**CS 4341 AI Assignment 1 - A\* Search**

**Write-Up**

**Keith DeSantis, John Higgins, Yonghua Wang**

[**Link to spreadsheet containing our data**](https://docs.google.com/spreadsheets/d/1RnJ0TP-bWAO4zsFsCcyXDKvqnSY35wu8BdJvZXjx0hg/edit?usp=sharing)

**Disclaimer and Notes:**

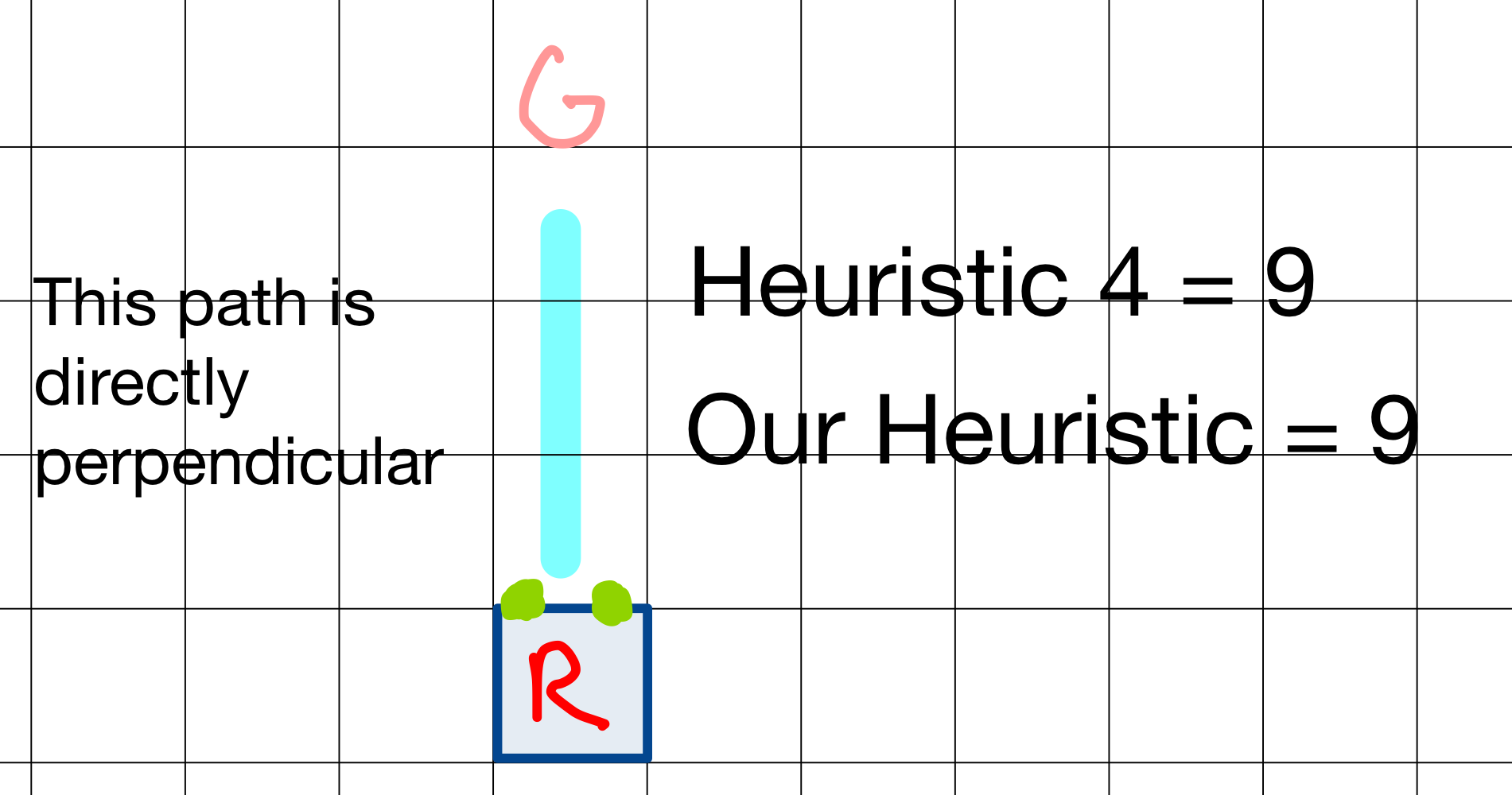
