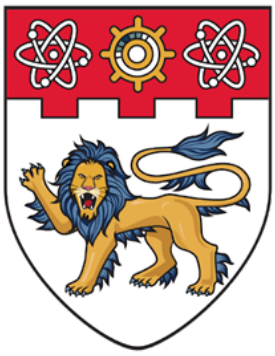

NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

CE 4046 Intelligent Agents

Assignment 1 Report

Agent Decision Making

Mervyn Chiong Jia Rong U1921023K

Content

Task 1: Optimal Policy

Q) Find the optimal policy and the utilities of all the (non-wall) states using both value iteration and policy iteration.

Firstly, I shall display the findings

Map used :

```
map = [['+', '#', '+', ' ', ' ', '+'],
       [' ', '-', ' ', '+', '#', '-'],
       [' ', ' ', '-', ' ', '+', ' '],
       [' ', ' ', ' ', '-', ' ', '+'],
       [' ', '#', '#', '#', '-', ' '],
       [' ', ' ', ' ', ' ', ' ', ' ']
       ]
```

With a starting position of (3,2)

This was done utilizing the max error of 20

1.1.1 Value iteration

number of iterations required: 147
---utility for each state (row, column)---
(0, 0) - utility: 66.907
(0, 1) - utility: 65.584
(0, 2) - utility: 65.966
(0, 3) - utility: 64.935
(0, 4) - utility: 66.042
(0, 5) - utility: 67.313
(1, 0) - utility: 65.540
(1, 1) - utility: 63.641
(1, 2) - utility: 64.747
(1, 3) - utility: 65.070
(1, 4) - utility: 64.979
(1, 5) - utility: 64.977
(2, 0) - utility: 64.280
(2, 1) - utility: 63.118
(2, 2) - utility: 62.656
(2, 3) - utility: 63.925
(2, 4) - utility: 64.925
(2, 5) - utility: 63.982
(3, 0) - utility: 63.060
(3, 1) - utility: 62.096
(3, 2) - utility: 61.652
(3, 3) - utility: 61.834
(3, 4) - utility: 63.664

(3, 5) - utility: 64.273
(4, 0) - utility: 61.880
(4, 1) - utility: 61.116
(4, 2) - utility: 60.705
(4, 3) - utility: 60.868
(4, 4) - utility: 61.483
(4, 5) - utility: 62.986
(5, 0) - utility: 60.730
(5, 1) - utility: 60.088
(5, 2) - utility: 59.716
(5, 3) - utility: 59.897
(5, 4) - utility: 60.792
(5, 5) - utility: 61.778
---optimal policy grid (# = wall)---
End result:

Fig 1.1.1 Outcome post Value iteration [Random max error value]

1.1.2 Value Iteration [tweaked]:

I then tweaked the max error to be 1.4 from the previous 20
number of iterations required: 425
---utility for each state (row, column)---
(0, 0) - utility: 98.590
(0, 2) - utility: 93.635
(0, 3) - utility: 92.465
(0, 4) - utility: 91.244
(0, 5) - utility: 91.918
(1, 0) - utility: 96.983
(1, 1) - utility: 94.473
(1, 2) - utility: 93.135
(1, 3) - utility: 92.987
(1, 5) - utility: 89.508
(2, 0) - utility: 95.538
(2, 1) - utility: 94.176
(2, 2) - utility: 91.884
(2, 3) - utility: 91.766

(2, 4) - utility: 91.692
(2, 5) - utility: 90.385
(3, 0) - utility: 94.144
(3, 1) - utility: 93.042
(3, 2) - utility: 91.822
(3, 3) - utility: 89.705
(3, 4) - utility: 90.404
(3, 5) - utility: 90.478
(4, 0) - utility: 92.902
(4, 4) - utility: 88.138
(4, 5) - utility: 89.156
(5, 0) - utility: 91.527
(5, 1) - utility: 90.318
(5, 2) - utility: 89.125
(5, 3) - utility: 87.946
(5, 4) - utility: 87.159
(5, 5) - utility: 87.887
---optimal policy grid (# = wall)---
End result:

Fig 1.1.2 Outcome post Value iteration [Optimal max error value]

Noticeably the utility values are much higher as a lower error tolerance is allowed, allowing it to reach a higher utility score as the convergence threshold is increased.

