Project 3 MDP

Xuzheng Lu
G34363475

GitHub: https://github.com/LeanderLXZ/mdp

1. Files description

```
mdp/
├── doc.docx                    // document file
├── README.md                   // readme file
├── input/                      // input folder
│   ├── i1.txt                  // TXT files of inputs of different boards
│   │   ⋮
│   └── i8.txt
├── results/                    // results folder
│   ├── results_1.csv           // CSV files of results
│   ├── results_2.csv
│   └── results_3.csv
└── src/                        // srouce code folder
    ├── value_iteration.py      // value iteration
    ├── policy_iteration.py     // policy iteration
    ├── policy_iteration_linear.py  // policy iteration based on linear equations solving
    ├── notebook.ipynb          // jupyter notebook for experiments
    └── utils.py                // utilities
```

2. Arguments for algorithms

board_file_path	string, the path of input file
threshold	float, default=0.01, the threshold for stop the iteration
init_policy_direction	int, default=None, the index of the chosen direction in ['up', 'right', 'down', 'left'] for initializing the policy. If None, randomly assign directions to the initial policy.
improve_p_with_v	Boolean, default=Flase, set True to improve the values while improving the policy
use_arrow	Boolean, default=False, set True to use arrows for display
verbose	Boolean, default=False, set True to display extra information

3. Run the code

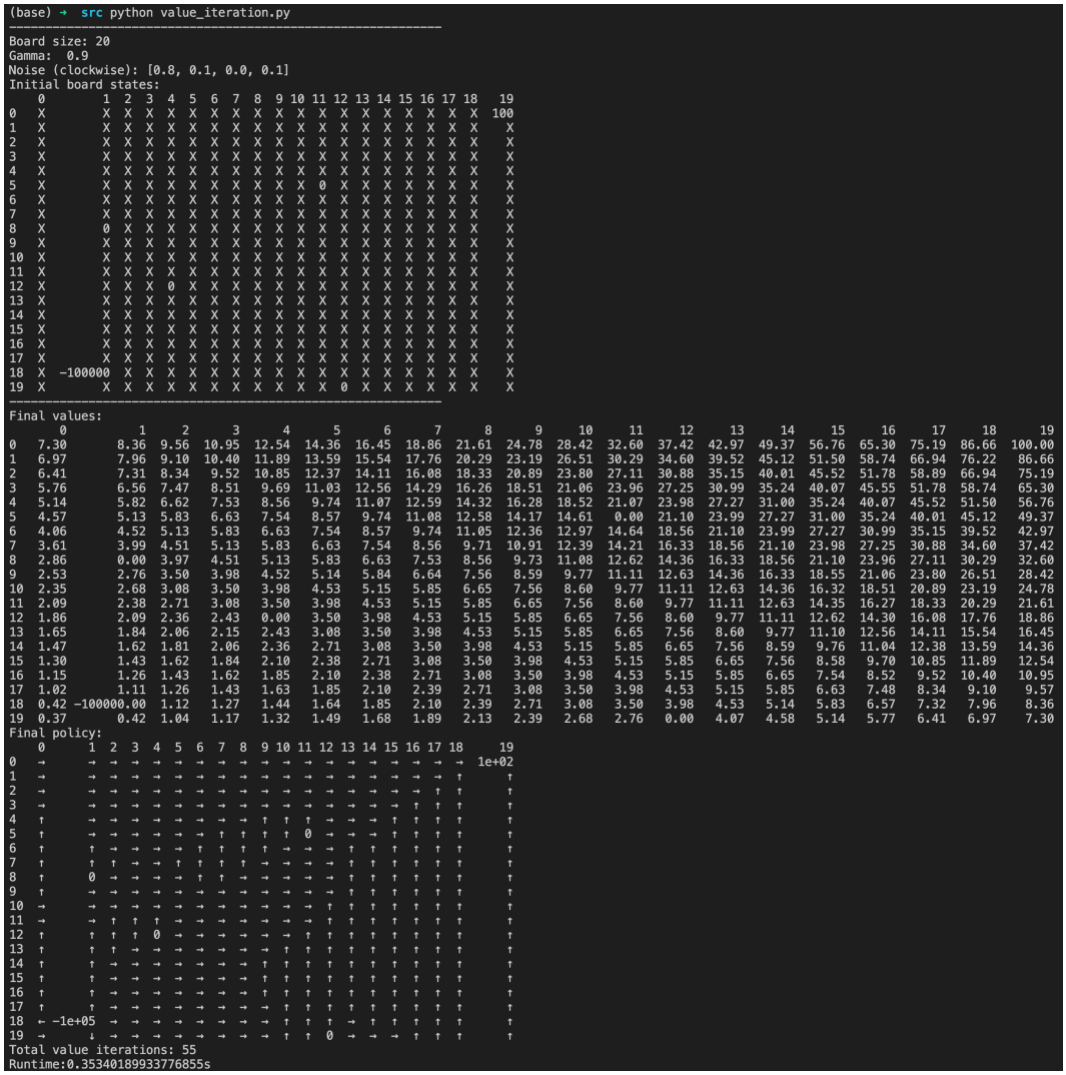
1) Run value iteration
Command:

```
python value_iteration.py
```