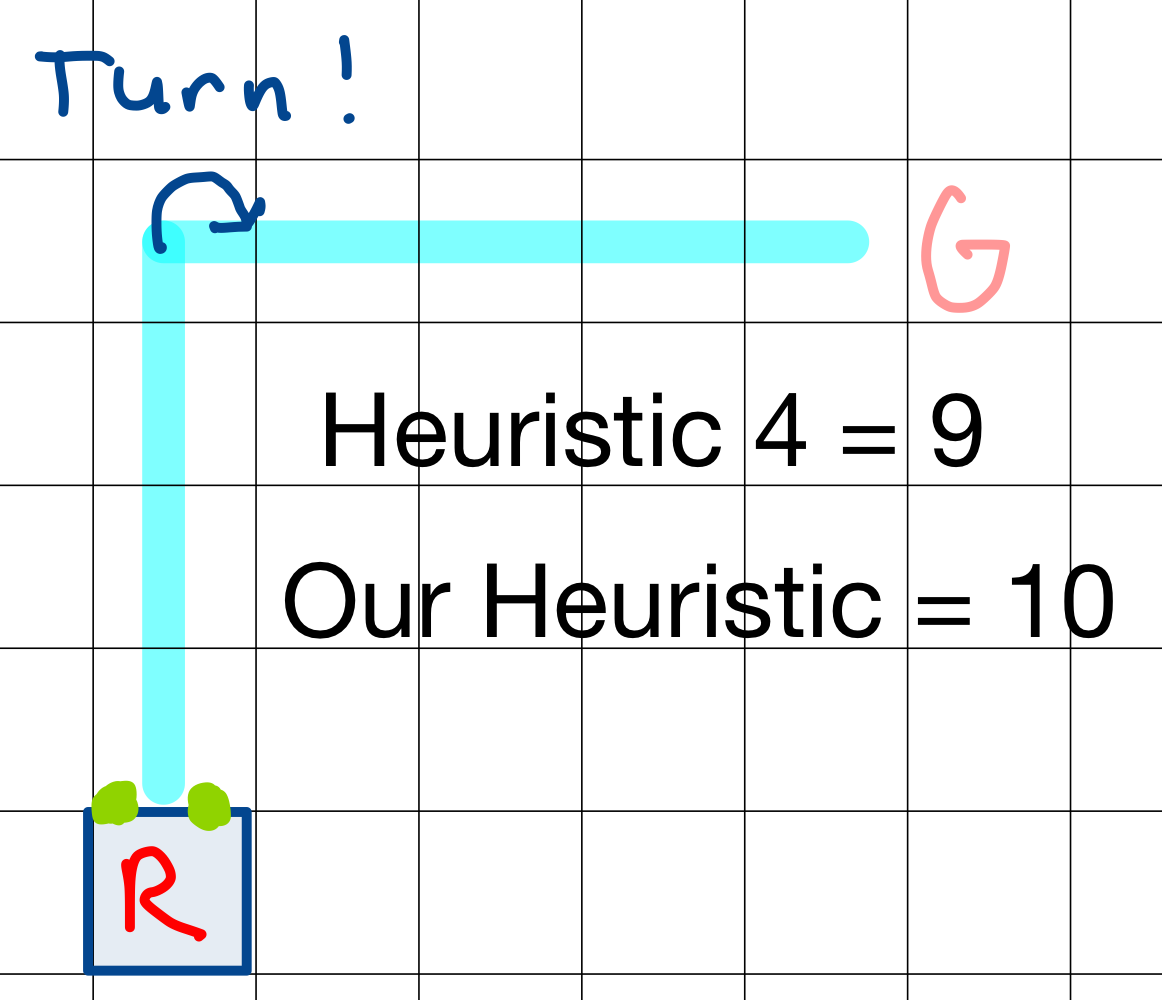
This report has the basic information requested (effective branching factor and number of nodes expanded) for our trials, but we have also included a spreadsheet which has much more information on each trial and all the averages, some values of which we reference in calculations here.