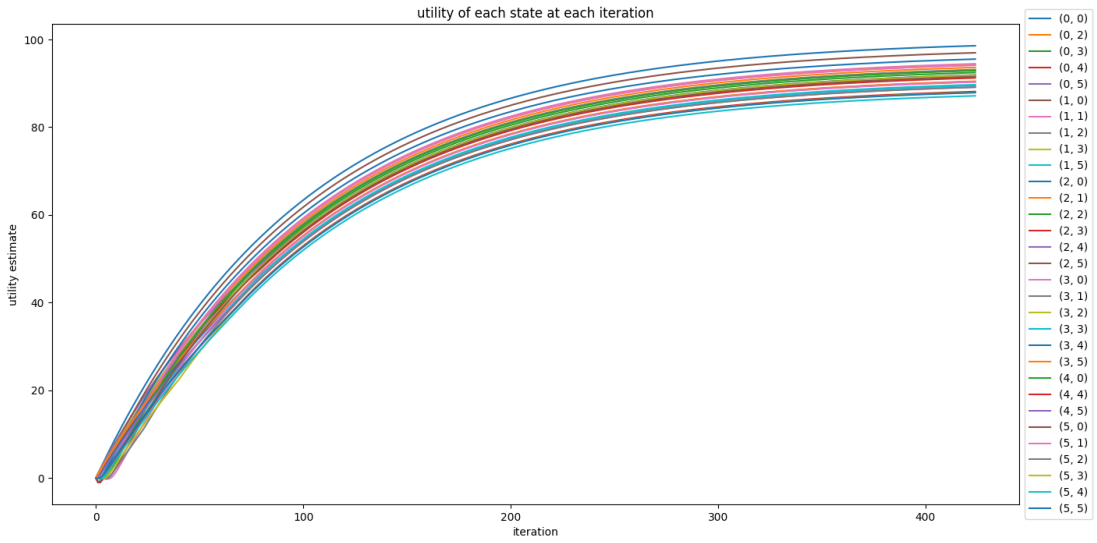

Fig 1.1.3 Graph of Iteration vs Utility scores for all squares [Value Iteration]

1.2.1 Policy iteration:

number of iterations required: 200
---utility for each state (row, column)---
(0, 0) - utility: 23.959
(0, 2) - utility: 22.328
(0, 3) - utility: 21.288
(0, 4) - utility: 20.950
(0, 5) - utility: 20.574
(1, 0) - utility: 16.460
(1, 1) - utility: 14.553
(1, 2) - utility: 14.665
(1, 3) - utility: 15.703
(1, 5) - utility: 9.726
(2, 0) - utility: 11.210
(2, 1) - utility: 11.043
(2, 2) - utility: 9.970
(2, 3) - utility: 10.084
(2, 4) - utility: 10.701
(2, 5) - utility: 10.143
(3, 0) - utility: 7.678
(3, 1) - utility: 7.898

(3, 2) - utility: 7.974
(3, 3) - utility: 6.987
(3, 4) - utility: 6.794
(3, 5) - utility: 7.709
(4, 0) - utility: 5.504
(4, 4) - utility: 2.607
(4, 5) - utility: 3.451
(5, 0) - utility: 4.470
(5, 1) - utility: 4.343
(5, 2) - utility: 4.217
(5, 3) - utility: 4.092
(5, 4) - utility: 3.820
(5, 5) - utility: 3.673
---optimal policy grid (# = wall)---
End result:

Fig 1.2.1 Outcome post Policy iteration

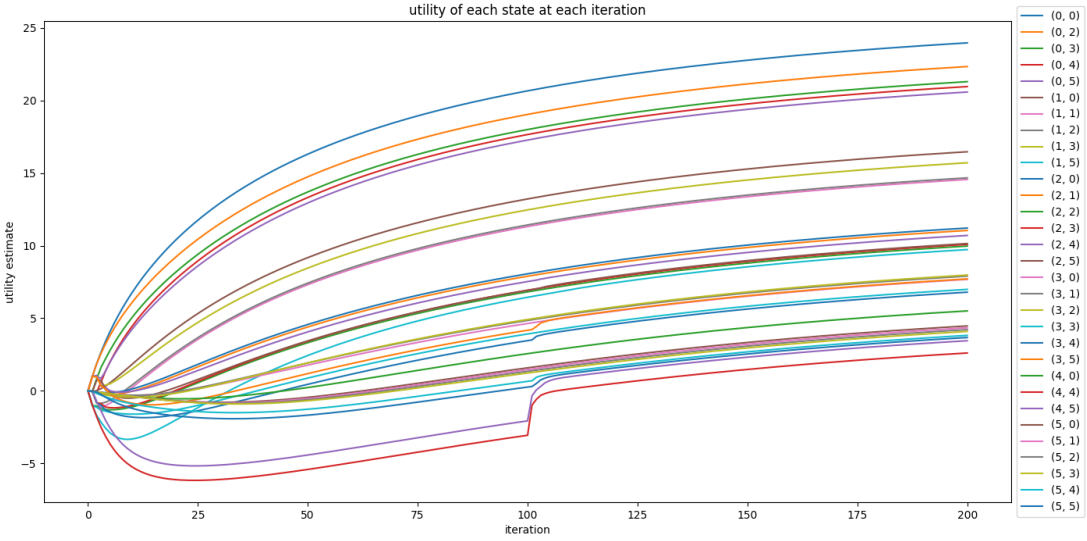

Fig 1.2.2 Graph of Iteration vs Utility scores for all squares [Policy Iteration]

It is quite evident that Policy iteration performs much better as compared to value iteration. Value iteration took 425 iterations but gave higher utility scores, when compared to policy

iteration which took only 200 iterations but with a much lower utility score. Optimally speaking, policy iteration was able to compute the results much quicker at the cost of utility values.

*Source code is in the zip file
The answer for Q1 will be at the conclusion because I believe to find the optimal policy we require more criterias aside from iteration count to decide it.

Part 2 Complicated Maze Exploration

Q How does the number of states and the complexity of the environment affect convergence?
Let us first show what needs to be changed to increase complexity. This will be showed in the explanations of the graphs and the theories we create

Q How complex can you make the environment and still be able to learn the right policy?
For now we impose certain criterias that our maze should have and from these criterias the optimal policy should be the one with the least amount of iterations while still presenting an optimal solution.