You can set the configuration of the algorithm ` value_iteration.py `:

```
_ = ValueIteration('../input/i7.txt', threshold=0.01,
                   use_arrow=True, verbose=False).run()
```

Output example:



2) Run policy iteration

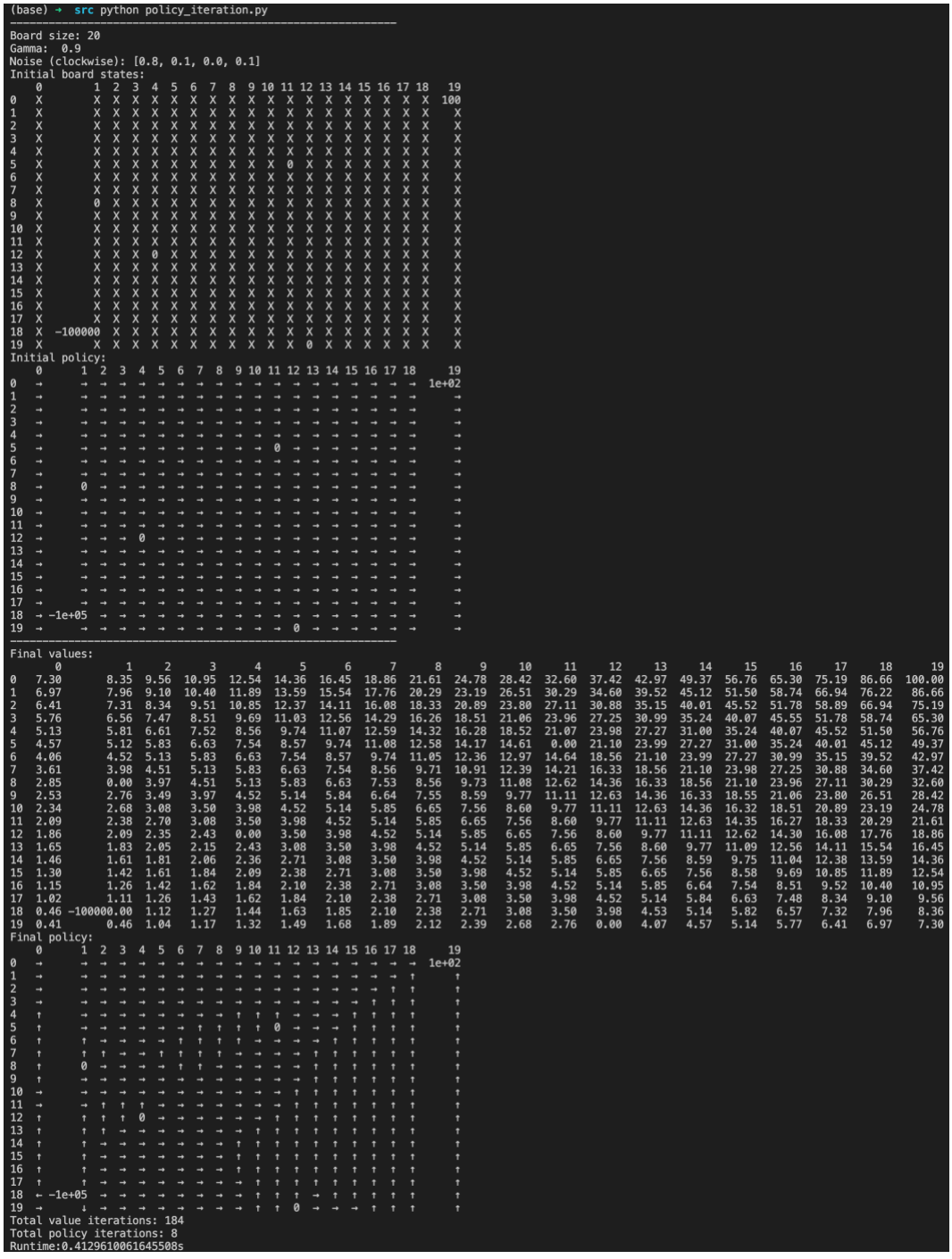
Command:

python policy_iteration.py

You can set the configuration of the algorithm in ` policy_iteration.py `:

```
_ = PolicyIteration('../input/i7.txt', threshold=0.01,
                    init_policy_direction=1, improve_p_with_v=True,
                    use_arrow=True, verbose=False).run()
```

Output example:



```
(base) ➜  src python policy_iteration.py
------------------------------------------------------------
Board size: 20
Gamma:  0.9
Noise (clockwise): [0.8, 0.1, 0.0, 0.1]
Initial board states:
    0        1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18   19
0   X        X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  100
1   X        X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X    X
2   X        X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X    X
3   X        X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X    X
4   X        X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X    X
5   X        X  X  X  X  X  X  X  X  X  X  0  X  X  X  X  X  X  X    X
6   X        X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X    X
7   X        X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X    X
8   X        0  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X    X
9   X        X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X    X
10  X        X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X    X
11  X        X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X    X
12  X        X  X  X  0  X  X  X  X  X  X  X  X  X  X  X  X  X  X    X
13  X        X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X    X
14  X        X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X    X
15  X        X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X    X
16  X        X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X    X
17  X        X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X    X
18  X  -100000  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X  X    X
19  X        X  X  X  X  X  X  X  X  X  X  0  X  X  X  X  X  X  X    X
Initial policy:
    0      1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18     19
0   →      →  →  →  →  →  →  →  →  →  →  →  →  →  →  →  →  →  →  1e+02
1   →      →  →  →  →  →  →  →  →  →  →  →  →  →  →  →  →  →      →
2   →      →  →  →  →  →  →  →  →  →  →  →  →  →  →  →  →  →      →
3   →      →  →  →  →  →  →  →  →  →  →  →  →  →  →  →  →  →      →
4   →      →  →  →  →  →  →  →  →  →  →  →  →  →  →  →  →  →      →
5   →      →  →  →  →  →  →  →  →  →  →  0  →  →  →  →  →  →      →
6   →      →  →  →  →  →  →  →  →  →  →  →  →  →  →  →  →  →      →