Another note is that while we gave our best estimates for board sizes in terms of dimensions (i.e. 7x7, 9x9, etc.) the actual time to solve boards varies wildly depending on the layout of the board and the positions of the Start and Goal markers, so sometimes a better metric for “size” is the depth of the answer.

Finally, we had a lot of trouble generating 10 boards that heuristic 1 took ~ 30 seconds for, partially because of the randomness of board generation and partially because of how it may differ from device to device. We were able to find a few boards that took in the neighborhood of 10-30 seconds, and other boards that took around 3-10 seconds. The data provided however was sound and useful for our analysis.

**New Heuristic:**

We came up with our heuristic after discussion about how heuristic 4 (horizontal + vertical) fails to account for the cost of a turn. For any given path, aside from those that are directly perpendicular to the robot, the robot must make a minimum of one turn. Given that the smallest possible value for a turn cost is one (the turn space has a terrain cost of 1, it is halved and rounded up to 1), we add value of one to all paths that are not directly perpendicular to the robot



The impact this heuristic has on A-Star is that it more realistically calculates the cost of the path. Initially we reasoned that this would simply scale up all paths, given that many paths turn, but running our simulation reduced the number of nodes expanded and the branching factor. We then realized that this first prioritizes straight line paths over other paths.

**Trial Data:**

### Board 1:

|  |  |  |
| --- | --- | --- |
| **Heuristic** | **Number Nodes Expanded** | **Effective Branching Factor** |
| **1 - 0** | 3858213 | 3.1955 |
| **2 - Min(vertical, horizontal)** | 336292 | 3.1202 |
| **3 - Max(vertical, horizontal)** | 108807 | 3.0299 |
| **4 - Vertical + Horizontal** | 30589 | 3.2407 |
| **5 - Accounting for Turn** | 14461 | 3.1294 |
| **6 - #5 x 3** | 35 | 2.6923 |

**Score 1-5: 83 Path : FLFRFLFRBF**

**Score 6: 82 Path : LBFRBFBF**

### Board 2:

|  |  |  |
| --- | --- | --- |
| **Heuristic** | **Number Nodes Expanded** | **Effective Branching Factor** |
| **1 - 0** | 3792247 | 2.9095 |
| **2 - Min(vertical, horizontal)** | 839838 | 2.8798 |
| **3 - Max(vertical, horizontal)** | 424865 | 2.8839 |
| **4 - Vertical + Horizontal** | 60414 | 2.8787 |
| **5 - Accounting for Turn** | 29517 | 2.8662 |
| **6 - #5 x 3** | 66 | 2.5384 |

**Score 1-5: 80 Path : FBFFRFBF**

**Score 6: 80 Path : FBFFRFBF**

### Board 3:

|  |  |  |
| --- | --- | --- |
| **Heuristic** | **Number Nodes Expanded** | **Effective Branching Factor** |
| **1 - 0** | 7725055 | 3.2244 |
| **2 - Min(vertical, horizontal)** | 1806731 | 3.2106 |
| **3 - Max(vertical, horizontal)** | 389341 | 3.1428 |
| **4 - Vertical + Horizontal** | 94805 | 3.2602 |
| **5 - Accounting for Turn** | 56722 | 3.0741 |
| **6 - #5 x 3** | 63 | 2.8636 |