2.1 Defining Complicated Maze

Starting positing is (9,0) [bottom left]
The created maze looks like this

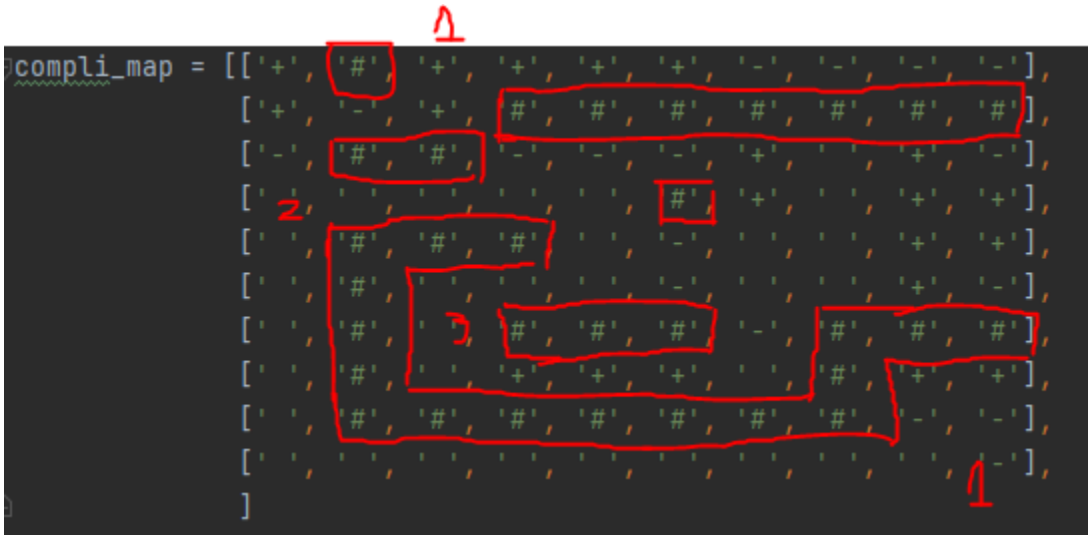

Fig 2.1 User created complicated maze

Complexity if [10x10] maze is based on the following reasons

1) Number of forks leading into dead ends or loops
    a) In this map, I have made 2 dead ends and also a loop in the middle, naturally a dead end is represented by negative reward values, evident in the top and bottom of the right side
2) Number of forks along the correct path
    a) I also implemented several forking paths from the user which ties into dead ends. In this case there is no actual “correct path”. The closest to a correct path would be to arrive at the middle of the right side
3) Length of incorrect paths
    a) Similar to dead ends just that they are meant to waste time or provide an incentive that is not worth it. As such i've created a loop in the middle, that simply is there to waste time with very mild incentive compared to going to the right side

4) Size of the maze
    a) The maze here is a 10x10 maze as such this will be explored using a different later on in the report

I will now only show the values for the tweaked optimal results

2.2 Value Iteration

number of iterations required: 425
---utility for each state (row, column)---
(0, 0) - utility: 98.590
(0, 2) - utility: 98.590
(0, 3) - utility: 98.590
(0, 4) - utility: 98.590
(0, 5) - utility: 98.590
(0, 6) - utility: 96.096
(0, 7) - utility: 93.633
(0, 8) - utility: 91.201
(0, 9) - utility: 88.800
(1, 0) - utility: 98.316
(1, 1) - utility: 96.096
(1, 2) - utility: 98.590
(2, 0) - utility: 95.825
(2, 3) - utility: 89.287
(2, 4) - utility: 91.535
(2, 5) - utility: 94.093
(2, 6) - utility: 96.561
(2, 7) - utility: 96.519
(2, 8) - utility: 97.769
(2, 9) - utility: 95.925
(3, 0) - utility: 94.409
(3, 1) - utility: 93.165
(3, 2) - utility: 91.936
(3, 3) - utility: 90.564
(3, 4) - utility: 90.559
(3, 6) - utility: 96.691
(3, 7) - utility: 96.817
(3, 8) - utility: 98.145
(3, 9) - utility: 98.186
(4, 0) - utility: 93.165
(4, 4) - utility: 91.770
(4, 5) - utility: 93.127
(4, 6) - utility: 95.624

(4, 7) - utility: 96.817
(4, 8) - utility: 98.145
(4, 9) - utility: 98.186
(5, 0) - utility: 91.936
(5, 2) - utility: 94.440
(5, 3) - utility: 93.195
(5, 4) - utility: 91.944
(5, 5) - utility: 92.801
(5, 6) - utility: 95.213
(5, 7) - utility: 96.519
(5, 8) - utility: 97.769
(5, 9) - utility: 95.925
(6, 0) - utility: 90.722
(6, 2) - utility: 95.856
(6, 6) - utility: 94.514
(7, 0) - utility: 89.523
(7, 2) - utility: 97.135
(7, 3) - utility: 98.590
(7, 4) - utility: 98.590
(7, 5) - utility: 98.590
(7, 6) - utility: 96.988
(7, 8) - utility: 98.590
(7, 9) - utility: 98.590
(8, 0) - utility: 88.339
(8, 8) - utility: 96.096
(8, 9) - utility: 96.096
(9, 0) - utility: 87.060
(9, 1) - utility: 86.166
(9, 2) - utility: 87.322
(9, 3) - utility: 88.493
(9, 4) - utility: 89.679
(9, 5) - utility: 90.879
(9, 6) - utility: 92.095
(9, 7) - utility: 93.326
(9, 8) - utility: 94.573
(9, 9) - utility: 93.736
---optimal policy grid (# = wall)---

End Result:

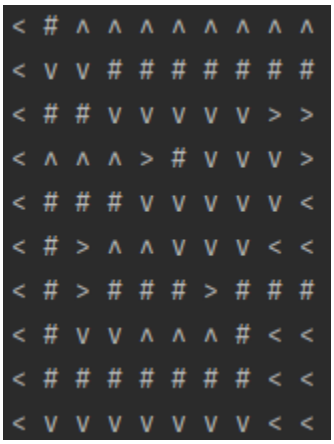

Fig 2.2.1 Outcome post Value iteration [Optimal max error value]

Value iteration Shows no increase in total number of iterations required. Showing that despite an increase in complexity the run time it took to process may only be slightly higher than the original 5x5 maze.

