==>   9852  ==>
82906  ==>  82907  ==>  82908  ==>  82909  ==>  95454  ==>  96539
==>  72369  ==>  94627  ==>  38553  ==>  72367  ==>  29007  ==>
94632  ==>  96540  ==>   9269  ==>  82890  ==>  29049  ==>  29026
==>  82682  ==>  71897  ==>  83380  ==>  96541  ==>  82904  ==>
96542  ==>  96543  ==>  96544  ==>  96545  ==>  96546  ==>  96547
==>  82911  ==>  82928  ==>  24841  ==>  24842  ==>  24843  ==>
5215  ==>  24844  ==>   9274  ==>  24845  ==>  24846  ==>  24847  ==>
24848  ==>  24849  ==>  24850  ==>  24851  ==>  24852  ==>  24853
==>  24854  ==>  24855  ==>  24856  ==>  24857  ==>  24858  ==>
24859  ==>  24860  ==>  24861  ==>  24862  ==>  24863  ==>  24864
==>  24865  ==>  24866  ==>  82208  ==>  82209  ==>  82210  ==>
21518  ==>  21431  ==>  21432  ==>  21433  ==>  21434  ==>  21435
==>  21436  ==>  21437  ==>  21438  ==>  21439  ==>  21440  ==>
21441  ==>  21442  ==>  21443  ==>  21444  ==>  21445  ==>  21446
==>  21447  ==>  21448  ==>  21449  ==>  21450  ==>  21451  ==>
621  ==>  21452  ==>  21453  ==>  21454  ==>  21495  ==>  21496  ==>
539  ==>    540  ==>    541  ==>    542  ==>    543  ==>    544  ==>
545  ==>    546  ==>    547  ==>    548  ==>    549]
```

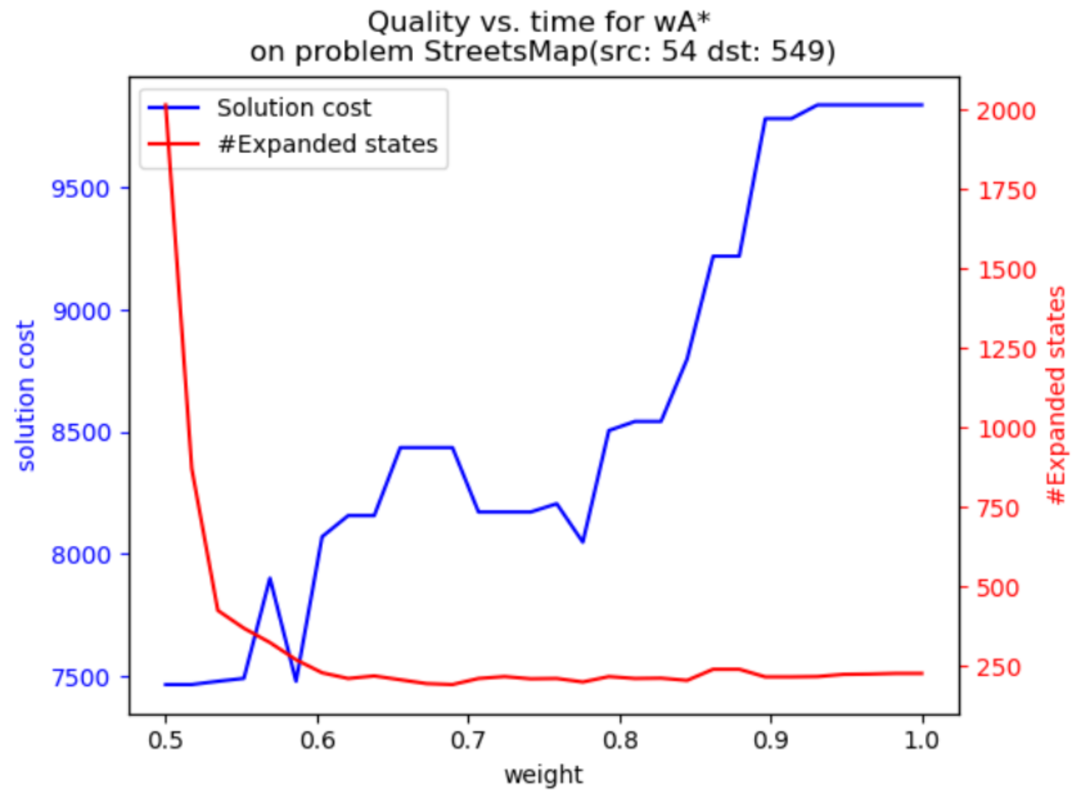
מס' פיתוחי מצבים יחסי שחסכנו :

$$\frac{17354-2015}{17354}=0.8838$$

כלומר חסכנו 88.38 אחוז מהפיתוחים.