**Score 1-5: 82 Path : RFRBFFLFRF**

**Score 6: 82 Path : RFRBFFLFRF**

### Board 4:

|  |  |  |
| --- | --- | --- |
| **Heuristic** | **Number Nodes Expanded** | **Effective Branching Factor** |
| **1 - 0** | 3596626 | 3.4488 |
| **2 - Min(vertical, horizontal)** | 428418 | 3.3596 |
| **3 - Max(vertical, horizontal)** | 410090 | 3.5631 |
| **4 - Vertical + Horizontal** | 50236 | 3.4211 |
| **5 - Accounting for Turn** | 35269 | 3.3458 |
| **6 - #5 x 3** | 26 | 2.8888 |

**Score 1-5: 84 Path : FLFRBFLFF**

**Score 6: 83 Path : FBFLFFF**

### Board 5:

|  |  |  |
| --- | --- | --- |
| **Heuristic** | **Number Nodes Expanded** | **Effective Branching Factor** |
| **1 - 0** | 1302013 | 3.0274 |
| **2 - Min(vertical, horizontal)** | 218175 | 3.2003 |
| **3 - Max(vertical, horizontal)** | 130275 | 3.1803 |
| **4 - Vertical + Horizontal** | 15743 | 3.1014 |
| **5 - Accounting for Turn** | 11077 | 3.1088 |
| **6 - #5 x 3** | 46 | 3.0666 |

**Score 1-5: 84 Path : LFFFRFBF**

**Score 6: 84 Path : LFFFRFBF**

### Board 6:

|  |  |  |
| --- | --- | --- |
| **Heuristic** | **Number Nodes Expanded** | **Effective Branching Factor** |
| **1 - 0** | 2562649 | 3.2141 |
| **2 - Min(vertical, horizontal)** | 633890 | 3.2509 |
| **3 - Max(vertical, horizontal)** | 327161 | 3.456 |
| **4 - Vertical + Horizontal** | 48234 | 3.2496 |
| **5 - Accounting for Turn** | 28518 | 3.3346 |
| **6 - #5 x 3** | 44 | 2.75 |

**Score 1-5: 81 Path : RRBFBFRBF**

**Score 6: 81 Path : RRBFBFRBF**

### Board 7:

|  |  |  |
| --- | --- | --- |
| **Heuristic** | **Number Nodes Expanded** | **Effective Branching Factor** |
| **1 - 0** | 4261480 | 3.0452 |
| **2 - Min(vertical, horizontal)** | 727213 | 3.0747 |
| **3 - Max(vertical, horizontal)** | 240059 | 2.995 |
| **4 - Vertical + Horizontal** | 32305 | 2.9934 |
| **5 - Accounting for Turn** | 12918 | 3.002 |
| **6 - #5 x 3** | 80 | 2.5806 |

**Score 1-5: 81 Path : LLBFFRFBFFLF**

**Score 6: 81 Path : LLBFFRFBFFLF**

### Board 8:

|  |  |  |
| --- | --- | --- |
| **Heuristic** | **Number Nodes Expanded** | **Effective Branching Factor** |
| **1 - 0** | 2773793 | 3.3003 |
| **2 - Min(vertical, horizontal)** | 450989 | 3.1751 |
| **3 - Max(vertical, horizontal)** | 269424 | 3.0319 |
| **4 - Vertical + Horizontal** | 37764 | 3.2648 |
| **5 - Accounting for Turn** | 12451 | 3.1617 |
| **6 - #5 x 3** | 38 | 2.7142 |

**Score 1-5: 81 Path : LFBFRFLFRBF**

**Score 6: 80 Path : BFFLBFFF**

### Board 9:

|  |  |  |
| --- | --- | --- |
| **Heuristic** | **Number Nodes Expanded** | **Effective Branching Factor** |
| **1 - 0** | 6175267 | 3.2668 |
| **2 - Min(vertical, horizontal)** | 5576482 | 3.334 |
| **3 - Max(vertical, horizontal)** | 250331 | 2.9923 |
| **4 - Vertical + Horizontal** | 213720 | 3.1704 |
| **5 - Accounting for Turn** | 203368 | 3.1894 |
| **6 - #5 x 3** | 6709 | 3.1074 |

