# Assingnment 1 Part 2

**Arth Sanskar Patel**
**z5228942**

| Table for Number of States expanded during the search (N) | | | | | |
|---|---|---|---|---|---|
| | start10 | start12 | start20 | start30 | start40 |
| UCS | 2565 | MEM | MEM | MEM | MEM |
| IDS | 2407 | 13812 | 5297410 | TIME | TIME |
| A* | 33 | 26 | 915 | MEM | MEM |
| IDA* | 29 | 21 | 952 | 17297 | 112571 |

## Part 2 Question 1

- For search algorithm ucs-Dijkstra, the need of memory and time increases geometrically when the tree grows deeper and hence it stops at start12. The no use of heuristics and storing every solution are the main reason for this bad efficiency.
    - Result:

      Time => 1/5

      Memory => 1/5
- For search algorithm IDS, the need for time grows with expanding tree as it needs time to go through every result. It doesn't store all the results, so it saves the memory.
    - Result:

      Time => 1/5

      Memory => 4/5
- For search algorithm A*, this is really advance compared to Dijkstra and IDS because this explore all the nodes required and still does achieve the same result. Since it stores previous results in memory, it eventually runs out of memory when the tree grows larger.
    - Result:

      Time => 5/5

      Memory => 3/5

- For search algorithm IDA*, this is giving a trade-off of time for better space optimization. It doesn't store all the paths found while searching, it searches for those paths again. It might sound time consuming, but it doesn't affect the speed by much.
    - Result:

      Time => 4/5

      Memory => 5/5

## Question 2 Part a,b,c

| | Start50 | | Start60 | | Start64 | |
|---|---|---|---|---|---|---|
| IDA* | 50 | 14642512 | 60 | 321252368 | 64 | 1209086782 |
| 1.2 | 52 | 191438 | 62 | 230861 | 66 | 431033 |
| 1.4 | 66 | 116174 | 82 | 3673 | 94 | 188917 |
| 1.6 | 100 | 34637 | 148 | 55626 | 162 | 235852 |
| Greedy | 164 | 5447 | 166 | 1617 | 184 | 2174 |

Part of the Changed code in hueristic.pl

⇨ As you can see, in line 44, the code was previously F1 is G1 + H1. Now it has changed to F1 is ((2-w)*G1) + w*H1 where in this case w = 1.2

```
34  % Keep searching until goal is found, or F_limit is exceeded.
35  depthlim(Path, Node, G, F_limit, Sol, G2)  :-
36      nb_getval(counter, N),
37      N1 is N + 1,
38      nb_setval(counter, N1),
39      % write(Node),nl,   % print nodes as they are expanded
40      s(Node, Node1, C),
41      not(member(Node1, Path)),      % Prevent a cycle
42      G1 is G + C,
43      h(Node1, H1),
44      F1 is ((2-1.2)*G1) + ((1.2)*H1),
45      F1 =< F_limit,
46      depthlim([Node|Path], Node1, G1, F_limit, Sol, G2).
47
```

d).

- The trade-off between speed and quality is highly visible as you can see from the table. For every different starting tree (start50, start60 and start64), it is clearly visible that the length of path (G) is increasing with the increase in w. for IDA* the value of w is 1 and int gradually increases to 2 for greedy.
- IDA* doesn't overestimate the heuristics and gives the most optimal result but takes a lot of time as it has to go through a high number of states to give this best result.
- On the other hand, greedy is very fast and gives the solutions quickly, but the solution is not at all optimal and the output is very bad.
- The algorithms in between are somewhere in between of greedy and IDA* in terms of time and quality. They have better time than IDA* and better quality than greedy. As w gradually increases from 1 (IDA*) to 2 (greedy), you can see the behaviour of the algorithm in between changes to greedy and starts giving out bad solutions quickly.
- To conclude, as
  - w ↑ time ↓
  - w ↑ quality ↓

    w is inversely proportional to time and quality.