תרגיל 14

הגרף שנוצר בסעיף זה :



הגרף הכחול מתאר לנו את טיב הפתרון (ככל שיותר נמוך יותר טוב) כפונקציה של המשקל w, והגרף האדום מתאר לנו את מספר המצבים שפותחו כפונקציה של המשקל w.

ניתן לראות כי עבור טיב הפתרון, ככל שנגדיל את w כך טיב הפתרון שלנו ירד, כלומר העלות של הפתרון גדלה (מסלול פתרון ארוך\ יקר יותר).

וככל שנגדיל את w מספר הצמתים שנפתח קטן יותר (עד ערך מסויים של w בו כמעט ואין שינוי במספר הפיתוחים).

נזכיר כי ככל שw גדול יותר כך נותנים משקל גדול יותר להיוריסטיקה שלנו, וכאשר w=0.5 נקבל את AStar שבמקרה זה מובטח לנו פתרון אופטימלי.

במקרה שלנו היינו בוחרים ערך w השווה בקירוב ל 0.585 , ניתן לראות בגרף כי עבור ערך זה טיב הפתרון כמעט וזהה לטיב הפתרון האופטימלי, ובנוסף לעומת הפתרון האופטימלי שמתקבל על ידי AStar , מספר המצבים שפותחו עבור w זה הינו קטן בהרבה מאשר בAStar .
כך בעצם אנו מורידים את מספר הפיתוחים אך שומרים על פתרון אופטימלי.

גם בגרף שלו ניתן לראות כי ככל ש w גדל איכות הפתרון יורדת ומספר הפיתוחים יורד גם הוא, אך ישנם מקרים בהם כאשר נגדיל את w נקבל איכות פתרון טובה יותר ומספר פיתוחים קטן יותר.
ניתן לראות מקרים אלה בגרף בחלקים בהם יש ירידה בגרף הכחול לדוג' בערך w=0.7.

תרגיל 19

נראה כי ההיוריסטיקה TruckDeliveriesMaxAirDistHeuristic הינה קבילה.

כלומר עלינו להראות כי לכל מצב, ערך היוריסטיקה של מצב זה יהיה קטן מהמרחק האמיתי של מצב זה מהפתרון ואי שלילי. $\forall s, 0 \leq h(s) \leq h^*(s)$

כאשר h^* הינה היוריסטיקה מושלמת, כלומר מחזירה את המרחק האמיתי למצב סופי.

יהי $A = \{a_1, a_2, \dots, a_n\}$ קבוצת הנקודות שנותרו למשאית לעבור דרכם, כולל המיקום הנוכחי של המשאית, נניח בלי הגבלת הכלליות כי מיקום זה הוא a_1.
יהי $h(a_1) = d$ ערך היוריסטיקה עבור צומת ההתחלה, כלומר d הינו המרחק האווירי המקסימלי בין זוג נק' בA .
נניח בה"כ כי d הינו המרחק האווירי בין a_2, a_3.

יהי R מסלול אופטימלי לבעיה המתחיל ב a_1 ועובר בכל הנק' בA והוא בעל סכום המרחקים הקטן ביותר = D .
כלומר עלינו להוכיח כי מתקיים : $0 \leq d \leq D$ על מנת להוכיח קבילות.

R מהצורה הבאה: $R = a_1 \rightarrow u_1 \rightarrow u_2 \rightarrow \cdots \rightarrow u_{n-1}$ ונניח כי קיים בR תת מסלול מ a_2 ל a_3 (R עובר בכל הנק' לכן בהכרח עובר בשתי נק' אלה, נניח כי הסדר הוא כזה).
תת מסלול זה הוא מהצורה : $a_2 \rightarrow u_i \rightarrow u_{i+1} \rightarrow \cdots \rightarrow a_3$.
נזכר כי מרחק אווירי בין שתי נק' הינו המרחק המינימלי ביניהן, כלומר אורך המסלול מ a_2 ל a_3 בהכרח $\leq d$.