**Score 1-5: 80 Path : FRFFFBFRF**

**Score 6: 77 Path : RBFBFF**

### Board 10:

|  |  |  |
| --- | --- | --- |
| **Heuristic** | **Number Nodes Expanded** | **Effective Branching Factor** |
| **1 - 0** | 8004248 | 3.0543 |
| **2 - Min(vertical, horizontal)** | 2440092 | 3.0841 |
| **3 - Max(vertical, horizontal)** | 363817 | 3.204 |
| **4 - Vertical + Horizontal** | 103906 | 3.1243 |
| **5 - Accounting for Turn** | 59037 | 3.2658 |
| **6 - #5 x 3** | 45 | 3 |

**Score 1-5: 79 Path : LLBFLFBFBF**

**Score 6: 79 Path : LLBFLFBFBF**

**Analysis:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Heuristic** | **Mean Effective Branching Factor** | **Mean Nodes Expanded** | **Mean Score** |
| **1 - 0** | 3.16863 | 4405159.1 | 81.5 |
| **2 - Min(horizontal, vertical)** | 3.16893 | 1345812 | 81.5 |
| **3 - Max(horizontal, vertical)** | 3.14792 | 291417 | 81.5 |
| **4- Horizontal + Vertical** | 3.17046 | 68771.6 | 81.5 |
| **5 - Accounting for Turn** | 3.14778 | 46333.8 | 81.5 |
| **6 - #5 x 3** | 2.82019 | 715.2 | 80.9 |

**How do the 5 heuristics vary in effectiveness?**

Since heuristics 1-5 always produces the same score value, effectiveness will be defined by the number of nodes expanded as this is most closely related to computations computed. For each of the heuristics the latter dominates the former (Heuristic 2 dominates Heuristic 1), this leads to a decrease in nodes expanded throughout heuristics.

Heuristic 1 expands the most nodes because it is practically exploring at random. With a heuristic of 0 the priority for each move will solely be the total cost it has taken to get to the point it is at. Heuristic 2 on average expanded less than half of the nodes of Heuristic 1. This is because it better approaches the actual cost of the path, but still is not very intelligent as it only represents part of the path. Heuristic 3 explores less than half of those explored by Heuristic 2, not because it is adding intelligence, but because it dominates it. Heuristic 4 is an enormous improvement from the first 3 heuristics because it adds a more intelligent way of calculating the heuristic. By incorporating both horizontal and vertical distance the path is a best case underestimate for the path it will take. Finally, Heuristic 5 is the best of any of the heuristics because it further adds intelligence based on the game mechanic turn. At the beginning of the file is an explanation for why this better measures a path.

**How much gain is there to using any heuristic (#1 vs #2)? Is #5 noticeably more effective than the other heuristics?**

In general the larger-indexed heuristic performed better. That is to say, heuristic #1 will be slower and expand more nodes than heuristic #2, and similarly #2 will expand more nodes than #3, #3 to #4, etc. Heuristic #5 is noticeably more effective than the other 4 admissible heuristics.

**For heuristic #6 how does its solution quality compare with #5? Is it performing noticeably worse? How much more efficient is it?**

In terms of number of nodes expanded, heuristic #6 always outperformed heuristics #5. The branching factor might stay the same or slightly decrease and on average decreased. However, for nodes expanded, there is a huge difference (up to ~65x according to our results).

What is worth mentioning is that H6 is not performing noticeably worse. It sometimes even finds the same optimal solution as other heuristics. Even if it underperforms sometimes, the score does not decrease significantly, usually by less than 5. Considering the significantly less number of nodes expanded, we believe it’s worth trading a small decrease on score with a faster running program.