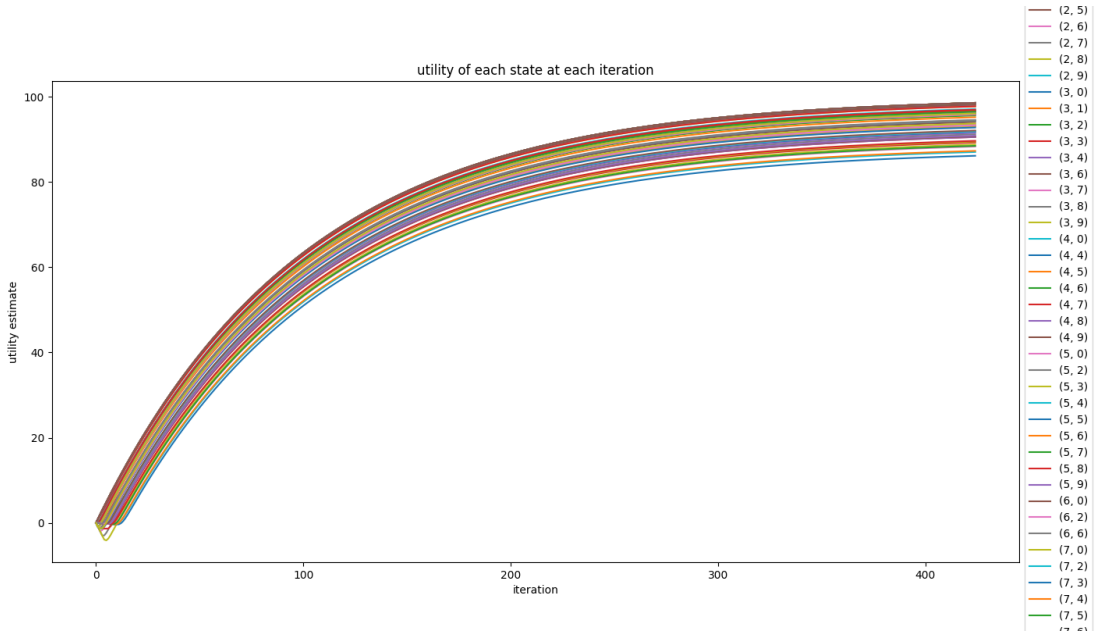

Fig 2.2.2 Graph of Iteration vs Utility scores for all squares [Value Iteration]

Utility values from Fig 2.2.2 are similar to Fig 1.1.3 in which they all have a rather high utility score. From this it is rather safe to conclude that there is no effect on value iteration regarding its convergence. We will check with a 100x100 maze to finalize this.

2.3.1 Policy iteration

number of iterations required: 600

---utility for each state (row, column)---
(0, 0) - utility: 31.035
(0, 2) - utility: 36.084
(0, 3) - utility: 48.191
(0, 4) - utility: 55.066
(0, 5) - utility: 57.404
(0, 6) - utility: 55.441
(0, 7) - utility: 53.502
(0, 8) - utility: 51.588
(0, 9) - utility: 49.698
(1, 0) - utility: 24.071
(1, 1) - utility: 22.523
(1, 2) - utility: 29.630
(2, 0) - utility: 9.438
(2, 3) - utility: 28.718
(2, 4) - utility: 39.096
(2, 5) - utility: 38.240
(2, 6) - utility: 51.386
(2, 7) - utility: 50.886
(2, 8) - utility: 52.455
(2, 9) - utility: 50.928
(3, 0) - utility: 5.862
(3, 1) - utility: 11.734
(3, 2) - utility: 19.208
(3, 3) - utility: 29.050
(3, 4) - utility: 42.262
(3, 6) - utility: 52.482
(3, 7) - utility: 51.325
(3, 8) - utility: 52.247
(3, 9) - utility: 52.647
(4, 0) - utility: 3.284
(4, 4) - utility: 44.502
(4, 5) - utility: 42.637
(4, 6) - utility: 48.818
(4, 7) - utility: 48.137
(4, 8) - utility: 51.714
(4, 9) - utility: 52.104
(5, 0) - utility: 1.443
(5, 2) - utility: 51.920
(5, 3) - utility: 46.087
(5, 4) - utility: 45.353
(5, 5) - utility: 43.437
(5, 6) - utility: 50.348
(5, 7) - utility: 48.268