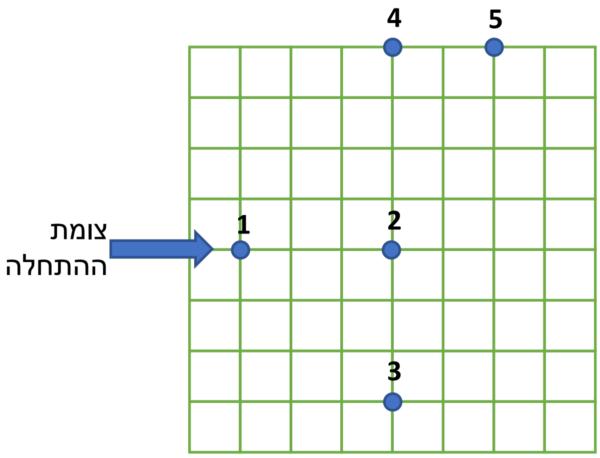
בגלל שאין מרחקים שליליים, מתקיים כי אורך המסלול $R \geq d$ ולכן מתקיים : $0 \leq d \leq D$ כאשר d בהכרח אי שלילי.

כלומר ערך ההיוריסטיקה שלנו תמיד יחזיר ערך אי שלילי אשר קטן שווה לערך הפתרון האופטימלי ולכן היוריסטיקה קבילה.

תרגיל 21

נראה כי ההיוריסטיקה *TruckDeliveriesSumAirDistHeuristic* אינה קבילה על ידי דוגמא נגדית.

כלומר נראה קבוצת נק' ונק' התחלה, כך שערך היוריסטיקה עבור נק' ההתחלה הינו גדול מאורך הפתרון האמיתי מנק' זו.



נשים לב כי המסלול שהיוריסטיקה תיצור הינו המסלול כך שתמיד נתקדם לצומת שקרובה אלינו ביותר. כלומר אם מתחילים מצומת 1 היוריסטיקה תיצור את המסלול הבא :

$$1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5$$

והערך שהיוריסטיקה תחזיר עבור צומת 1 הינו סכום האורכים במסלול זה , כלומר :

$$h(1) = 3 + 3 + 7 + 2 = 15$$

אך נסתכל על המסלול הבא : (לא בהכרח אופטימלי)

$$1 \rightarrow 3 \rightarrow 2 \rightarrow 4 \rightarrow 5$$

אורך מסלול זה הינו : $d = \sqrt{18} + 3 + 4 + 2 \approx 13.24$

כלומר, מצאנו מסלול העובר בכל הנק', ואורכו קטן מהערך שההיוריסטיקה החזירה לנו.

כלומר מתקיים :

$$h(1) > d \geq h^*(1)$$

כלומר לא ייתכן כי ההיוריסטיקה קבילה.

תרגיל 25

נראה כי ההיוריסטיקה TruckDeliveriesMSTAirDistHeuristic הינה קבילה.

כלומר עלינו להראות כי לכל מצב, ערך היוריסטיקה של מצב זה יהיה קטן מהמרחק האמיתי של מצב זה מהפתרון ואי שלילי. $\forall s, 0 \leq h(s) \leq h^*(s)$

יהי $A = \{a_1, a_2, \dots, a_n\}$ קבוצת הנקודות שנותרו למשאית לעבור דרכם, כולל המיקום הנוכחי של המשאית, נניח בלי הגבלת הכלליות כי מיקום זה הוא a_1.
ערך ההיוריסטיקה עבור הנקודה a_1 הינו סכום המשקלים של הקשתות בעץ הפורש את הגרף המלא בו הצמתים הן *A* וכל קשת במשקל המרחק האווירי בין הצמתים, נסמן גרף מלא זה ב*G*.

מחיר של פתרון הינו סכום האורכים של הקשתות במסלול העובר בכל הנק' ב*A* ומתחיל בצומת ההתחלה a_1.
נשים לב כי מתקיים:
לכל מסלול המתחיל ב a_1 ועובר בכל שאר הנק' ב*A* ניתן להתאים בצורה חד-חד ערכית עץ פורש של *G* (נתעלם מכיוון הקשתות) , מפני שהמסלול קשיר ,ללא מעגלים ומכיל את כל צמתי *G* .
