**UNIVERSITY OF MACEDONIA** 

DEPARTMENT: APPLIED INFORMATICS

SUBJECT: ARTIFICIAL INTELLIGENCE

PROFESSOR: REFANIDIS IOANNIS

FIRST NAME: GEORGE

LAST NAME: PAPOULIOS

REG. NUMBER: DAI 16080

# Peg Solitaire Computational Study

# INTRODUCTION

The objective of this assignment was the creation of a program which solves Peg Solitaire type problems, with the use of two different search algorithms. These two algorithms are the “depth first search” and the “best first search”. The input files should have the following form:

6 5

0 0 2 0 0

0 2 1 2 0

2 1 1 1 2

0 2 1 2 0

0 2 1 2 0

0 0 2 0 0

Where the first two numbers represent the dimensions of the problem

0: stands for “out of board”

1: stands for “peg”

2: stands for “empty spot”

And the puzzle grid must be rectangular.  
The solution file exported by the program should have the following form:

5  
3 3 3 5  
5 3 3 3  
3 2 3 4  
3 5 3 3  
2 3 4 3  
  
Where the first number is the number of moves and the following rows are the moves needed to reach the solution. The solution given here is a solution produced from the program, for the problem above. The two digits in every row represent the starting coordinates of the peg moved and the other two  
its new coordinates on the board.

**VARIATIONS WITH THE REQUESTED SOLUTION**

A)

The program requested from the exercise should have three parameters.

1. The name of the input file
2. The search method
3. The name of the output file

The program should be called with a computer command like this:

“Pegsol.exe depth input.txt solution.txt”

The program produced, does not exactly work like that. The user runs the program and a frame appears on the screen. With this, the user is able to choose the input file from the explorer and then select the search method. The solution file is automatically saved in the workspace.

B)  
The exercise asks for a second program which would test the solutions produced from the main program. Something like that was not created. The solutions produced by the main program are working anyway because the test is made inside the program itself. Given the starting board and the moves made by the search algorithm the program checks every single time the number of pegs left on the board, so we know if we have a solution or not. Also, the moves that were made from the program are very specific, and checks every time before it makes a move which moves are valid. So there is no chance that the program produces a false solution.   
Something that was not created but it would be a useful tool, is a program which somehow tests if the starting Peg-Puzzle has a solution to begin with. The program produced in this exercise can answer that question only when the whole tree has been accessed. In small problems this is a very easy job for the “depth first search”  
but in bigger problems we would need a program made for that difficult operation.

**PARTS OF CODE WHICH HELP IN BETTER COMPREHENSION**

If you would like to solve a big problem with the program and you have the code available then there are some lines of comments which contain some console outputs that help in comprehension and make the waiting time much more interesting.  
In the “Solver” class lines {127,146,160,167,173}, which are in the “BestFirst Search” method contain some console outputs which inform the user every single time

1. Which node we are currently visiting
2. What is the value of the current node
3. And in the end(When solution is found) prints the total number of nodes visited. (Variable –> int f )

Something similar exists in “Solver” class lines {69,76,90,101} in the “Depth First Search”. Those lines also inform the user about the current node and print the total number of the visited nodes.  
The interesting part if you enable those lines of code is that you can actually see how much faster the depth first search is and the bigger consistency of the best first search. Those outputs where very helpful during developing, and lead to many vital improvements. These difficulties and improvements are all referenced on the last chapter of this paper.

**COMPUTANIONAL STUDY**

The program was launch for 10 different PegSolitaire problems with different levels of difficulty. Three – easy problems, Two – medium difficulty problems, Four - difficult problems and One – very difficult. Each algorithm was launched five times for every problem. If an algorithm could not solve the peg solitaire within five minutes the program was manually terminated. The results and the statistics are represented below.

EQUIPMENT:

The program was launch in a laptop with:

RAM: 4GB

CPU: i3 – 5010U

Base Frequency: 2.10 GHz

Cores: 2

Of Threads: 4

Cache: 3MB

Bus Speed: 5GT/s DMI2

**Peg-Puzzle 1**

Name: Simplest Cross

Difficulty: Very Easy

Number of Pegs: 6

Input file:

6 5

0 0 2 0 0

0 2 1 2 0

2 1 1 1 2

0 2 1 2 0

0 2 1 2 0

0 0 2 0 0

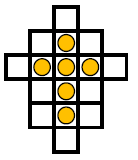


Image1:Simplest Cross

Board 1: Simplest Cross Results (Given Outputs)

|  |  |  |  |
| --- | --- | --- | --- |
| Output No. 1 | Output No. 2 | Output No. 3 | Output No.4 |
| 3 3 3 1  5 3 3 3  3 4 3 2  3 1 3 3  3 3 1 3 | 3 3 3 5  5 3 3 3  3 2 3 4  3 5 3 3  2 3 4 3 | 3 3 3 1  5 3 3 3  3 4 3 2  3 1 3 3  2 3 4 3 | 3 3 3 5  5 3 3 3  3 2 3 4  3 5 3 3  3 3 1 3 |

**Depth First Algorithm**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | FIRST LAUNCH | SECOND LAUNCH | THIRD LAUNCH | FOURTH LAUNCH | FIFTH LAUNCH |
| Time taken | 8 milliseconds | 4 milliseconds | 1 millisecond | 5 milliseconds | 2 milliseconds |
| Given Output | Output No.1 | Output No. 2 | Output No. 1 | Output No. 3 | Output No.3 |
| Nodes Visited | 11 | 16 | 5 | 18 | 6 |

**Best First Search Algorithm**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | FIRST LAUNCH | SECOND LAUNCH | THIRD LAUNCH | FOURTH LAUNCH | FIFTH LAUNCH |
| Time taken | 113milliseconds | 210 milliseconds | 201 milliseconds | 113 milliseconds | 114 milliseconds |
| Given Output | Output No.2 | Output No.4 | Output No. 2 | Output No.2 | Output No.2 |
| Nodes Visited | 6 | 11 | 11 | 6 | 6 |

Comments: Both searches had some good results here. At the beginning the heuristic function of best first search was the following:  
  
**Value** = Max Horizontal Peg Distance + Max Vertical Peg Distance + Isolated Pegs + **Number Of Pegs** + **Number Of Children**

On the results above the value is determined ONLY by the blue factors, which is the heuristic in which we were eventually led.   
When we launched the program using the starting heuristic, the results of the Best-First search where more or less the same with those represented. This happened because the problem is very small and simple. As we see the outputs are almost the same and both techniques can be led in any of those results.   
For best first algorithm the starting move will always be the one which leads in a dead end. Hopefully the values of its next moves have worse values than the other starting options, so the algorithm will never continue with that mistaken move. Instead it is going to select one of the other three equal valued options where the one (3 3 1 3) also leads to a dead end. If it takes it has to go deep to realize that it is not a solution because it has good valued children. Then it comes back and after visiting 11 nodes gives the solution. If it chooses one of the other options the solution is found after 6 nodes. Because the tree of that puzzle is pretty small and simple the depth-first search has visited all the nodes before the best first has even valued its first node. In that case the depth first algorithm is the by far the best choice.