(5, 8) - utility: 51.507
(5, 9) - utility: 49.887
(6, 0) - utility: 0.154
(6, 2) - utility: 53.360
(6, 6) - utility: 52.163
(7, 0) - utility: -0.713
(7, 2) - utility: 54.089
(7, 3) - utility: 59.995
(7, 4) - utility: 61.900
(7, 5) - utility: 59.995
(7, 6) - utility: 54.089
(7, 8) - utility: 7.093
(7, 9) - utility: 7.080
(8, 0) - utility: -1.247
(8, 8) - utility: -2.290
(8, 9) - utility: -2.406
(9, 0) - utility: -1.501
(9, 1) - utility: -1.533
(9, 2) - utility: -1.564
(9, 3) - utility: -1.594
(9, 4) - utility: -1.624
(9, 5) - utility: -1.654
(9, 6) - utility: -1.684
(9, 7) - utility: -1.713
(9, 8) - utility: -1.802
(9, 9) - utility: -2.958
---optimal policy grid (# = wall)---
End result:

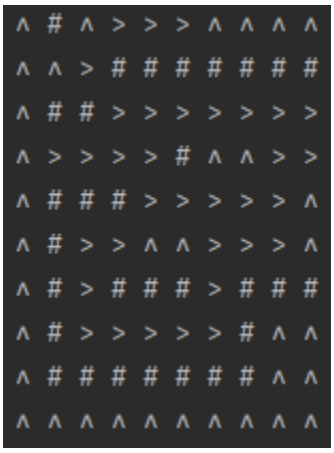

Fig 2.3.1 Outcome post Policy iteration

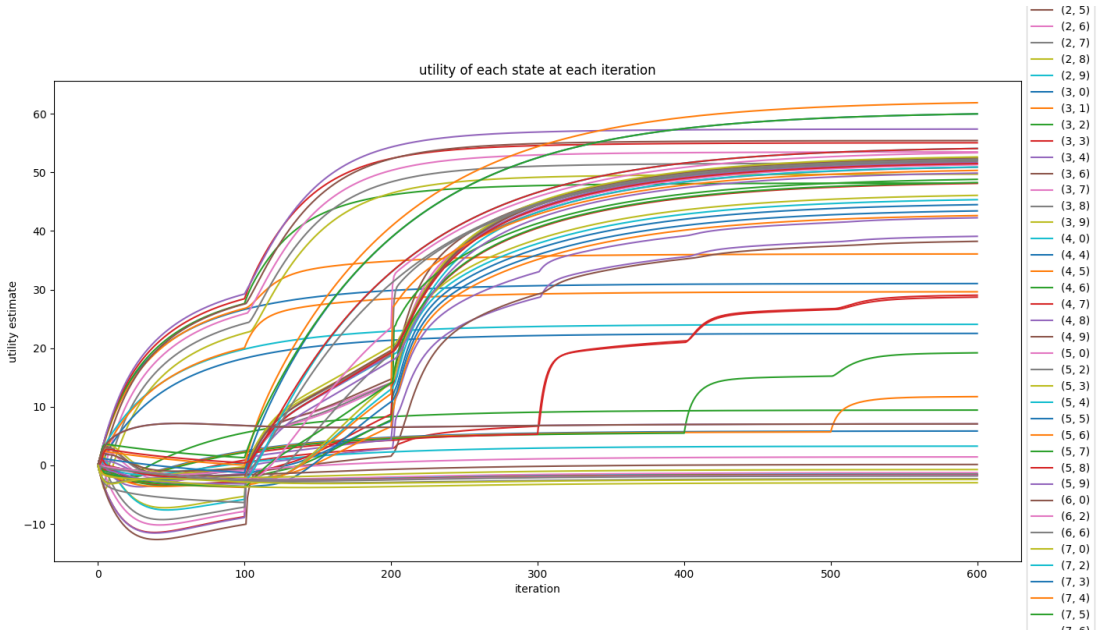

Fig 2.3.2 Graph of Iteration vs Utility scores for all squares [Policy Iteration]