7   →      →  →  →  →  →  →  →  →  →  →  →  →  →  →  →  →  →      →
8   →      0  →  →  →  →  →  →  →  →  →  →  →  →  →  →  →  →      →
9   →      →  →  →  →  →  →  →  →  →  →  →  →  →  →  →  →  →      →
10  →      →  →  →  →  →  →  →  →  →  →  →  →  →  →  →  →  →      →
11  →      →  →  →  →  →  →  →  →  →  →  →  →  →  →  →  →  →      →
12  →      →  →  →  0  →  →  →  →  →  →  →  →  →  →  →  →  →      →
13  →      →  →  →  →  →  →  →  →  →  →  →  →  →  →  →  →  →      →
14  →      →  →  →  →  →  →  →  →  →  →  →  →  →  →  →  →  →      →
15  →      →  →  →  →  →  →  →  →  →  →  →  →  →  →  →  →  →      →
16  →      →  →  →  →  →  →  →  →  →  →  →  →  →  →  →  →  →      →
17  →      →  →  →  →  →  →  →  →  →  →  →  →  →  →  →  →  →      →
18  →  -1e+05  →  →  →  →  →  →  →  →  →  →  →  →  →  →  →  →      →
19  →      →  →  →  →  →  →  →  →  →  →  0  →  →  →  →  →  →      →
------------------------------------------------------------
Final values:
     0        1     2     3     4     5     6     7     8     9    10    11    12    13    14    15    16    17    18     19
0   7.30     8.35  9.56  10.95 12.54 14.36 16.45 18.86 21.61 24.78 28.42 32.60 37.42 42.97 49.37 56.76 65.30 75.19 86.66 100.00
1   6.97     7.96  9.10  10.40 11.89 13.59 15.54 17.76 20.29 23.19 26.51 30.29 34.60 39.52 45.12 51.50 58.74 66.94 76.22  86.66
2   6.41     7.31  8.34   9.51 10.85 12.37 14.11 16.08 18.33 20.89 23.80 27.11 30.88 35.15 40.01 45.52 51.78 58.89 66.94  75.19
3   5.76     6.56  7.47   8.51  9.69 11.03 12.56 14.29 16.26 18.51 21.06 23.96 27.25 30.99 35.24 40.07 45.55 51.78 58.74  65.30
4   5.13     5.81  6.61   7.52  8.56  9.74 11.07 12.59 14.32 16.28 18.52 21.07 23.98 27.27 31.00 35.24 40.07 45.52 51.50  56.76
5   4.57     5.12  5.83   6.63  7.54  8.57  9.74 11.08 12.58 14.17 14.61  0.00 21.10 23.99 27.27 31.00 35.24 40.01 45.12  49.37
6   4.06     4.52  5.13   5.83  6.63  7.54  8.57  9.74 11.05 12.36 12.97 14.64 18.56 21.10 23.99 27.27 30.99 35.15 39.52  42.97
7   3.61     3.98  4.51   5.13  5.83  6.63  7.54  8.56  9.71 10.91 12.39 14.21 16.33 18.56 21.10 23.98 27.25 30.88 34.60  37.42
8   2.85     0.00  3.97   4.51  5.13  5.83  6.63  7.53  8.56  9.73 11.08 12.62 14.36 16.33 18.56 21.10 23.96 27.11 30.29  32.60
9   2.53     2.76  3.49   3.97  4.52  5.14  5.84  6.64  7.55  8.59  9.77 11.11 12.63 14.36 16.33 18.55 21.06 23.80 26.51  28.42
10  2.34     2.68  3.08   3.50  3.98  4.52  5.14  5.85  6.65  7.56  8.60  9.77 11.11 12.63 14.36 16.32 18.51 20.89 23.19  24.78
11  2.09     2.38  2.70   3.08  3.50  3.98  4.52  5.14  5.85  6.65  7.56  8.60  9.77 11.11 12.63 14.35 16.27 18.33 20.29  21.61