**Finally, investigate the maximum problem size your program can handle. How large of a map can you solve in 30 seconds using Heuristic 5?**

Based on our trials, heuristic 5 expands approximately:

46333.8 moves expanded / .0335s (33.5 ms) = 1,383,098 moves/second.

Therefore, given 30 seconds heuristic 5 could expand around 40,000,000 moves.

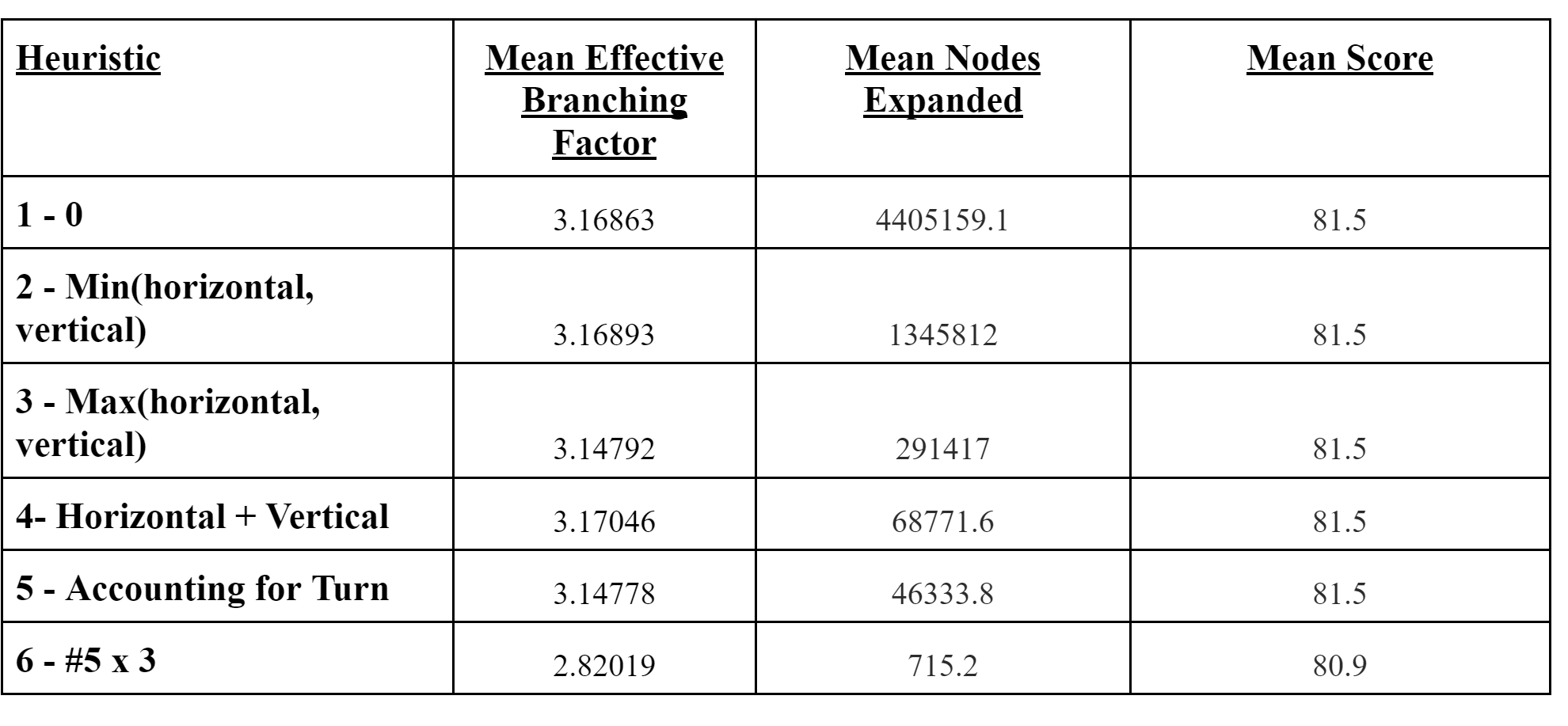
So the spacial complexity of the program would be ~40,000,000. The spacial complexity of A\* is , and since we have an effective branching factor we are able to solve 40,000,000 = , getting that d = ~15. Meaning it would most take heuristic 5 approximately 30 seconds to solve a board with depth 15. We reason that this would equate to around board of size 8x8, however depth and board dimensions are not equivalent, and similarly sized boards vary greatly in solve time.