From these 2 outcomes we can conclude certain findings:

- For Policy iteration, The change in number of iterations is as such :5x5 maze [200] vs 10x10 [600]. We can hypothesize that potentially policy iteration is more susceptible to the increase in both size and complexity when compared to value iteration.
- Furthermore, the dead ends produced negative utility scores. As a matter of fact, the whole row[row 9], so the bottom most row, all contained negative utility scores. Which is a stark contrast to utility values of the 5x5 maze result.

As such, I concur that Policy iteration's convergence is highly susceptible to minute changes in the environment and increase in complexity of the maze.

2.4.1 Maze Exploration Effects with Size

Now to explore the effects of a larger maze on its complexity.

Because we are using a 100x100 the end result will be rather big it is stored in a txt file and as such i will only show the graphs
From rigorous testing we can conclude that the max_error value greatly changes the iteration counts of the policy iteration. As such in this case, we used an arbitrary number 20 as the max error as it would take too long to find out the optimal number for a 100x100 maze. In order to find a more optimal value, more trial and error testing is required which is not feasible for large mazes such as this.

Value iteration:
[bonus_value_iteration_result.txt]
Iterations : 161

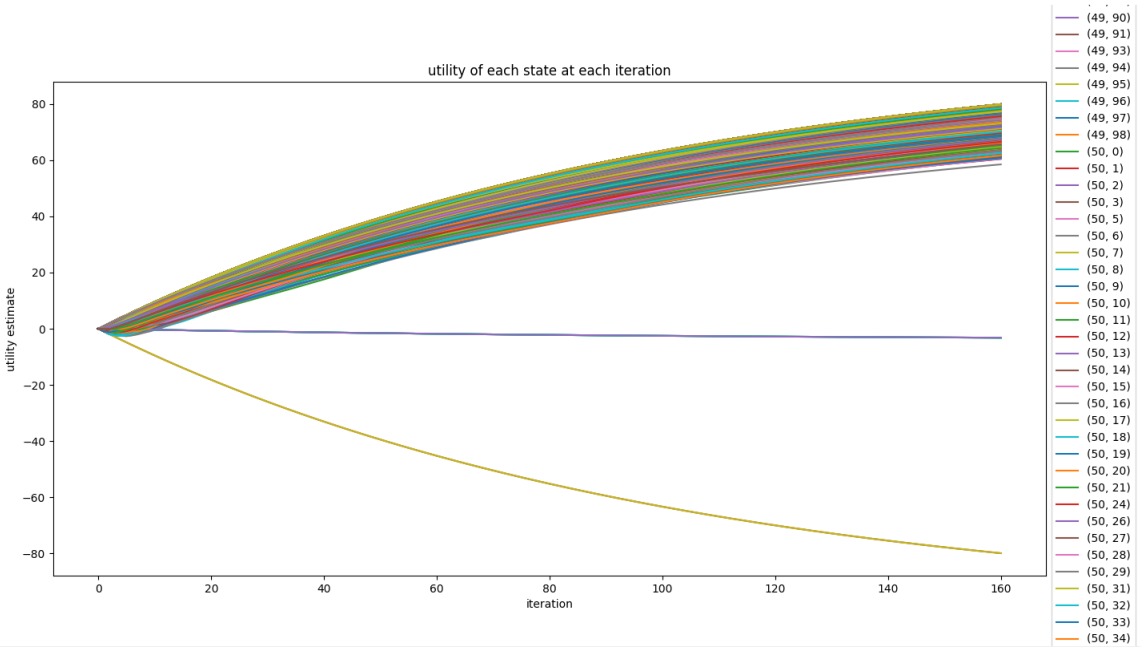

Fig 2.4.1 Graph of Iteration vs Utility scores for all squares [Value Iteration]

Policy iteration[NUM_POLICY_EVALUATION remained unchanged at 100]:
Most optimal value for testing as increasing the value greatly increases computation time as iteration count only goes higher.