Peg puzzle 1[Simplest cross]  
Result Board: 1

|  |  |  |
| --- | --- | --- |
|  | **Depth First Algorithm** | **Best First Search Algorithm** |
| ***Average Time*** | ***4 milliseconds*** | ***150.2 milliseconds*** |
| Worst Time | 8 milliseconds | 210 milliseconds |
| Best Time | **1 millisecond** | 113 milliseconds |
| ***Average Nodes*** | **11.2** | **8** |
| Most Nodes | 16 | 11 |
| Least Nodes | **5** | **6** |

Best Choice: **Depth First [1 – 0]**

**Peg-Puzzle 2**

Name: Small Square

Difficulty: Very Easy

Number of Pegs: 8

Input file:

4 3

1 1 1

1 1 1

1 1 2

0 2 2



Image1:Small Square

Board 2: Small Square Results (Given Outputs)

|  |  |
| --- | --- |
| Output No. 1 | Output No. 2 |
| 3 1 3 3  1 1 3 1  2 3 4 3  1 2 3 2  3 1 3 3  4 3 2 3  1 3 3 3 | 1 3 3 3  2 1 2 3  1 1 1 3  2 3 4 3  3 1 3 3  4 3 2 3  1 3 3 3 |

**Depth First Algorithm**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | FIRST LAUNCH | SECOND LAUNCH | THIRD LAUNCH | FOURTH LAUNCH | FIFTH LAUNCH |
| Time taken | 1 milliseconds | 1 milliseconds | 1 millisecond | 1 milliseconds | 1 milliseconds |
| Given Output | Output No.1 | Output No.1 | Output No. 1 | Output No. 2 | Output No.1 |
| Nodes Visited | 15 | 8 | 14 | 11 | 12 |

**Best First Search Algorithm**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | FIRST LAUNCH | SECOND LAUNCH | THIRD LAUNCH | FOURTH LAUNCH | FIFTH LAUNCH |
| Time taken | 370 milliseconds | 400 milliseconds | 396 milliseconds | 376 milliseconds | 394 milliseconds |
| Given Output | Output No.2 | Output No.1 | Output No.2 | Output No.1 | Output No.1 |
| Nodes Visited | 28 | 28 | 28 | 28 | 30 |

Comments: In that puzzle the heuristic used at the beginning was better than the final one. Let us see why:

The special thing about this puzzle is that, almost every starting move is correct. This means that there is not a case in which, one of the first 4 moves eventually leads us to a dead end. The problem is that those first 4 moves have all the same value. So, if the heuristic is not affected by the number of pegs, we are visiting new nodes with fewer pegs and go back to the starting moves again and again, which is pointless because as we said before the four first moves are all going to be correct anyway. This is what made me think that the heuristic has to depend on the total number of pegs. In the next experiments, it is made clear why something like that is not eventually happening.

Peg puzzle 2[Small Square]  
Result Board: 2

|  |  |  |
| --- | --- | --- |
|  | **Depth First Algorithm** | **Best First Search Algorithm** |
| ***Average Time*** | ***1 milliseconds*** | ***387.2 milliseconds*** |
| Best Time | **1 milliseconds** | **370 milliseconds** |
| Worst Time | **1 millisecond** | **396 milliseconds** |
| Average Nodes | **12** | **28.4** |
| Most Nodes | **15** | **30** |
| Least Nodes | **8** | **28** |

Best Choice: By far - **Depth First [2 – 0]**

**Peg-Puzzle 3**

Name: Random\_12\_in\_6x4

Difficulty: Easy

Number of Pegs: 12

Input file:

6 4

1 2 1 2

2 1 1 2

1 1 2 1

2 1 1 2

2 1 2 1

2 2 2 1



Image3: Random\_12\_in\_6x4

Board 3: Random\_12\_in\_6x4 Results (Given Outputs)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Output No. 1 | Output No. 2 | Output No.3 | Output No. 4 | Output No. 5 | Output No.6 | Output No.7 | Output No.8 | Output No.9 | Output No.10 | Output No.11 |
| 3 2 1 2  6 4 4 4  1 3 3 3  4 3 2 3  5 2 3 2  3 1 3 3  3 4 3 2  1 1 1 3  1 3 3 3  3 2 3 4  3 4 5 4 | 2 3 2 1  6 4 4 4  3 1 3 3  5 2 3 2  4 3 2 3  1 1 3 1  3 1 3 3  3 4 3 2  1 3 3 3  3 2 3 4  3 4 5 4 | 6 4 4 4  4 4 2 4  3 1 3 3  5 2 3 2  3 2 3 4  3 4 1 4  1 4 1 2  1 1 1 3  1 3 3 3  4 3 2 3  2 3 2 1 | 2 2 2 4  3 1 3 3  5 2 3 2  4 3 2 3  3 4 1 4  1 4 1 2  1 1 1 3  6 4 4 4  1 3 3 3  3 2 3 4  3 4 5 4 | 3 1 3 3  6 4 4 4  4 4 2 4  5 2 3 2  3 2 3 4  3 4 1 4  1 4 1 2  1 1 1 3  1 3 3 3  4 3 2 3  2 2 2 4 | 1 3 3 3  4 3 4 1  6 4 4 4  4 1 2 1  1 1 3 1  2 2 4 2  3 4 3 2  3 1 3 3  5 2 3 2  3 2 3 4  4 4 2 4 | 1 3 3 3  4 3 4 1  4 1 2 1  1 1 3 1  2 2 4 2  3 4 3 2  3 1 3 3  6 4 4 4  5 2 3 2  3 2 3 4  3 4 5 4 | 6 4 4 4  3 2 1 2  5 2 3 2  3 1 3 3  3 4 3 2  1 3 3 3  1 1 1 3  4 3 2 3  1 3 3 3  3 2 3 4  3 4 5 4 | 6 4 4 4  3 2 1 2  5 2 3 2  1 3 3 3  4 3 2 3  1 1 1 3  3 1 3 3  3 4 3 2  1 3 3 3  3 2 3 4  3 4 5 4 | 6 4 4 4  3 1 3 3  5 2 3 2  2 3 2 1  4 3 2 3  1 1 3 1  3 1 3 3  3 4 3 2  1 3 3 3  3 2 3 4  4 4 2 4 | 6 4 4 4  3 1 3 3  5 2 3 2  3 2 1 2  3 4 3 2  1 3 3 3  1 1 1 3  4 3 2 3  1 3 3 3  3 2 3 4  4 4 2 4 |