12  1.86     2.09  2.35   2.43  0.00  3.50  3.98  4.52  5.14  5.85  6.65  7.56  8.60  9.77 11.11 12.62 14.30 16.08 17.76  18.86
13  1.65     1.83  2.05   2.15  2.43  3.08  3.50  3.98  4.52  5.14  5.85  6.65  7.56  8.60  9.77 11.09 12.56 14.11 15.54  16.45
14  1.46     1.61  1.81   2.06  2.36  2.71  3.08  3.50  3.98  4.52  5.14  5.85  6.65  7.56  8.59  9.75 11.04 12.38 13.59  14.36
15  1.30     1.42  1.61   1.84  2.09  2.38  2.71  3.08  3.50  3.98  4.52  5.14  5.85  6.65  7.56  8.58  9.69 10.85 11.89  12.54
16  1.15     1.26  1.42   1.62  1.84  2.10  2.38  2.71  3.08  3.50  3.98  4.52  5.14  5.85  6.64  7.54  8.51  9.52 10.40  10.95
17  1.02     1.11  1.26   1.43  1.62  1.84  2.10  2.38  2.71  3.08  3.50  3.98  4.52  5.14  5.84  6.63  7.48  8.34  9.10   9.56
18  0.46 -100000.00  1.12   1.27  1.44  1.63  1.85  2.10  2.38  2.71  3.08  3.50  3.98  4.53  5.14  5.82  6.57  7.32  7.96   8.36
19  0.41     0.46  1.04   1.17  1.32  1.49  1.68  1.89  2.12  2.39  2.68  2.76  0.00  4.07  4.57  5.14  5.77  6.41  6.97   7.30
Final policy:
    0      1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18     19
0   →      →  →  →  →  →  →  →  →  →  →  →  →  →  →  →  →  →  →  1e+02
1   →      →  →  →  →  →  →  →  →  →  →  →  →  →  →  →  →  →  ↑      ↑
2   →      →  →  →  →  →  →  →  →  →  →  →  →  →  →  →  ↑  ↑      ↑
3   →      →  →  →  →  →  →  →  →  →  →  →  →  →  →  ↑  ↑  ↑      ↑
4   ↑      →  →  →  →  →  →  →  →  ↑  ↑  ↑  →  →  →  ↑  ↑  ↑  ↑      ↑
5   ↑      →  →  →  →  →  →  ↑  ↑  ↑  ↑  0  →  →  →  ↑  ↑  ↑  ↑      ↑
6   ↑      ↑  →  →  →  →  ↑  ↑  ↑  ↑  →  →  →  →  ↑  ↑  ↑  ↑  ↑      ↑
7   ↑      ↑  ↑  →  →  ↑  ↑  ↑  ↑  →  →  →  →  →  ↑  ↑  ↑  ↑  ↑      ↑
8   ↑      0  →  →  →  →  ↑  ↑  →  →  →  →  →  ↑  ↑  ↑  ↑  ↑  ↑      ↑
9   ↑      →  →  →  →  →  →  →  →  →  →  →  →  →  ↑  ↑  ↑  ↑  ↑      ↑
10  →      →  →  →  →  →  →  →  →  →  →  →  →  →  ↑  ↑  ↑  ↑  ↑      ↑
11  →      →  ↑  ↑  ↑  →  →  →  →  →  →  →  →  ↑  ↑  ↑  ↑  ↑  ↑      ↑
12  ↑      ↑  ↑  ↑  0  →  →  →  →  →  →  ↑  ↑  ↑  ↑  ↑  ↑  ↑  ↑      ↑
13  ↑      ↑  ↑  →  →  →  →  →  →  →  ↑  ↑  ↑  ↑  ↑  ↑  ↑  ↑  ↑      ↑
14  ↑      ↑  →  →  →  →  →  →  →  ↑  ↑  ↑  ↑  ↑  ↑  ↑  ↑  ↑  ↑      ↑
15  ↑      ↑  →  →  →  →  →  →  →  ↑  ↑  ↑  ↑  ↑  ↑  ↑  ↑  ↑  ↑      ↑
16  ↑      ↑  →  →  →  →  →  →  →  ↑  ↑  ↑  ↑  ↑  ↑  ↑  ↑  ↑  ↑      ↑
17  ↑      ↑  →  →  →  →  →  →  →  →  ↑  ↑  ↑  ↑  ↑  ↑  ↑  ↑  ↑      ↑
18  ←  -1e+05  →  →  →  →  →  →  →  →  ↑  ↑  ↑  →  ↑  ↑  ↑  ↑  ↑      ↑
19  →      ↓  →  →  →  →  →  →  →  →  ↑  ↑  0  →  →  →  ↑  ↑  ↑      ↑
Total value iterations: 184
Total policy iterations: 8
Runtime:0.4129610061645508s
```

