**Question 1:**

a)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Start10 | Start12 | Start20 | Start30 | Start40 |
| ucsdijkstra | 2565 | Mem | Mem | Mem | Mem |
| ideepsearch | 2407 | 13812 | 5297410 | Time | Time |
| astar | 33 | 26 | 915 | Mem | Mem |
| idastar | 29 | 21 | 952 | 17297 | 112571 |

b)

1. According to this table above, we can know that **IDA\* Search** is the most efficient algorithm in this four, since it does not appear any out of global stack error and runtime error from start10 to start40.

2. Compared with other three searching algorithm, **A\* Search** is the second efficient because of the fewer node, although it has memory error in start30 and start40.

3. **Iterative Deepening Search** is not efficient since it generates a lot of nodes and runtime error in Start30 and Start40, but it is better than **Uniformed Cost Search**.

4. From this table, **Uniformed Cost Search** is not efficient algorithm, because it not only generates overmuch nodes in start10, and it causes memory error in the rest of the test.

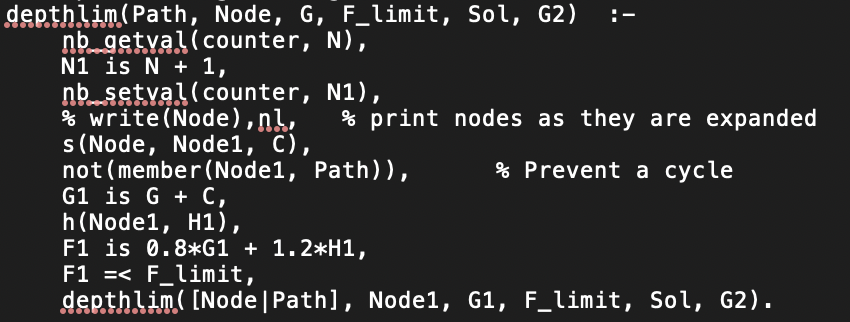
**Question 2:**

a)

The answer is from the table below.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Start50 | | Start60 | | Start64 | |
| IDA\* | 50 | 14642512 | 60 | 321252368 | 64 | 1209086782 |
| 1.2 | 52 | 191438 | 62 | 230861 | 66 | 431033 |
| 1.4 | 66 | 116342 | 82 | 4432 | 94 | 190278 |
| 1.6 | 100 | 33504 | 148 | 55626 | 162 | 235848 |
| Greedy | 164 | 5447 | 166 | 1617 | 184 | 2174 |

b)



c)

The answer is from the table above.

: 

: 

d)

**Greedy Search** and **IDA\* Search** are both stable algorithms, since it does not cause any memory or runtime error from start50 to start 64.

According to the table above and formula , the number of nodes is the larger than other four situation when , but the length of path is the least.

From ,and , we can get that if closes to 2, the number of nodes is reducing, and the length of path is increasing.

Though **Greedy Search** is not complete and optimal, it uses less time than **IDA\* Search**. As a result, if we need to get the least path, should be 1. However, although we give up the optimal solution, the speed of run can be faster when is increasing.

**Question 3:**

a)

b)

(i)

No. If we need to move the point from (1,1) to (5,6), we can get . Actually, the true cost () from (1,1) to (5,6) is 5. Thus, because of , the Straight-Line-Distinct heuristic is not admissible.

(ii)

No. Using the same example as the part (b) first question, we also get . So, the heuristic from part (a) is also not admissible.

(iii)

**Question 4:**

Light

F

Other cars

something in the front

red

green

right

F

no / left

no

T

yes

F

1. Considering that traffic light is red or green, agent cannot pass the road if the light is red. If traffic light is green is green, agent needs think about another situation.
2. If there are something such as pedestrians and cyclists in the front of agent, agent cannot move forward.
3. If there are other cars in the right road, agent has to give way to those cars.
4. Agent can move forward, if there are no other cars or other cars in left road.

In my opinion, this case is unable to use in an agent since there are a lot of unexpected situation and random forecast. For example, if there is an ambulance or fire truck, the agent has to give a way for them.

**Question 5:**

a)

These tables below are to determine whether customers have ability to repay the loans.

i) two attributes:

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Salary | House | Class |
| 1 | 60,000 | Yes | Yes |
| 2 | 40,000 | No | No |
| 3 | 100,000 | No | Yes |
| 4 | 40,000 | Yes | Yes |

Since , either salary or house can be as the first node.

Salary

>= 50,000

< 50,000

No

House

Yes

No

Yes

Yes

ii) three attributes:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Marital Status | Salary | House | Class |
| 1 | Single | 60,000 | Yes | Yes |
| 2 | Single | 40,000 | No | No |
| 3 | Married | 100,000 | No | Yes |
| 4 | Married | 40,000 | Yes | Yes |
| 5 | Single | 100,000 | No | Yes |
| 6 | Married | 30,000 | No | Yes |

Since , we can get decision tree below:

Salary

>= 50,000

< 50,000

No

Married

Yes

No

Yes

Yes

House

Yes

No

Yes

iii) four attributes:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Marital Status | Salary | House | Credit | Class |
| 1 | Single | 60,000 | Yes | Good | Yes |
| 2 | Single | 40,000 | No | Bad | No |
| 3 | Married | 100,000 | No | Bad | No |
| 4 | Married | 40,000 | Yes | General | Yes |
| 5 | Single | 100,000 | No | General | Yes |
| 6 | Married | 30,000 | No | Good | Yes |
| 7 | Single | 55,000 | Yes | General | Yes |
| 8 | Single | 45,000 | No | General | No |
| 9 | Single | 40,000 | Yes | Good | Yes |
| 10 | Married | 38,000 | Yes | General | Yes |
| 11 | Married | 41,000 | No | Good | Yes |
| 12 | Single | 33,000 | Yes | Bad | No |
| 13 | Single | 52,000 | No | Good | Yes |
| 14 | Married | 31,000 | Yes | General | Yes |
| 15 | Single | 70,000 | No | Bad | No |
| 16 | Married | 81,000 | Yes | Bad | No |

Thus, ‘Credit’ gives us the maximum information gain. We can get decision tree below:

Credit

good

bad

No

Yes

Salary

>= 50,000

< 50,000

No

Married

Yes

No

Yes

Yes

House

Yes

No

Yes

b)

In my opinion, this learning algorithm could not return absolutely correct tree. However, the algorithm can generate tree which is logically equivalent since there are many different trees can be generated by the same training-set example and different methods.