**Depth First Algorithm**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | FIRST LAUNCH | SECOND LAUNCH | THIRD LAUNCH | FOURTH LAUNCH | FIFTH LAUNCH |
| Time taken | 1 second | 201 milliseconds | 314 milliseconds | 29 milliseconds | 64 milliseconds |
| Given Output | Output No.1 | Output No.2 | Output No.3 | Output No.4 | Output No.5 |
| Nodes Visited | 14458 | 1864 | 4459 | 864 | 2587 |

**Best First Search Algorithm**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | FIRST LAUNCH | SECOND LAUNCH | THIRD LAUNCH | FOURTH LAUNCH | FIFTH LAUNCH |
| Time taken | 5 seconds | 2 seconds | 2.5 seconds | 3 seconds | 1 second |
| Given Output | Output No. 8 | Output No.9 | Output No.8 | Output No.10 | Output No.11 |
| Nodes Visited | 405 | 161 | 197 | 257 | 71 |

Comments: The Outputs No.6 and No.7 were given with the old heuristic function. With the new one, which is unaffected by the number of pegs and the number of children it is impossible for us to get them. The first move of the best first solution will always be [6 4 4 4], because it is the only one that reduces the Vertical range of the problem. I created that map so that I could test what would happen when the “Children” factor conflicts with someone else. In that case it had a conflict with the “vertical range” factor. The starting move [1 3 3 3] is also acceptable because it creates fewer children than the one that reduces the vertical range, so the two moves have the same value. This is when I realized that, if we have many factors that determine the value of the heuristic function, then we are going have an even more stochastic heuristic because these conflicts lead to completely different results. So after that puzzle, I decided to remove the children factor from the value.

Peg puzzle 3[Random\_12\_in\_6x4]  
Result Board: 3

|  |  |  |
| --- | --- | --- |
|  | **Depth First Algorithm** | **Best First Search Algorithm** |
| ***Average Time*** | ***321.6 milliseconds*** | *2.7 seconds* |
| Worst Time | 1185 milliseconds | 5 seconds |
| Best Time | 29 milliseconds | 1037 milliseconds |
| Average Nodes | 4846.4 | 218.2 |
| Most Nodes | 14458 | 405 |
| Least Nodes | 864 | **71** |

In this puzzle we can see that with the peg’s increase and with a little more complex board the best first search algorithm might seldom give a faster solution than the Depth first. Although the depth’s first average time is still way better. An interesting observation in this board is the fact that we can create an approach of the algorithms speed on a given time.  
Depth-first in its worst launch visited 14458 nodes in one second.  
Best-first on the same time had visited only 71 to find the solution.  
As we see, the differences in both speed and accuracy between those two algorithms are huge.

Best Choice: **Depth First [3 – 0]**

**Peg-Puzzle 4**

Name: Teetotum

Difficulty: Medium Difficulty

Number of Pegs: 21

Input file:

6 7

0 0 2 1 2 0 0

2 1 1 1 1 1 0

1 1 1 2 1 1 2

2 1 1 1 1 1 1

0 2 1 1 1 2 2

0 2 2 1 2 2 0

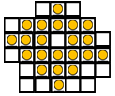


Image 4: Teetotum

Board 4: Teetotum (Given Outputs)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Output No.1 | Output No.2 | Output No.3 | Output No.4 | Output No.5 | Output No.6 | Output No.7 | Output No.8 |
| 4 5 6 5  3 6 5 6  2 5 4 5  3 3 1 3  6 5 6 3  4 4 4 6  4 3 4 1  1 4 3 4  5 6 3 6  4 1 2 1  2 6 4 6  4 7 4 5  6 3 4 3  2 1 2 3  1 3 3 3  3 3 3 5  3 5 5 5  5 5 5 3  5 3 3 3  3 3 3 1 | 5 4 5 6  2 3 2 1  3 6 3 4  4 6 6 6  4 4 4 6  2 5 2 3  2 1 4 1  3 3 3 5  4 7 4 5  4 5 2 5  4 2 2 2  2 6 2 4  1 4 3 4  2 2 2 4  2 4 4 4  4 4 4 2  4 1 4 3  4 3 6 3  6 3 6 5  6 6 6 4 | 3 2 5 2  5 4 3 4  3 5 3 7  4 3 6 3  2 3 4 3  5 5 3 5  4 7 4 5  6 4 6 2  3 4 3 6  3 7 3 5  3 5 1 5  6 2 4 2  4 3 4 1  4 1 2 1  2 1 2 3  2 3 2 5  1 5 3 5  4 5 2 5  2 6 2 4  1 4 3 4 | 3 2 5 2  3 3 1 3  5 4 3 4  4 5 6 5  5 3 3 3  6 4 6 6  3 6 5 6  3 4 3 2  3 1 3 3  1 4 3 4  2 6 2 4  3 4 3 2  2 2 4 2  5 2 3 2  6 6 4 6  4 7 4 5  4 5 2 5  2 5 2 3  1 3 3 3  3 3 3 1 | 1 4 3 4  3 5 3 7  4 4 2 4  6 4 4 4  2 3 2 1  3 2 3 4  5 5 3 5  2 1 4 1  5 3 3 3  4 1 4 3  3 4 3 6  2 5 2 3  4 3 4 5  2 3 4 3  4 6 4 4  2 6 4 6  4 3 4 5  3 7 5 7  4 5 4 7  5 7 3 7 | 1 4 3 4  2 2 2 4  2 5 2 3  4 3 6 3  4 5 4 3  4 7 4 5  2 6 4 6  4 6 4 4  4 3 4 5  2 3 4 3  3 1 3 3  4 2 4 4  4 5 4 3  3 3 5 3  6 3 6 5  6 5 4 5  5 3 5 5  3 4 3 6  5 5 3 5  3 6 3 4 | 1 4 3 4  4 5 6 5  4 7 4 5  2 2 2 4  4 3 2 3  5 3 5 5  3 4 5 4  3 1 3 3  2 3 4 3  4 2 4 4  4 4 4 6  6 5 4 5  6 4 4 4  3 6 5 6  4 4 4 6  5 6 3 6  3 6 3 4  3 4 1 4  2 6 2 4  1 4 3 4 | 1 4 3 4  4 4 2 4  6 4 4 4  3 6 3 4  5 5 3 5  4 7 4 5  3 4 5 4  2 3 2 1  3 2 3 4  2 4 4 4  2 6 2 4  4 5 2 5  2 1 4 1  2 5 2 3  5 3 3 3  4 1 4 3  4 4 4 2  2 3 4 3  4 2 4 4  5 4 3 4 |