לכן מחיר פתרון כלשהו הינו בהכרח $\geq$ משקל של העץ הפורש המתאים לו (מפני שמחיר הקשתות במסלול לא יכול להיות קטן מהמשקל של הקשת המתאימה בעץ, מפני שמשקל הקשתות הוא המרחק האווירי בין הצמתים ולכן מרחק מינימלי).

יהי *R* פתרון אופטימלי לבעיה, כלומר *R* הינו מסלול העובר בכל צמתי *A* ומתחיל בa_1 והינו בעל סכום אורכים מינימלי.
יהי *T* העץ הפורש של *G* המתאים ל*R* , מתקיים : $weight(T) \leq cost(R)$
עפ"י הגדרה עפ"מ הוא העץ הפורש בעל המשקל המינימלי עבור גרף *G*.
כלומר מתקיים :

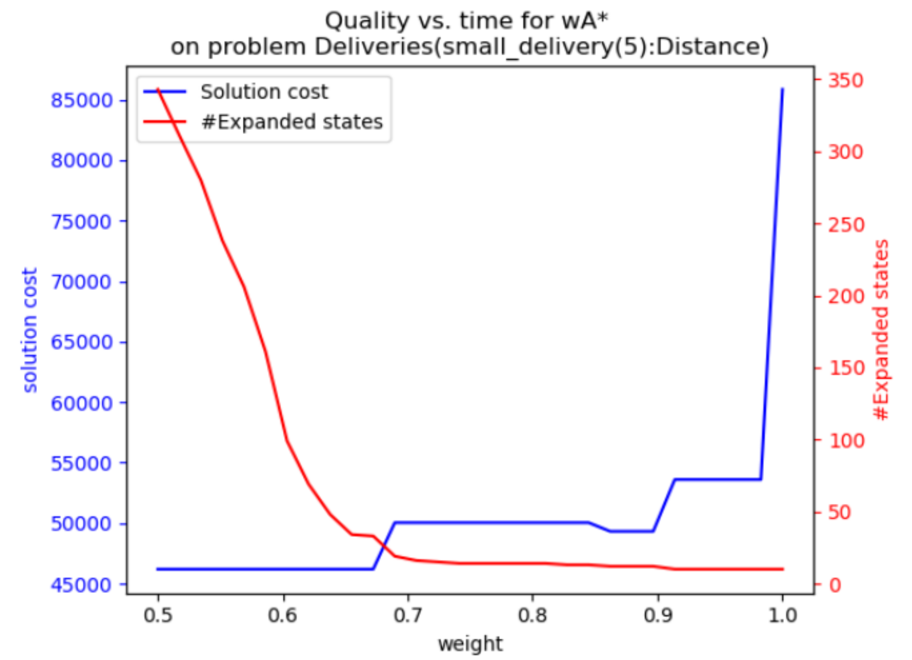
$$0 \leq h(a_1) = weight(MST(G)) \leq weight(T) \leq cost(R) = h^*(a_1)$$

ולכן ההיוריסטיקה קבילה.

תרגיל 26

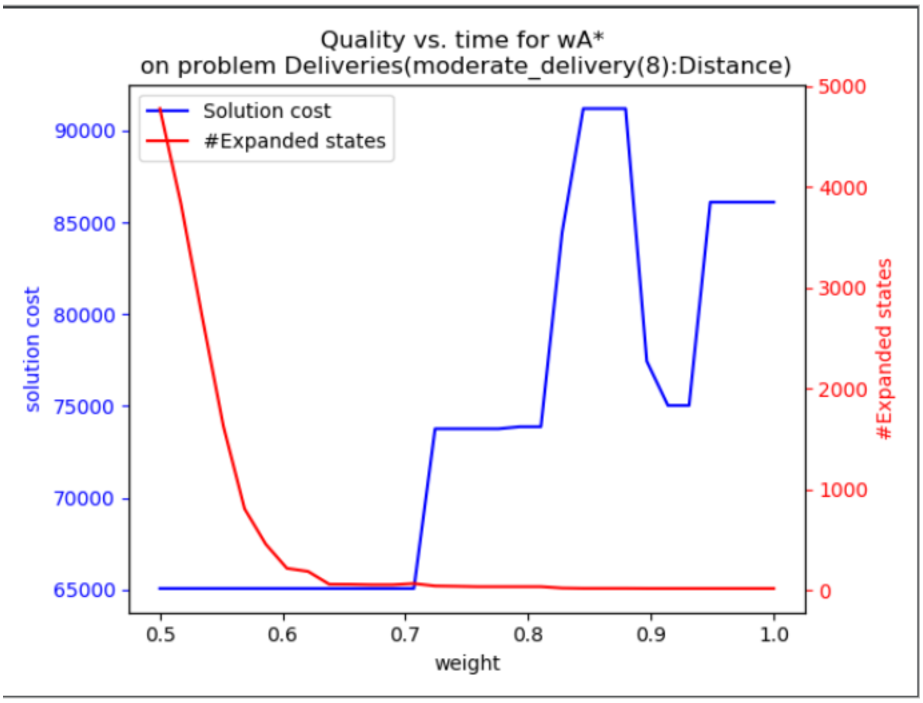
הגרפים שקיבלנו הם :

עבור הרצה עם היוריסטיקה *MST* :



ניתן לראות בגרף כי ככל שמעלים את *w* , מספר הפיתוחים שנבצע יורד, ועלות הפתרון עולה כמו שציפינו.