**How much memory is needed?**

Using the equation , for heuristic 5, we have an average b of 3.14778, average d of 9.6 and average memory usage of 30481618.4 bytes. Solving this equation we get k = 505 (~504.88916) for heuristic 5.

From there, we can plug in our depth of 15 for d and solve for memory usage. Solving, we get memory usage = 1.49 x bytes, or about 15 GB.

**Perform a “back of the envelope” calculation for how large a board you could solve with 16 GB of memory using each of the 6 heuristics (you can assume the board is square for this analysis).**



**Heuristic 1**

Using the equation , for heuristic 1, we have an average b of 3.16863, average d of 9.6 and average memory usage of 684523686.4 bytes. Solving this equation we get k = ~ 10642 bytes for heuristic 1.

Using our k value and plugging in 16 GB for memory usage, we can solve for d.

>>>> d = ~14 (14.3292).

So heuristic 1 could solve a board with depth ~14 with 16GB of memory (around a 7x7 ish).

**Heuristic 2**

Using the equation , for heuristic 2, we have an average b of 3.16893, average d of 9.6 and average memory usage of 297900843.2 bytes. Solving this equation we get k = ~ 4627 bytes for heuristic 2.

Using our k value and plugging in 16 GB for memory usage, we can solve for d.

>>>> d = ~15 (15.0511)

So heuristic 2 could solve a board with depth ~15 with 16GB of memory (around a 8x8 ish).

**Heuristic 3**

Using the equation , for heuristic 3, we have an average b of 3.14792, average d of 9.6 and average memory usage of 153883613.6 bytes. Solving this equation we get k = 2547 bytes for heuristic 3.

Using our k value and plugging in 16 GB for memory usage, we can solve for d.

>>>> d = ~ 16 (15.6578).

So heuristic 1 could solve a board with depth ~16 with 16GB of memory (around a 9x9 ish).

**Heuristic 4**

Using the equation , for heuristic 4 , we have an average b of 3.17046, average d of 9.6 and average memory usage of 43018007.2 bytes. Solving this equation we get k = ~665 (665.08387) bytes for heuristic 5.

Using our k value and plugging in 16 GB for memory usage, we can solve for d.

>>>> d =~ 17 (16.7251)

So heuristic 4 could solve a board with depth ~17 with 16GB of memory (around a 9x9 ish).

**Heuristic 5**

Using the equation , for heuristic 5 , we have an average b of 3.14778, average d of 9.6 and average memory usage of 30481618.4 bytes. Solving this equation we get k = ~505 (504.88916) bytes for heuristic 5.

Using our k value and plugging in 16 GB for memory usage, we can solve for d.

>>>> d = ~17 (17.0698)

So heuristic 5 could solve a board with depth ~15 with 16GB of memory (around a 8x8 ish).

**Heuristic 6**

Using the equation , for heuristic 6, we have an average b of 2.82019, average d of 8.6 and average memory usage of 6134672 bytes. Solving this equation we get k = 823 (823.47233) bytes for heuristic 6.

Using our k value and plugging in 16 GB for memory usage, we can solve for d.

>>>> d = ~18 (18.408).

So heuristic 6 could solve a board with depth ~18 with 16GB of memory (around a 10x10 ish).

**How large of a board could you solve with Heuristic 6 and 16GB of memory? How long would it take to solve that board?**

Using the equation , for heuristic 6, we have an average b of 2.82019, average d of 8.6 and average memory usage of 6134672 bytes. Solving this equation we get k = 823 (823.47233) bytes for heuristic 6.