**Depth First Algorithm**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | FIRST LAUNCH | SECOND LAUNCH | THIRD LAUNCH | FOURTH LAUNCH | FIFTH LAUNCH |
| Time taken | **Out of time** | 278 milliseconds | 23 seconds | 2 seconds | 36 seconds |
| Given Output | **N/A** | Output No.1 | Output No.2 | Output No.3 | Output No.4 |
| Nodes Visited | **N/A** | 4199 | 760615 | 95724 | 1272033 |

**Best First Search Algorithm**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | FIRST LAUNCH | SECOND LAUNCH | THIRD LAUNCH | FOURTH LAUNCH | FIFTH LAUNCH |
| Time taken | 3.5 seconds | 422 milliseconds | 18 seconds | 1.5 second | 3 seconds |
| Given Output | Output No.5 | Output No.6 | Output No.5 | Output No.7 | Output No.8 |
| Nodes Visited | 252 | 21 | 988 | 88 | 186 |

Comments: This peg puzzle has very many solutions. The best first search will always start with the move [1 4 3 4]. It will be lead to similar solutions and sometimes to the same (like first and third launch). Hopefully for depth – first the starting moves cannot be lethal, but because the number of pegs is 21, the tree is quite deep. So, with multiple bad moves it might get trapped at a part of the tree with big depth and no solution. That chance is not that great. It is even less than 20% actually. There were made some more launches out of the record just to estimate that percentage.  
When best first search had the old node evaluation system, the results were very similar to the represented ones.

Peg puzzle 4 [Teetotum]  
Result Board: 4

|  |  |  |
| --- | --- | --- |
|  | **Depth First Algorithm** | **Best First Search Algorithm** |
| ***Average Time(seconds)*** | 15.3 | 5.28 |
| Worst Time | **>5 minutes** | 18 seconds |
| Best Time(seconds) | **0.2** | **0.4** |
| Average Nodes | **533142.75** | 307 |
| Most Nodes | **1272033** | 988 |
| Least Nodes | 4199 | **21** |
| AVG Speed (Nodes/s) | 34845.91 | 58.14 |
| Failures | 1 | 0 |

This is the first medium difficulty’s puzzle. Here the best first has better results. Depth first in its best will suppress best first search but still the average time is better, without considering the depth’s first failure chance.  
Depth’s first speed is amazingly higher, but best first can be unbelievably accurate. Depth first visits almost 35K nodes every second while best first search only 58. However, the least nodes visited by the best first search on the second launch were 21 which means, that it visited only one node which eventually was out of the solution.

Best Choice: **Best First Search [3 – 1]**

**Peg-Puzzle 5**

Name: Trapezium

Difficulty: Medium Difficulty

Number of Pegs: 20

Input file:

4 8

0 2 2 1 1 2 2 0

0 2 1 1 1 1 2 0

2 1 1 1 1 1 1 2

1 1 1 1 1 1 1 1

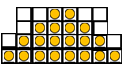


Image 5: Trapezium

Board 5: Trapezium (Given Outputs)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Output No.1 | Output No.2 | Output No.3 | Output No.4 | Output No.5 | Output No.6 | Output No.7 | Output No.8 | Output No.9 | Output No.10 |
| 4 2 2 2  3 6 3 8  1 4 1 6  3 5 1 5  1 5 1 7  2 3 2 5  4 4 4 2  4 1 4 3  4 3 2 3  2 2 2 4  2 5 2 3  4 6 4 4  4 4 2 4  4 8 4 6  2 3 2 5  2 5 2 7  1 7 3 7  3 8 3 6  4 6 2 6 | 4 7 2 7  3 6 1 6  4 5 4 7  3 3 1 3  4 3 4 5  4 8 4 6  2 4 2 6  4 5 4 7  3 5 3 3  4 1 4 3  3 2 3 4  1 5 1 7  1 3 1 5  1 7 3 7  4 7 2 7  2 7 2 5  1 5 3 5  3 5 3 3  4 3 2 3 | 3 3 1 3  3 6 3 8  2 5 2 7  4 2 2 2  4 4 4 2  1 4 1 6  4 1 4 3  4 6 4 4  4 8 4 6  4 3 4 5  4 5 2 5  2 5 2 3  2 2 2 4  3 4 1 4  1 3 1 5  1 5 1 7  1 7 3 7  3 8 3 6  4 6 2 6 | 3 6 1 6  3 3 1 3  2 4 2 6  4 4 2 4  1 4 3 4  1 6 3 6  4 6 4 4  4 8 4 6  3 5 3 3  3 7 3 5  4 3 4 5  4 5 2 5  4 1 4 3  1 5 3 5  4 3 2 3  1 3 3 3  3 2 3 4  3 4 3 6  4 6 2 6 | 4 2 2 2  1 5 1 3  2 5 2 7  3 4 3 2  3 6 3 4  1 3 3 3  3 3 3 5  4 4 4 2  4 6 4 4  4 8 4 6  2 7 4 7  4 7 4 5  4 5 4 3  4 2 4 4  2 2 4 2  4 1 4 3  4 3 4 5  4 5 2 5  2 4 2 6 | 3 6 1 6  2 4 2 2  4 3 2 3  4 1 4 3  3 5 3 3  2 2 2 4  3 2 3 4  2 4 2 6  1 6 3 6  3 7 3 5  4 5 2 5  4 3 4 5  4 6 4 4  4 4 2 4  4 8 4 6  2 4 2 6  1 4 1 6  1 6 3 6  4 6 2 6 | 2 4 2 2  4 3 2 3  4 1 4 3  1 5 1 3  1 3 3 3  4 4 4 2  4 6 4 4  4 8 4 6  2 5 2 7  2 7 4 7  4 7 4 5  4 4 2 4  3 6 3 4  2 4 4 4  4 5 4 3  4 3 4 1  2 2 4 2  4 1 4 3  4 3 2 3 | 2 4 2 2  4 3 2 3  4 1 4 3  1 5 1 3  1 3 3 3  4 4 4 2  2 5 2 7  4 6 2 6  4 8 4 6  3 4 3 6  3 7 3 5  2 7 2 5  4 6 4 4  2 5 4 5  4 5 4 3  4 3 4 1  2 2 4 2  4 1 4 3  4 3 2 3 | 3 3 3 1  3 5 3 3  1 5 3 5  1 4 3 4  4 5 2 5  4 3 4 5  4 6 4 4  4 8 4 6  2 6 2 4  3 7 3 5  2 3 2 5  3 4 3 2  4 1 4 3  3 1 3 3  4 3 4 5  4 6 4 4  2 5 4 5  4 5 4 3  4 3 2 3 | 1 4 1 6  2 4 2 2  4 3 2 3  4 1 4 3  3 5 3 3  2 2 2 4  3 2 3 4  3 7 3 5  1 6 3 6  2 4 2 6  4 5 2 5  4 3 4 5  4 6 4 4  4 8 4 6  2 5 2 7  4 6 2 6  2 7 2 5  4 4 2 4  2 4 2 6 |

