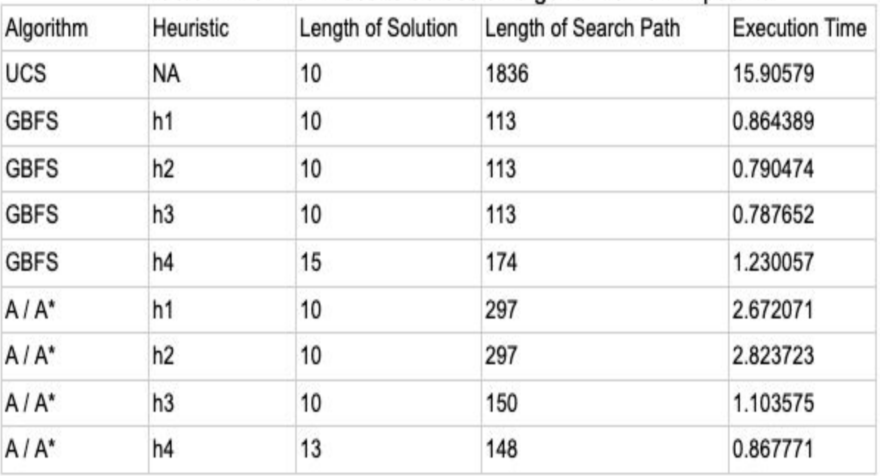
# Mini-Project 2: Rush Hour

Presented by Artificial Unintelligence

# UCS

- Always gives lowest cost solution
- Longest Search Path Length of all
- Longest Execution Time of all

Table 1: Puzzle 2 results across all algorithms from input file



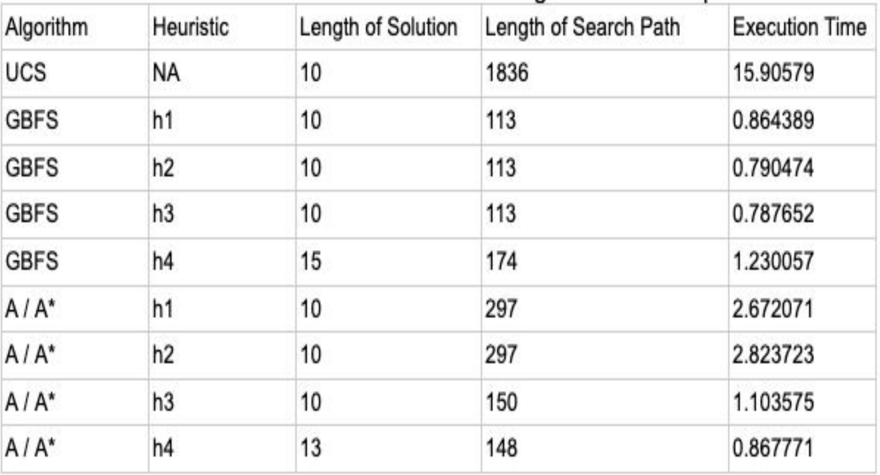
| Algorithm | Heuristic | Length of Solution | Length of Search Path | Execution Time |
|---|---|---|---|---|
| UCS | NA | 10 | 1836 | 15.90579 |
| GBFS | h1 | 10 | 113 | 0.864389 |
| GBFS | h2 | 10 | 113 | 0.790474 |
| GBFS | h3 | 10 | 113 | 0.787652 |
| GBFS | h4 | 15 | 174 | 1.230057 |
| A / A* | h1 | 10 | 297 | 2.672071 |
| A / A* | h2 | 10 | 297 | 2.823723 |
| A / A* | h3 | 10 | 150 | 1.103575 |
| A / A* | h4 | 13 | 148 | 0.867771 |

- If need of lowest cost solution -> good
- If need of a quick solution -> bad

# GBFS

- Rarely gives lowest cost solution
- Much shorter search path length
- Much shorter execution time

Table 1: Puzzle 2 results across all algorithms from input file



| Algorithm | Heuristic | Length of Solution | Length of Search Path | Execution Time |
|---|---|---|---|---|
| UCS | NA | 10 | 1836 | 15.90579 |
| GBFS | h1 | 10 | 113 | 0.864389 |
| GBFS | h2 | 10 | 113 | 0.790474 |
| GBFS | h3 | 10 | 113 | 0.787652 |
| GBFS | h4 | 15 | 174 | 1.230057 |
| A / A* | h1 | 10 | 297 | 2.672071 |
| A / A* | h2 | 10 | 297 | 2.823723 |
| A / A* | h3 | 10 | 150 | 1.103575 |
| A / A* | h4 | 13 | 148 | 0.867771 |