Using our k value and plugging in 16 GB for memory usage, we can solve for d.

>>>> d = ~18 (18.408).

So heuristic 6 could solve a board with depth ~18 with 16GB of memory (around a 10x10 ish).

Heuristic 6 expanded around:

715.2 moves expanded / 0.0032 (3.2 ms) = 223,500 moves/second.

The spacial complexity of a board with depth 18 for heuristic six is

Meaning there would be moves to be made.

So, with moves at 223,500 moves/second, it would take heuristic 6:

or ~ 570 seconds (10 ish minutes).

**How much memory would you need for a problem that requires 24 hours to solve with Heuristic 5? With heuristic 6?**

**Heuristic 5**

Using the equation , for heuristic 5 , we have an average b of 3.14778, average d of 9.6 and average memory usage of 30481618.4 bytes. Solving this equation we get k = ~505 (504.88916) bytes for heuristic 5.

Using our k value and plugging in 16 GB for memory usage, we can solve for d.

>>>> d = ~17 (17.0698)

So heuristic 5 could solve a board with depth ~17 with 16GB of memory (around a 8x8 ish).

Heuristic 5 expanded around:

46333.8 moves expanded / 0.0335 (33.5 ms) = 1,383,099 (1,383,098.50746) moves/second.

The time taken is 24 hours (86400 seconds).

So, the number of moves expanded in that time would be 86,400 \* 1,383,099 = 119,499,753,600 moves.

Given that we know the effective branching factor of heuristic 5 is 3.14778 we can solve

To get d = ~22 (22.2435).

Then, using the equation we can figure out the memory usage, since we know k for heuristic 5 is about 505 from earlier calculations.

505 \* = bytes = ~ 46,000 GB.

**Heuristic 6**

We know H6’s average moves per second from the prior calculations (~ 223,500). The time taken is 24 hours (86400 seconds).

So, the number of moves expanded in that time would be 86,400 \* 223,500 = 19,310,400,000 moves.

Given that we know the effective branching factor of heuristic 6 is 2.82019 we can solve

To get d = ~23 (22.8432).

Then, using the equation we can figure out the memory usage, since we know k for heuristic 6 is about 823 from earlier calculations.

823 \* = bytes = ~ 19,000 GB.

**Boards:**

**Board 1:**

4 G 4 3 6

2 9 9 8 6

4 2 1 9 5

1 4 2 3 8

2 3 5 S 1

**Board 2:**

4 1 2 4 4

2 1 4 G 7

5 3 9 1 3

6 3 1 7 8

4 5 6 8 5

S 3 4 9 1

**Board 3:**

1 S 2 6 2

3 5 9 2 3

5 8 1 4 1

4 3 4 2 4

3 8 7 G 1

**Board 4:**

9 G 3 3 4

1 3 6 5 6

4 9 8 1 1

4 2 8 4 S

9 3 5 9 4

**Board 5:**

4 3 8 7 7

5 G 5 1 5

8 4 1 8 3

4 2 5 9 4

7 3 2 2 S

**Board 6:**

1 6 9 S 4

2 8 3 7 2

4 4 1 2 1

9 3 6 9 2

4 G 8 3 6

**Board 7:**

6 7 3 5 S

2 9 3 2 8

3 2 8 4 1

1 2 4 3 1

G 6 7 8 3

**Board 8:**

G 3 4 8 2

3 9 7 5 3

1 2 7 3 5

4 1 9 S 4

1 1 1 1 4

**Board 9:**

3 2 3 4 7 1

S 6 6 9 9 G

4 7 2 5 8 7

5 4 5 6 1 9

**Board 10:**

3 S 9 8 1 3 9

7 9 2 7 3 7 5

7 3 2 8 2 3 G

4 7 4 9 9 5 6