**Depth First Algorithm**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | FIRST LAUNCH | SECOND LAUNCH | THIRD LAUNCH | FOURTH LAUNCH | FIFTH LAUNCH |
| Time taken | 15 seconds | 33 seconds | 20 seconds | 2.5 minutes | 8 seconds |
| Given Output | Output No.1 | Output No.2 | Output No.3 | Output No.4 | Output No.5 |
| Nodes Visited | 432097 | 1046582 | 645005 | 5092812 | 258471 |

**Best First Search Algorithm**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | FIRST LAUNCH | SECOND LAUNCH | THIRD LAUNCH | FOURTH LAUNCH | FIFTH LAUNCH |
| Time taken | 579 milliseconds | 1.5 second | 862 milliseconds | 20 seconds | 3 seconds |
| Given Output | Output No.6 | Output No.7 | Output No.8 | Output No.9 | Output No.10 |
| Nodes Visited | 34 | 100 | 49 | 1331 | 284 |

Comments: This Peg puzzle was the most useful during the creation of the evaluation function. As mentioned many times before the starting evaluation function for every node was the following:  
**Value** = Max Horizontal Peg Distance + Max Vertical Peg Distance + Isolated Pegs + **Number of Pegs** + **Number of Children**

After that puzzle the Red variables were removed. The reason why this happened is the fact that best’s first result board looked like this:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | FIRST LAUNCH | SECOND LAUNCH | THIRD LAUNCH | FOURTH LAUNCH | FIFTH LAUNCH |
| Time taken | **Out of time** | **Out of time** | **Out of time** | **Out of time** | **Out of time** |
| Given Output | **N/A** | **N/A** | **N/A** | **N/A** | **N/A** |
| Nodes Visited | **N/A** | **N/A** | **N/A** | **N/A** | **N/A** |

Result board: Old best first search

The problem was the fact that best first search didn’t want to leave any isolated pegs so it usually started with the moves [1 5 1 3] || [1 4 1 6]. Those starting moves hardly lead to a solution. However, while best first search was continuing to remove pegs, the values were becoming better and better, because the “Number of Pegs” was decreasing, even though a solution would rarely be found by following such a path.   
So with this evaluation function, the depth’s first main problem (The fact that it can get trapped in a huge branch which does not contain a solution) was transmitted to the best first. A good idea would be to keep the Number of pegs in the evaluation function multiplied by a coefficient lesser than 1.  
Value = Max Horizontal Peg Distance + Max Vertical Peg Distance +   
Isolated Pegs + ½ **Number of Pegs (or ¼ Number of Pegs).**That change was not occurred, because we would need to make a whole new computational study where we would choose the optimal coefficient. Instead, the whole variable was removed, because even at the most difficult maps best first’s performance was good enough.   
In huge puzzle maps, we might need our search to reach at the deepest nodes as fast as possible, so we can have a better time performance. This is what makes the Number of Pegs to be considered a good addition on the evaluation function. Although this is still happening, because as long as pegs are being removed the vertical and horizontal surface gets smaller and smaller, it surely will not get us very deep as fast as “Number of Pegs” would.

Peg puzzle 5 [Trapezium]  
Result Board: 5

|  |  |  |
| --- | --- | --- |
|  | **Depth First Algorithm** | **Best First Search Algorithm** |
| ***Average Time(seconds)*** | 45.2 | 5.16 |
| Worst Time(seconds) | 150 | 20 |
| Best Time(seconds) | 8 | **0.5** |
| Average Nodes | 1494993.4 | 359.6 |
| Most Nodes | **5092812** | 1331 |
| Least Nodes | 258471 | 34 |
| AVG Speed (Nodes/s) | 33075.08 | 69.69 |

As mentioned above, this Peg Puzzle has many branches that contain no solution. However depth first algorithm is very fast, it visits  
33K (nodes/second) which means that it gets itself out of the area that has no solution on time. Best first has a great performance in both time and accuracy. It has an accuracy of (19/34) = 55% at its best launch and an average accuracy of (19/359.6) = 5%. Depth first visited 5million nodes at its worst launch but it still found a solution. Best first remains slow and accurate. However, in this Peg Puzzle it has a perceived excellence in performance. It has a much better average time and a better “worst time”.

Best Choice: **Best First Search [3 – 2]**

**Peg-Puzzle 6**

Name: Standard Cross

Difficulty: Difficult

Number of Pegs: 32

Input file:

7 7

0 0 1 1 1 0 0

0 0 1 1 1 0 0

1 1 1 1 1 1 1

1 1 1 2 1 1 1

1 1 1 1 1 1 1

0 0 1 1 1 0 0

0 0 1 1 1 0 0

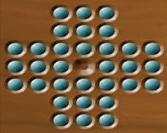


Image 6: Standard Cross

Board 6: Standard Cross (Given Outputs)

|  |
| --- |
| None found |

**Depth First Algorithm**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | FIRST LAUNCH | SECOND LAUNCH | THIRD LAUNCH | FOURTH LAUNCH | FIFTH LAUNCH |
| Time taken | **Out of time** | **Out of time** | **Out of time** | **Out of time** | **Out of time** |
| Given Output | **N/A** | **N/A** | **N/A** | **N/A** | **N/A** |
| Nodes Visited | **N/A** | **N/A** | **N/A** | **N/A** | **N/A** |

**Best First Search Algorithm**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | FIRST LAUNCH | SECOND LAUNCH | THIRD LAUNCH | FOURTH LAUNCH | FIFTH LAUNCH |
| Time taken | **Out of time** | **Out of time** | **Out of time** | **Out of time** | **Out of time** |
| Given Output | **N/A** | **N/A** | **N/A** | **N/A** | **N/A** |
| Nodes Visited | **N/A** | **N/A** | **N/A** | **N/A** | **N/A** |

Comments: When neither of the searches worked on Standard Cross, a thought that the searches do not work in difficult maps came in mind. But, no hasty conclusion was made. The testing continued with different puzzles.