- Good for easy puzzles but less good for longer puzzles as will give longer solution length than optimal

# A / A*

- Here we only did A as it asked us A or A*
- Usually results in lowest cost solution (not always)
- Search path length much shorter than UCS
- Execution time much quicker than UCS

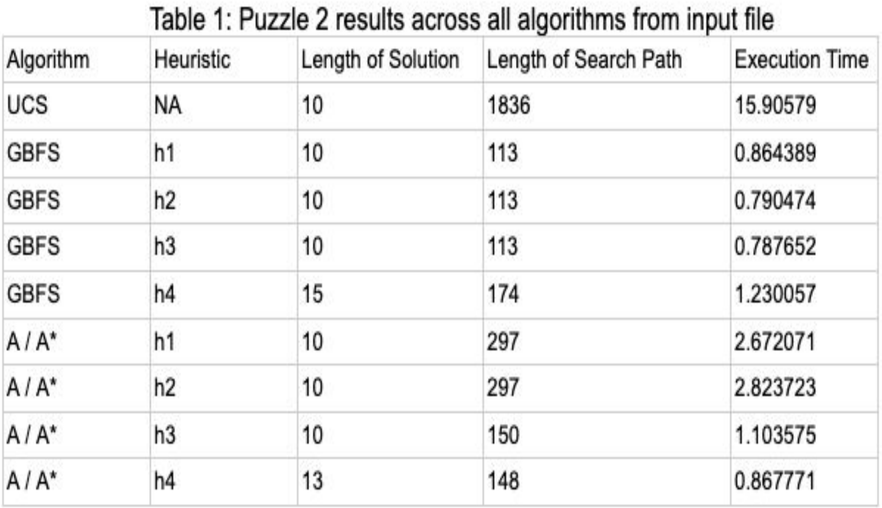


Table 1: Puzzle 2 results across all algorithms from input file

| Algorithm | Heuristic | Length of Solution | Length of Search Path | Execution Time |
|---|---|---|---|---|
| UCS | NA | 10 | 1836 | 15.90579 |
| GBFS | h1 | 10 | 113 | 0.864389 |
| GBFS | h2 | 10 | 113 | 0.790474 |
| GBFS | h3 | 10 | 113 | 0.787652 |
| GBFS | h4 | 15 | 174 | 1.230057 |
| A / A* | h1 | 10 | 297 | 2.672071 |
| A / A* | h2 | 10 | 297 | 2.823723 |
| A / A* | h3 | 10 | 150 | 1.103575 |
| A / A* | h4 | 13 | 148 | 0.867771 |

## h1

- h1(n) = number of blocked cars
- Logical heuristic since we want to move cars in front of A
- Doesn’t accurately estimate the number of moves to completion
- Gave good results compared to other heuristics

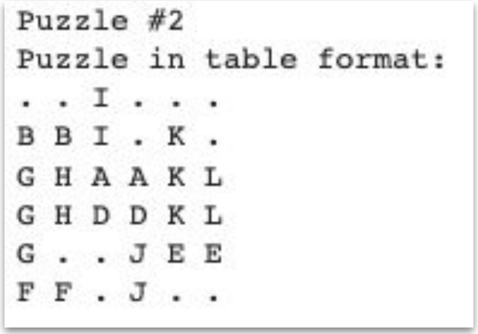
## h2

- Exactly the same as h1 in almost all cases.
- If vertical car in front of A -> number of blocked positions = number of blocked cars
- Only time it is different is if there is a horizontal car in front of A which is rare
- In that case h2 had a slightly better performance than h1

# h3

- GBFS → same output as h1 since it only considers h(n) and every h(n) is being multiplied with a constant value. Since all of the will have same denominator, it will act as h1 exactly.
- A → usually it is gives worst effects then h1 and h2. Since algorithm A considers both g(n) and h(n) then the constant number is having effect and worsening the length of the solution. Different constant values will have different effects.

