מבוא לבינה מלאכותית 236501

**תרגיל בית מספר 1**

מגישים: פבל רסטופצ'ין 311137095 אורי קירשטיין 311137095

**פרק ראשון**

חלק א

1. להלן הטבלה המבוקשת:

|  |  |  |
| --- | --- | --- |
| **K** | **K!** | **K!\*L^(K-1)** |
| 1 | 1 | 1 |
| 2 | 2 | 10 |
| 3 | 6 | 150 |
| 4 | 24 | 3000 |
| 5 | 120 | 75000 |
| 6 | 720 | 2250000 |
| 7 | 5040 | 78750000 |
| 8 | 40320 | 3150000000 |
| 9 | 362880 | 1.4175E+11 |
| 10 | 3628800 | 7.0875E+12 |

חלק ב

1. מקדם הסיעוף המינימלי האפשרי במרחב החיפוש הוא 0 – יתקבל למשל כאשר נתחיל בלי דלק כלל. לא נוכל להגיע לאף מצב אחר במרחב והגרף יישאר בעל צומת בודד. מקדם הסיעוף המקסימלי האפשרי במרחב החיפוש הוא , שיתקבל, למשל, אם נוכל להגיע לכל המצבים מנקודת ההתחלה. נוכיח כי זהו המקדם המקסימלי בשלילה. נניח כי קיים מקדם סיעוף . כלומר, קיים מצב במרחב ממנו ניתן להגיע ל w מצבים אחרים. אבל יש רק צמתים אחרים לכל היותר בגרף אליהם ניתן להגיע בעזרת האופרטור. סתירה.
2. ייתכנו מעגלים במרחב המצבים שלנו. למשל, עבור המרחב המתואר למטה, נסתכל על סדרת המצבים.



מצב התחלתי -

אם נמשיך לתחנת הדלק הקרובה ביותר נגיע למצב

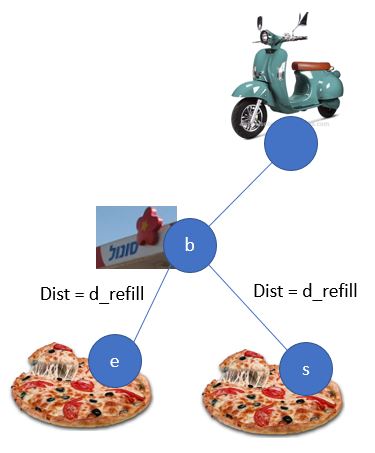
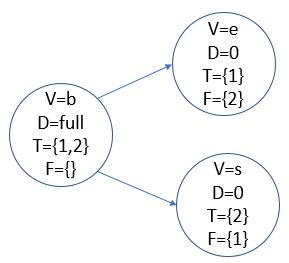
ממנו נמשיך לתחנת הדלק השנייה .

עכשיו יש רק אופרטור אפשרי אחד – ניתן לחזור לתחנת הדלק הקודמת בלבד, ונגיע שוב למצב הקודם,

הגענו לאותו הצומת פעמיים, לכן מצאנו מעגל כנדרש.



1. ישנם אינסוף מצבים במרחב. אמנם מספר האפשרויות השונות לT חסום ע"י , F נקבע ע"י בחירת T באופן יחיד, וכן מספר הצמתים בגרף הוא , מספר האפשרויות לבחירת d אינסופי כי d פרמטר רציף. לא כל המצבים ישיגים, למשל במפה מהסעיף הקודם שום מצב בצומת התחתון אינו ישיג.
2. ייתכנו בורות שאינם מצבי מטרה במרחב המצבים. למשל במפה שלמטה – המצבים בצמתים e וs הם בורות, כיוון שבכל דרך בה נגיע אליהם מיכל הדלק יהיה ריק ולא נוכל להמשיך לאף צומת אחר, לכן לא יהיו יותר מעברים חוקיים.

1. חסם תחתון מינימלי הוא מספר ההזמנות. במקרה הכי טוב, בכל צעד נגיע לצומת הזמנה אחד, ולא ניאלץ לתדלק כלל. לכן החסם הוא

חלק ד

load\_map\_from\_csv: 3.34sec

Solve the map problem.

Map(src: 54 dst: 549) UniformCost time: 1.51 #dev: 17355 total\_cost: 7465.52897 |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]

חלק ה

11.

Solve the map problem.

Map(src: 54 dst: 549) UniformCost time: 1.14 #dev: 17355 total\_cost: 7465.52897 |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]

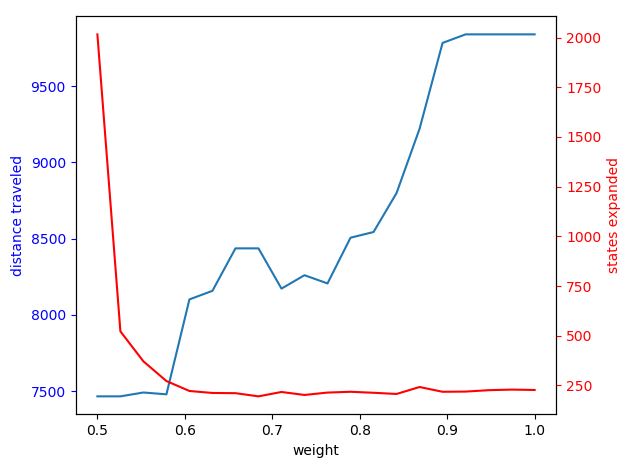
Map(src: 54 dst: 549) A\* (h=0, w=0.500) time: 1.16 #dev: 17355 total\_cost: 7465.52897 |path|: 137

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]

Map(src: 54 dst: 549) A\* (h=AirDist, w=0.500) time: 0.34 #dev: 2016 total\_cost: 7465.52897 |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]

12.



כלל שהמשקל מתקרב ל-1, כך האלגוריתם חמדן יותר, כלומר הוא מסתמך יותר על היוריסטיקה. ככל שההסתמכות על היוריסטיקה גדולה יותר, הפתרון יתקבל מהר יותר אך יהיה פחות אופטימלי.