**Peg-Puzzle 7**

Name: Big Square

Difficulty: Difficult

Number of Pegs: 35

Input file:

6 6

1 1 1 1 1 1

1 1 1 1 1 1

1 1 1 2 1 1

1 1 1 1 1 1

1 1 1 1 1 1

1 1 1 1 1 1

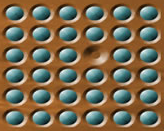


Image 7: Big square

Board 7: Big Square (Given Outputs)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Output No.1 | Output No.2 | Output No.3 | Output No.4 | Output No.5 |
| 3 6 3 4  1 5 3 5  1 6 3 6  2 3 2 5  1 3 1 5  2 1 2 3  1 1 1 3  4 6 2 6  4 4 2 4  4 2 2 2  4 1 2 1  6 1 4 1  6 2 4 2  6 4 4 4  6 6 6 4  6 3 6 5  4 3 6 3  4 1 4 3  3 3 5 3  6 3 4 3  4 4 4 2  1 3 3 3  2 1 2 3  2 3 4 3  4 2 4 4  4 4 4 6  6 5 4 5  5 6 3 6  3 6 1 6  2 4 2 6  4 5 2 5  1 6 1 4  2 6 2 4  1 4 3 4 | 3 6 3 4  1 6 3 6  4 6 2 6  6 6 4 6  1 4 1 6  5 5 3 5  6 4 6 6  3 4 3 6  4 3 4 5  6 2 6 4  3 6 5 6  4 2 6 2  6 1 6 3  4 1 6 1  2 3 4 3  2 1 2 3  1 3 3 3  6 6 4 6  5 4 5 2  1 6 3 6  1 1 1 3  3 3 5 3  4 6 4 4  3 1 3 3  2 5 2 3  6 4 6 2  6 1 6 3  6 3 4 3  4 4 4 2  5 2 3 2  3 2 3 4  1 3 3 3  3 3 3 5  3 6 3 4 | 1 4 3 4  1 2 1 4  2 2 2 4  2 5 2 3  1 5 1 3  4 5 2 5  4 4 2 4  4 2 2 2  6 2 4 2  6 4 4 4  6 6 6 4  6 3 6 5  6 5 4 5  4 6 6 6  4 4 4 6  3 6 5 6  6 6 4 6  1 6 3 6  4 6 2 6  4 3 6 3  4 1 4 3  6 1 4 1  3 3 5 3  6 3 4 3  1 3 3 3  2 1 2 3  4 1 2 1  2 4 2 2  2 6 2 4  2 1 2 3  2 4 2 2  4 3 2 3  2 3 2 1  1 1 3 1 | 3 6 3 4  1 5 3 5  4 5 2 5  3 3 3 5  1 6 3 6  1 3 3 3  4 3 4 5  1 1 1 3  1 3 1 5  3 6 3 4  1 5 3 5  2 4 4 4  5 4 3 4  2 1 2 3  2 3 4 3  5 6 5 4  3 4 3 6  4 2 4 4  6 3 4 3  3 1 3 3  3 3 5 3  3 6 5 6  6 1 6 3  4 5 4 3  6 4 6 2  4 1 6 1  6 1 6 3  6 6 6 4  6 4 6 2  6 2 4 2  4 2 4 4  5 3 5 5  5 6 5 4  4 4 6 4 | 3 6 3 4  3 3 3 5  1 3 3 3  3 2 3 4  5 3 3 3  4 1 4 3  6 1 4 1  3 4 3 6  5 5 3 5  3 1 5 1  2 1 2 3  5 1 5 3  1 1 1 3  3 6 3 4  4 3 4 5  6 4 4 4  4 5 4 3  6 2 6 4  1 6 3 6  2 4 4 4  6 5 6 3  1 5 3 5  4 3 4 5  6 3 4 3  3 6 3 4  4 6 4 4  6 6 4 6  4 3 4 5  4 6 4 4  4 4 2 4  1 4 3 4  3 4 3 2  1 3 3 3  3 2 3 4 |

**Depth First Algorithm**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | FIRST LAUNCH | SECOND LAUNCH | THIRD LAUNCH | FOURTH LAUNCH | FIFTH LAUNCH |
| Time taken | 3 seconds | **Out of time** | **Out of time** | **Out of time** | **Out of time** |
| Given Output | Output No.2 | **N/A** | **N/A** | **N/A** | **N/A** |
| Nodes Visited | 36924 | **N/A** | **N/A** | **N/A** | **N/A** |

**Best First Search Algorithm**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | FIRST LAUNCH | SECOND LAUNCH | THIRD LAUNCH | FOURTH LAUNCH | FIFTH LAUNCH |
| Time taken | 0.8 seconds | **Out of time** | 16 seconds | 15 second | 3.5 minutes |
| Given Output | Output No.1 | **N/A** | Output No.3 | Output No.4 | Output No.5 |
| Nodes Visited | 59 | **N/A** | 943 | 393 | 2222 |

Comments: Here we need to mention that, although “Big Square” has more pegs than “Standard Cross” both algorithms solved it once. We can see that Best first search has no problem in solving this peg solitaire. The time taken is more than those in smaller peg solitaire problems, which is something expected because the least nodes visited will surely be more.

Peg puzzle 7 [Big Square]  
Result Board: 7

|  |  |  |
| --- | --- | --- |
|  | **Depth First Algorithm** | **Best First Search Algorithm** |
| ***Average Time(seconds)*** | Out of time | 60.45 |
| Worst Time(seconds) | Out of time | Out of time |
| Best Time(seconds) | 3 | 0.8 |
| Average Nodes | Millions | 904.25 |
| Most Nodes | Millions | 2222 |
| Least Nodes | 36924 | 59 |
| Failures | 4 | 1 |

Best Choice: **Best First Search [3 –3]**

Best first search acts normally in this peg solitaire. As expected gives results sometimes in unbelievably short time (800 milliseconds) and sometimes it can get lost (Out of time). The fact is that on average it is going to give a solution in the peg solitaire in about a minute.  
Depth first is able to give a solution here in a short amount of time, but on average it is going to get lost.