[bonus_policy_iteration_result.txt]
Iterations :
800

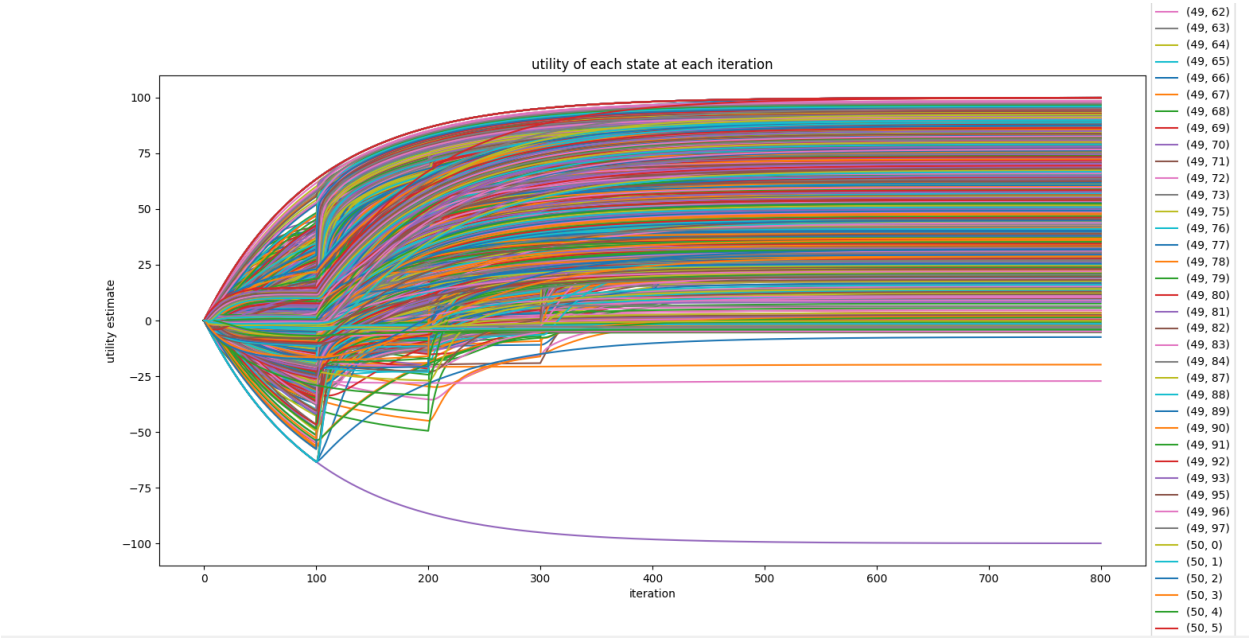

Fig 2.4.2 Graph of Iteration vs Utility scores for all squares [Policy Iteration]

Observations:

- Policy iteration managed to reach 800 iterations before fully converging. After more detailed testing with the NUM_POLICY_EVALUATION value where I tweaked it to 600 and got an iteration count of 6000. I can fairly say that the iterations required for convergence for the policy iteration is extremely dependent on the value and map you choose. Since we are only looking at how size affects the complexity and iterations. It is safe to conclude that yes there is a direct relationship with the size of the map and the iteration count.
- Aside from this, another hypothesis regarding the convergent point, the point at which all squares converge, is also confirmed. From Fig 2.4.2 and Fig 2.3.2 we can see that there are multiple states that actually converge at different iteration values. As such it is no wonder that policy iteration will have a much higher iteration count when compared to value iteration. This would also mean that it is more susceptible to an increase in complexity. Which is evident in Fig 2.4.2
    - This ties in with our earlier theory that policy iteration's convergence is dependent on complexity and size thereby confirming it.
- Shockingly for value iteration, there was an actual decrease in the number of iterations required. For the 100x100 matrix it only required 161 iterations for full convergence. As such, we can say that the complexity of a maze increases the number of iterations required for convergence. However, if the maze is large and of a lower complexity level, the number of convergences required is much lesser.

Conclusion

From the results it is quite evident to me that Value iteration in this particular scenario performs much better in most cases. Not only in iteration count but just in general when handling a larger scale maze. It is even able to reduce the number of iterations required which on the flip side, Policy iteration actually increased in number of interactions required. I do know that in most cases policy is better, but for this specific case. I believe that Value is far superior