במקרה זה היינו בוחרים ב*w* השווה בערך ל0.685 , ניתן לראות כי עבור *w* זה איכות הפתרון נשמרת, ומספר הצמתים שפותחו הינו קטן יחסית לכל פתרון אופטימלי אחר שמצאנו.

עבור הרצה עם היוריסטיקה *SUM*:



ניתן לראות בגרף זה בדומה לגרפים קודמים כי ככל שנגדיל את *w* כך מספר הצמתים שנפתח קטן, בנוסף איכות הפתרון יורדת ככל שנגדיל את *w* . בדומה לגרף הראשון שהצגנו בתרגיל זה, גם כאן ישנם מקרים בהם נגדיל את *w* אך נקבל איכות פתרון טובה יותר, לדוגמא עבור *w=0.9* נקבל איכות פתרון טובה יותר מאשר עבור *w=0.85*.
במקרה זה נבחר *w=0.7* מסיבות דומות להסבר הקודם, עבור *w* זה מספר הצמתים שפיתחנו מינימלי ואיכות הפתרון עדיין אופטימלית.

תרגיל 29

תוצאות הרצה זו :

```
Solve the truck deliveries problem (small input, time & money
objectives).
```

הרצה עם מזעור הזמן:

```
Deliveries(small_delivery(5):Time)     A*
(h=TruckDeliveriesMSTAirDist, w=0.500)    time:  40.54   #dev: 425
|space|: 562      total_g_cost:    30.18082   total_cost:
DeliveryCost(dist=  46771.762 meter, time=     30.181 minutes, money=
151.961 nis)    |path|: 11
```

הרצה עם מזעור הכסף:

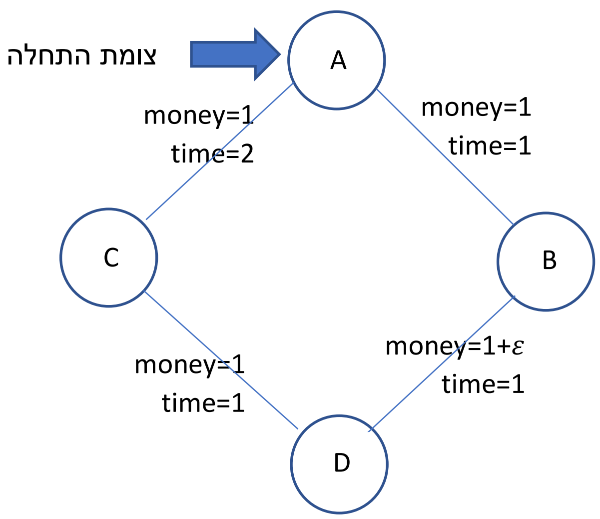
```
Deliveries(small_delivery(5):Money)    A*
(h=TruckDeliveriesMSTAirDist, w=0.500)    time:  61.70   #dev: 518
|space|: 591      total_g_cost:   104.23259   total_cost:
DeliveryCost(dist=  46763.990 meter, time=     35.248 minutes, money=
104.233 nis)    |path|: 11
```

ניתן לראות כי אכן הפתרון הראשון מקבל תוצאה עם עלות זמן נמוכה יותר, וכי הפתרון השני מקבל תוצאה עם עלות כסף נמוכה יותר.

תרגיל 30

נראה דוגמא בה התוצאה המתקבלת אינה אופטימלית.