**Peg-Puzzle 8**

Name: Heart

Difficulty: Difficult

Number of Pegs: 40

Input file:

7 9

0 1 1 1 1 1 1 1 0

1 1 1 1 1 1 1 1 1

1 1 1 2 1 1 1 1 1

0 1 1 1 1 1 1 1 0

0 0 1 1 1 1 1 0 0

0 0 0 1 1 1 0 0 0

0 0 0 0 1 0 0 0 0



Image 8: Heart

Board 8: Heart(Given Outputs)

|  |  |
| --- | --- |
| Output No.1 | Output No.2 |
| 3 2 3 4  3 5 3 3  5 5 3 5  7 5 5 5  3 6 3 4  3 4 3 2  1 3 3 3  4 7 4 5  2 7 4 7  3 9 3 7  2 9 2 7  3 2 3 4  5 5 3 5  4 8 4 6  2 7 4 7  1 6 3 6  1 8 1 6  4 6 2 6  1 5 1 3  3 4 3 6  1 2 1 4  5 7 3 7  3 7 3 5  1 6 3 6  1 4 3 4  6 6 4 6  4 6 2 6  2 6 2 4  2 1 2 3  2 3 2 5  2 5 4 5  5 3 3 3  4 5 4 3  6 4 4 4  3 4 5 4  4 2 4 4  5 4 3 4  3 4 3 2  3 1 3 3 | 3 6 3 4  5 5 3 5  7 5 5 5  4 7 4 5  3 4 3 6  2 7 4 7  3 9 3 7  2 9 2 7  1 4 3 4  1 2 1 4  4 5 6 5  4 4 2 4  3 3 1 3  2 1 2 3  3 1 3 3  4 2 4 4  1 4 3 4  1 6 1 4  1 8 1 6  4 8 4 6  3 7 1 7  5 7 5 5  1 7 1 5  3 6 5 6  6 5 4 5  1 4 1 6  3 3 3 5  4 5 4 3  1 6 3 6  1 3 3 3  6 6 4 6  2 5 4 5  5 3 5 5  3 6 5 6  5 6 5 4  6 4 4 4  3 3 5 3  4 5 4 3  5 3 3 3 |

**Depth First Algorithm**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | FIRST LAUNCH | SECOND LAUNCH | THIRD LAUNCH | FOURTH LAUNCH | FIFTH LAUNCH |
| Time taken | **Out of time** | **Out of time** | **Out of time** | **Out of time** | **Out of time** |
| Given Output | **N/A** | **N/A** | **N/A** | **N/A** | **N/A** |
| Nodes Visited | **N/A** | **N/A** | **N/A** | **N/A** | **N/A** |

**Best First Search Algorithm**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | FIRST LAUNCH | SECOND LAUNCH | THIRD LAUNCH | FOURTH LAUNCH | FIFTH LAUNCH |
| Time taken | **Out of time** | **Out of time** | **Out of time** | 57 Seconds | 9 Seconds |
| Given Output | **N/A** | **N/A** | **N/A** | Output No.1 | Output No.2 |
| Nodes Visited | **N/A** | **N/A** | **N/A** | 1662 | 285 |

|  |  |  |
| --- | --- | --- |
|  | **Depth First Algorithm** | **Best First Search Algorithm** |
| ***Average Time*** | Out of time | 33 seconds |
| Worst Time | Out of time | Out of time |
| Best Time | Out of time | 9 seconds |
| Average Nodes | Millions | 904.25 |
| Most Nodes | Millions | Almost ten thousand |
| Least Nodes | Millions | 285 |
| Failures | 5 | 3 |

Best Choice: **Best First Search [3 – 4]**

Comments: This map has the most pegs (40). Depth first cannot deal with this problem at all. On the other hand best first search can solve this peg solitaire, even though it has a hard time.   
Best first accuracy is pretty good at its best launch (39/285). Best first search can always solve this problem, but because rating of every new node takes time, it is going to be late.

**Peg-Puzzle 9**

Name: European

Difficulty: Difficult

Number of Pegs: 36

Input file:

7 7

0 0 1 1 1 0 0

0 1 1 1 1 1 0

1 1 1 2 1 1 1

1 1 1 1 1 1 1

1 1 1 1 1 1 1

0 1 1 1 1 1 0

0 0 1 1 1 0 0

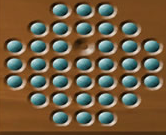


Image 9: European

Board 9: European (Given Outputs)

|  |
| --- |
| Output No.1 |
| 3 6 3 4  5 5 3 5  3 4 3 6  1 5 3 5  4 7 4 5  2 3 2 5  7 5 5 5  4 5 6 5  5 4 3 4  7 3 7 5  7 5 5 5  4 3 2 3  6 3 6 5  4 1 4 3  3 1 3 3  4 3 6 3  5 1 5 3  2 2 2 4  2 5 4 5  6 2 6 4  6 5 6 3  6 3 4 3  4 3 2 3  1 3 3 3  3 3 3 5  1 4 3 4  4 5 6 5  2 6 4 6  3 4 3 6  5 7 5 5  3 6 5 6  6 6 4 6  6 5 4 5  4 5 4 7  3 7 5 7 |

**Depth First Algorithm**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | FIRST LAUNCH | SECOND LAUNCH | THIRD LAUNCH | FOURTH LAUNCH | FIFTH LAUNCH |
| Time taken | **Out of time** | **Out of time** | **Out of time** | **Out of time** | **Out of time** |
| Given Output | **N/A** | **N/A** | **N/A** | **N/A** | **N/A** |
| Nodes Visited | **N/A** | **N/A** | **N/A** | **N/A** | **N/A** |

**Best First Search Algorithm**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | FIRST LAUNCH | SECOND LAUNCH | THIRD LAUNCH | FOURTH LAUNCH | FIFTH LAUNCH |
| Time taken | **Out of time** | **Out of time** | **Out of time** | 17 Seconds | **Out of time** |
| Given Output | **N/A** | **N/A** | **N/A** | Output No.1 | **N/A** |
| Nodes Visited | **N/A** | **N/A** | **N/A** | 501 | **N/A** |

Comments: A difficult peg solitaire. Impossible for depth first to solve it in 5 minutes with this CPU. Best first made it once, solving it in 17 seconds. In this peg solitaire, Best-first faces the same problem with depth first most of the times. It goes deep on branches that have no solution. It does not reach up to the terminal nodes, because it realizes sooner that it is not going to find a solution there, but still in five minutes it cannot do much. When it reaches at depth > 20 it ends up doing a breadth – first kind of search because most branches at every level have the similar values. The efficency of Best first search is 30%. In “boards used/European” directory there are the output files of the 6th and 10th launch, where the program found solutions and pritned results normally, within the 5 minutes time limit. So the best first search gave an output on 3 of the 10 times that we launched the program.