# h4



```
Puzzle #2
Puzzle in table format:
. . I . . .
B B I . K .
G H A A K L
G H D D K L
G . . J E E
F F . J . .
```

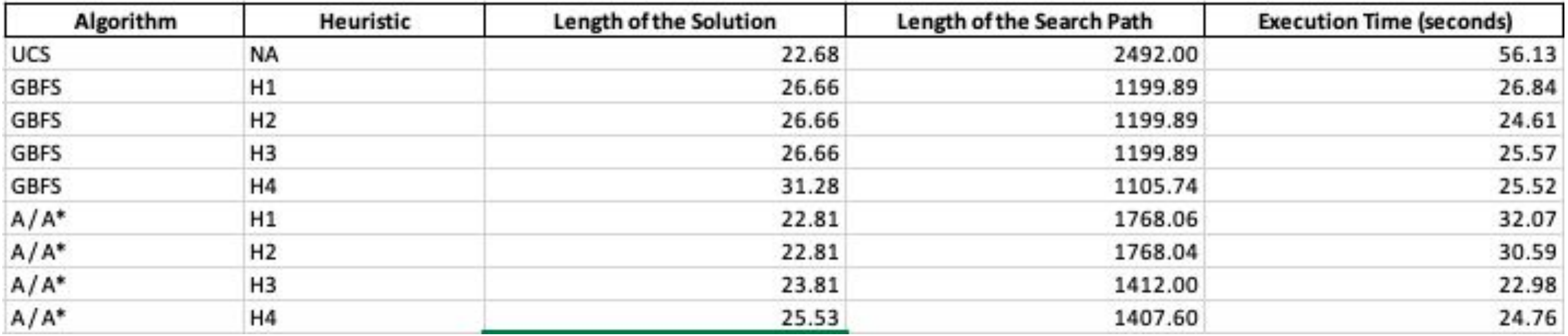
h4(n) = 7 in examples

h4(n) = all blocked positions to the right of A

Resulted in a faster algorithm (because A always wanted to move to the right)

However, no optimal moves since A was moving more than it had too.

# Concluding on the Algorithms and Heuristics



| Algorithm | Heuristic | Length of the Solution | Length of the Search Path | Execution Time (seconds) |
|---|---|---|---|---|
| UCS | NA | 22.68 | 2492.00 | 56.13 |
| GBFS | H1 | 26.66 | 1199.89 | 26.84 |
| GBFS | H2 | 26.66 | 1199.89 | 24.61 |
| GBFS | H3 | 26.66 | 1199.89 | 25.57 |
| GBFS | H4 | 31.28 | 1105.74 | 25.52 |
| A/A* | H1 | 22.81 | 1768.06 | 32.07 |
| A/A* | H2 | 22.81 | 1768.04 | 30.59 |
| A/A* | H3 | 23.81 | 1412.00 | 22.98 |
| A/A* | H4 | 25.53 | 1407.60 | 24.76 |

Lowest cost solution -> UCS

Lowest search path length and execution time -> GBFS

Combination of both -> A

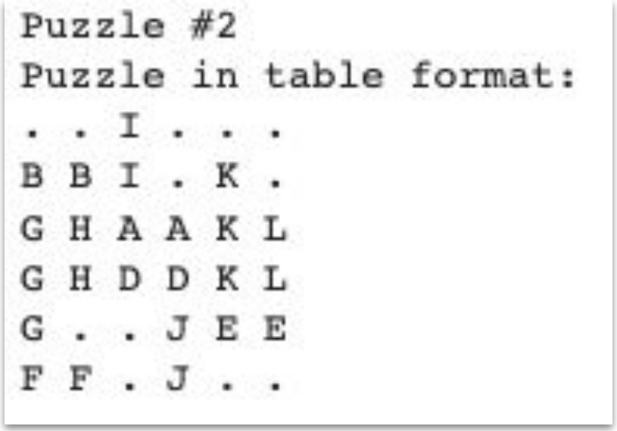
Lowest cost solution and quicker time -> A* (known from theory)

# Interesting Facts

- No possible best algorithm, depends on requirements, what you want from algorithm

- Other ideas for h4

- A* implementation



```
Puzzle #2
Puzzle in table format:
. . I . . .
B B I . K .
G H A A K L
G H D D K L
G . . J E E
F F . J . .
```

# Thank you!

# Questions?