נסמן ערך של קשת עבור *cost_money* ב*money* וערך עבור *cost_time* ב *time* ונשתמש בהיוריסטיקה מושלמת עבור כל שלב.



נסתכל על הדוגמא שהראנו ונראה כי אינה מחזירה פתרון אופטימלי לפי המדד החדש:

בשלב הראשון *A* נמצא ב *OPEN* והינו היחיד ולכן יבחר , לאחר מכן *OPEN* יכיל את *B* ו *C*

כעת שניהם בעלי אותו מחיר כספי ולכן יופיעו ב*FOCAL* ונבחר את *B* לפיתוח הבא בגלל שמחירו בזמן נמוך יותר.

כעת *OPEN* מכיל את *C* ו *D* , נבחר את *C* לפיתוח כי המחיר הכספי שלו נמוך יותר (1 לעומת 2+אפסילון) .

כעת בפיתוח *C* נפתח את *D* שוב, ונבדוק האם שיפרנו את המדד של המחיר הכספי ונראה כי אכן שיפרנו באפסילון, כלומר נשנה לצומת *D* את האב להיות *C* במקום *B* ובכך נסיים בעצם כי *D* הינה צומת סיום והינה היחידה ב*OPEN*.

כעת נסתכל על הפתרון שקיבלנו : $A \to C \to D$

מחירו בכסף הינו 2, ומחירו בזמן הינו 3.

נסתכל על פתרון אחר : $A \to B \to D$

מחירו בכסף הינו $2 + \varepsilon$, ומחירו בזמן הינו 2.

כלומר מצאנו כי קיים פתרון שונה מהפתרון שהחזיר האלגוריתם, והוא פתרון אופטימלי יותר במדדים שקבענו.

תרגיל 31

נציע עדכון לאלגוריתם *Astar* על מנת לקבל פתרון אופטימלי על פי המדד שהוגדר, כלומר לקבל פתרון בעל *cost_time* מינימלי מבין הפתרונות עם *cost_money* המקיים $cost_{money} \leq (1+\varepsilon)C^*$ כאשר C^* הינו מחיר הפתרון האופטימלי.

נציע את השינויים הבאים :

תחילה נשנה את תנאי העצירה של האלגוריתם, ברגע הוצאת מצב מסיים מ*OPEN* נשמור את ערך f שלו במשתנה *best_money*

כעת במקום לעצור ולהחזיר פתרון נמשיך את ריצת האלגוריתם עד אשר נמצא פתרון המקיים $f > (1+\varepsilon) * best\_money$.

שינוי שני יהיה עבור כל צומת נשמור רשימה של אבות במקום רק אב יחיד, כאשר נשמור בצורה הבאה:

לכל צומת נשמור את המשתנים הבאים :

Father list, money list, time list, best money father, best money, best time

כאשר נגיע לצומת חדשה שהגענו אליה לפני או שלא, נוסיף לכל אחת מהרשימות את הנתונים , את האב ממנו הגענו לרשימת האבות, את הכסף שעלה לנו להגיע מאבא זה לרשימת הכסף ואת הזמן שלקח מהאבא הזה לרשימת הזמן. בנוסף נעדכן את ערכי ה*best* אם הם משפרים, כלומר *best money father* אם האב שהגענו ממנו משפר את ערך הכסף, וכדומה.

כאשר נמצא פתרונות לבעיה אשר לא גורמים לעצירת האלגו' לפי מה שתואר מעל, נוסיף אותם לרשימת פתרונות שמצאנו.

בסיום האלגוריתם, כלומר בזמן שנמצא פתרון במחיר גדול מהתנאי, נעבור על הפתרונות שמצאנו עד כה.

במעבר על הפתרונות עלינו למצוא מתוכן את הפתרון בעל הזמן המינימלי, כלומר נתחזק משתנה שיחזיק בכל רגע את הפתרון הכי טוב מבחינת זמן שראינו עד כה.

נעבור על כל הפתרונות שמצאנו, נוודא שני דברים בכל פעם, אחד כי הפתרון עומד בתנאי של $(1+\varepsilon) * best\_money$ (ייתכן כי יהיו מסלולים בדרך שלא יעמדו בתנאי זה למרות שחלק מהמסלול שלהם שייך למסלול שכן עומד בתנאי) ודבר שני שנבדוק זה את הזמן שלהם.