Best Choice: **Best First Search [3 – 5]**

**Peg-Puzzle 10**

Name: Diamond

Difficulty: Very Difficult

Number of Pegs: 40

Input file:

9 9

0 0 0 0 1 0 0 0 0

0 0 0 1 1 1 0 0 0

0 0 1 1 2 1 1 0 0

0 1 1 1 1 1 1 1 0

1 1 1 1 1 1 1 1 1

0 1 1 1 1 1 1 1 0

0 0 1 1 1 1 1 0 0

0 0 0 1 1 1 0 0 0

0 0 0 0 1 0 0 0 0

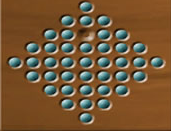


Image 10: Diamond

Board 10: Diamond (Given Outputs)

|  |  |
| --- | --- |
| Output No.1 | Output No.2 |
|  |  |

**Depth First Algorithm**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | FIRST LAUNCH | SECOND LAUNCH | THIRD LAUNCH | FOURTH LAUNCH | FIFTH LAUNCH |
| Time taken | **Out of time** | **Out of time** | **Out of time** | **Out of time** | **Out of time** |
| Given Output | **N/A** | **N/A** | **N/A** | **N/A** | **N/A** |
| Nodes Visited | **N/A** | **N/A** | **N/A** | **N/A** | **N/A** |

**Best First Search Algorithm**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | FIRST LAUNCH | SECOND LAUNCH | THIRD LAUNCH | FOURTH LAUNCH | FIFTH LAUNCH |
| Time taken | **Out of time** | **Out of time** | **Out of time** | **Out of time** | **Out of time** |
| Given Output | **N/A** | **N/A** | **N/A** | **N/A** | **N/A** |
| Nodes Visited | **N/A** | **N/A** | **N/A** | **N/A** | **N/A** |

Comments: The diamond was not solved by neither of our methods. Depth first need to be very lucky, so it will not make that any bad moves at the starting levels of the tree and Best first search needs much more time to reach at the terminal nodes because the number of pegs is 40 here which means that the tree is bigger than in any other peg puzzle.

**CONCLUSIONS**

**EASY PEG SOLITAIRES**

***DEPTH FIRST SEARCH***

|  |  |
| --- | --- |
| EFFICIENCY(Solutions/Launches) | **100%** |
| AVERAGE TIME (milliseconds) | **108.866** |
| AVERAGE NODES VISITED | **1623.2** |
| SPEED(NODES/second) | **14909** |

***BEST FIRST SEARCH***

|  |  |
| --- | --- |
| EFFICIENCY(Solutions/Launches) | 100% |
| AVERAGE TIME (milliseconds) | 179.466 |
| AVERAGE NODES VISITED | 16.13 |
| SPEED(NODES/second) | 89.89 |

**MEDIUM DIF. PEG SOLITAIRES**

***DEPTH FIRST SEARCH***

|  |  |
| --- | --- |
| EFFICIENCY(Solutions/Launches) | **90%** |
| AVERAGE TIME (seconds) | 20.519 |
| AVERAGE NODES VISITED | 686252.7 |
| SPEED(NODES/second) | 33443.34 |
| ACCURACY (CORRECT NODES/TOTAL NODES) | 0.002% |

***BEST FIRST SEARCH***

|  |  |
| --- | --- |
| EFFICIENCY(Solutions/Launches) | 100% |
| AVERAGE TIME (seconds) | 23.148 |
| AVERAGE NODES VISITED | 222.2 |
| SPEED(NODES/second) | 9.59 |
| ACCURACY (CORRECT NODES/TOTAL NODES) | 6% |

**DIFFICULT PEG SOLITAIRES**

***DEPTH FIRST SEARCH***

|  |  |
| --- | --- |
| EFFICIENCY(Solutions/Launches) | **3%** |
| AVERAGE TIME (seconds) | **Out of time** |
| AVERAGE NODES VISITED | **Millions** |
| SPEED(NODES/second) | **Tens of thousands** |
| ACCURACY (CORRECT NODES/TOTAL NODES) | **0%** |

***BEST FIRST SEARCH***

|  |  |
| --- | --- |
| EFFICIENCY(Solutions/Launches) | 20% |
| AVERAGE TIME (seconds) | 46.4 |
| AVERAGE NODES VISITED | 866.4 |
| SPEED(NODES/second) | 18.6 |
| ACCURACY (CORRECT NODES/TOTAL NODES) | **4%** |

The final conclusion is: “As the number of pegs grows, the more we need a best first approach to the problem”. The Depth first algorithm find quickly the solutions when the tree is not very deep. There is also one more case where the the “depth first search” is still a better approach. This is when the peg solitaire is easy to get solved and there are not many early moves that can lead us to multiple dead ends. In all the other cases, the more difficult and complex the problem gets, the chances of the depth first search to get lost are becoming greater and greater. As we see in difficult problems, there is no chance for the depth first search to solve the problem. Even at the medium difficulty problems, the average time of the “Best first search” is very close to those of the “depths first” search. Therefore, we can easily conclude that the creation of a heuristic function helps very much a program to solve those peg solitaire problems.

**DIFFICULTIES AND IMPROVEMENTS**

1. **CREATION OF THE HEURISTIC**

The creation of the evaluation function is not something simple. There had to be created and tested many different factors that could define the final value of every node. Every time a factor was considered as not useful, then the whole computanional study had to start all over again. The changes of the heuristic function were three, so whole launches were three times more than those represented. The heuristic can become even better of course, but this requires for the computanional study to start all over again. When a satisfying level was reached, and the “Best first search” could solve most of the difficult problems, the changes on the heuristic stopped. When “pegs left” and “number of children” were part of the heuristic, the program printed solutions only in one of the difficult problems. So, the improvement here is very clear.

1. **DEPTH FIRST SEARCH**

The depth first algorithm is very well constructed. There could be some spot where the implementation could get a little better but it wont make any difference on the results. The only way for it to find solutions on the difficult problems within five minutes is a stronger Hardware.

1. **BEST FIRST SEARCH & CODE OPTIMIZATION**

The whole creation of the best first search was the hardest part of the program. The finding and the implementation of the heuristic function was difficult, but the part in which the most time was occupied is the importing of that heuristic in the code in the optimized way.  
At the beginning, the calculation of the horizontal and the vertical surface was implemented like that:  
  
  
  
for( i = 0; i<rows;i++)  
 for(j=0;j<columns;j++)  
 if(Board(i,j)=1 & first peg found = false)  
 first peg found=true;  
 first peg = i;

If(Board(i,j)=1)  
 last peg = i;   
  
Vertical distance = last peg – first peg  
  
And to calculate the horizontal distance there was another double for-loop. Therefore the best first search was very slow.   
When this problem was fixed, the best first search had some results more quickly but there has still room for improvement.  
On “Solver” class at the beginning there was an array list which contained all the revealed but unvisited nodes and at every loop the best – valued node had to be found. Hopefully, this was quickly replaced with a hash table which minimizes the fetching time.

George Papoulios, 4/22/2018  
Dai16080