שינוי זה לאלגוריתם, פוגע בביצועים גם מבחינת מקום וגם מבחינת זמן.
תחילה עלינו לשמור יותר מידע, בכל צומת אנו שומרים מידע על כל האבות של אותה צומת.
בנוסף הארכנו את תנאי העצירה של האלגוריתם ולכן בניגוד ל *ASTAR* סטנדרטי שעוצר ברגע שמצא פתרון, אנחנו נמשיך ונחפש עוד פתרונות.
ודבר אחרון זה הוספת הזמן בבחירת הפתרון המתאים לאחר סיום האלגוריתם. יהיה עלינו לעבור על כל הפתרונות שמצאנו ועל כל המסלולים האפשריים בכל אחד מהם ולמצוא את הפתרון האופטימלי מביניהם ולכן נבזבז בו עוד זמן.

תרגיל 33

```
Solve the truck deliveries problem (moderate input, distance
objective, using A*eps, use non-acceptable heuristic as focal
heuristic).
Deliveries(moderate_delivery(8):Distance)   A*
(h=TruckDeliveriesMSTAirDist, w=0.500)   time:  10.58   #dev: 6376
|space|: 9045     total_g_cost: 65062.81195   total_cost:
DeliveryCost(dist=  65062.812 meter, time=     42.946 minutes, money=
210.589 nis)   |path|: 17

Deliveries(moderate_delivery(8):Distance)   A*eps
(h=TruckDeliveriesMSTAirDist, w=0.500)   time:  58.33   #dev: 6339
|space|: 9025     total_g_cost: 65062.81195   total_cost:
DeliveryCost(dist=  65062.812 meter, time=     42.946 minutes, money=
210.589 nis)   |path|: 17
```

ניתן לראות כי אכן חסכנו בפיתוחים, יש הבדל של 37 צמתים בפיתוח לטובת הפתרון שמשתמש בשתי ההיוריסטיקות.

הציפייה שלנו לחסכון בפיתוח צמתים נבעה מכך שאנו מוסיפים מידע נוסף למערכת שלנו.
בנוסף, אפשרנו לפתח צמתים שאינם הצמתים המינימלים, כלומר אפשרו ל*COST* לצאת מנקודות עוקף או בורות בהן לא יכלנו להתקדם, בנוסף ההנחה שלנו היא כי ייתכן שמזעור הזמן יוביל בנוסף למזעור של הכסף, כלומר ננסה לצאת מהעוקף או מהבור על ידי התקדמות לכיוון הממזער את הזמן.

תרגיל 35

התוצאות שקיבלנו:

Solve the truck deliveries problem (moderate input, only distance objective, Anytime-A, MSTAirDist heuristics).*

Deliveries(moderate_delivery(8):Distance)   Anytime-A (h=TruckDeliveriesMSTAirDist, w=0.688)   time: 1.08   #dev: 335 |space|: 200   total_g_cost: 65459.89155   total_cost:*

DeliveryCost(dist= 65459.892 meter, time=   43.581 minutes, money=   212.405 nis)
|path|: 17

Solve the truck deliveries problem (big input, only distance objective, Anytime-A, SumAirDist & MSTAirDist heuristics).*

Deliveries(big_delivery(15):Distance)   Anytime-A (h=TruckDeliveriesSumAirDist, w=0.844)*
time: 196.05   #dev: 2031 |space|: 1861   total_g_cost: 155572.84122   total_cost:
DeliveryCost(dist= 155572.841 meter, time=   103.868 minutes, money=   508.621 nis)
|path|: 31

Deliveries(big_delivery(15):Distance)   Anytime-A (h=TruckDeliveriesMSTAirDist, w=0.875)*
time: 48.80   #dev: 2400 |space|: 1848   total_g_cost: 147869.82115   total_cost:
DeliveryCost(dist= 147869.821 meter, time=   98.608 minutes, money=   483.292 nis)
|path|: 31

נשים לב כי עבור הבעיה הבינונית לקח לנו שנייה אחת לפתור יחסית ל 70 שניות שחישבנו בחלק א'

ועבור הבעיה הגדולה לקח לנו 196 שניות לפתרון לעומת מיליוני שנים שחישבנו בחלק א'

כלומר הרצה בצורה כזו חסכה לנו כמות זמן שלא הייתה אפשרית לריצה.