3) Run policy iteration based on linear equations solving

Command:

```
python policy_iteration_linear.py
```

You can set the configuration of the algorithm in ` policy_iteration.py `:

```
_ = PolicyIterationLinear('../inputs/i3.txt', threshold=0.01,
                          init_policy_direction=1, improve_p_with_v=True,
                          use_arrow=True, verbose=True).run()
```

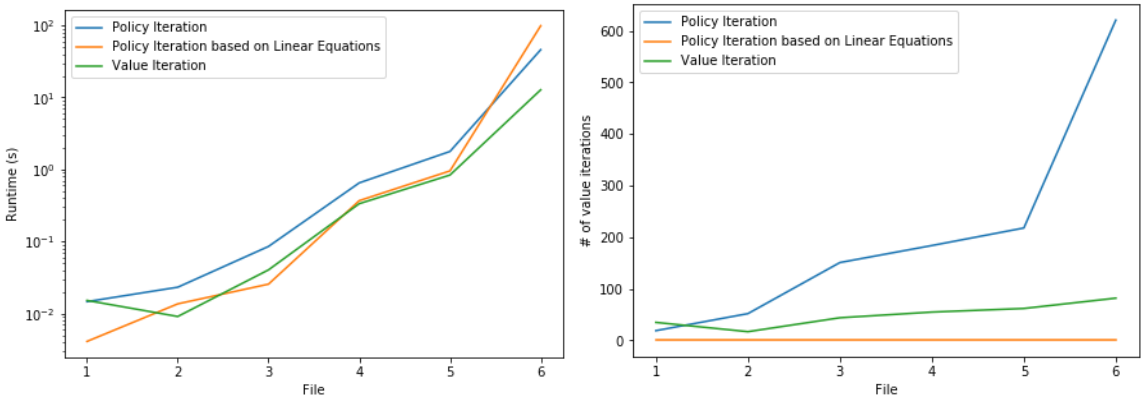
Output example:

```
(base) ➜  src git:(master) ✗ python policy_iteration_linear.py
------------------------------------------------------------
Board size: 10
Gamma:  0.9
Noise (clockwise): [0.8, 0.1, 0.0, 0.1]
Initial board states:
         0       1    2       3       4       5       6       7       8       9
0  -1000.0       X    X       X  -100.0  -100.0       X       X       X       X
1         X     0.0    X     0.0       X       X     0.0     0.0       X  1000.0
2       0.0  -100.0    X       X       X       X       X       X       X       X
3       0.0       X    X       X  1000.0       X  -10000.0       X  -10000.0       X
4         X       X    X       X       X       X    -100.0  -1000.0     0.0       X
5       0.0       X    X   -100.0       X  -100.0       X       X       X  -100.0
6  -1000.0  -100.0    X  -1000.0       X     0.0       X       X     0.0     0.0
7         X       X    X       X       X  -100.0    1000.0     0.0       X       X
8         X       X  0.0       X     0.0       X       X       X       X       X
9         X       X  0.0       X       X       X       X       X     0.0       X
Initial policy:
       0      1  2      3      4      5      6      7      8      9
0 -1e+03      →  →      → -1e+02 -1e+02      →      →      →      →
1      →      0  →      0      →      →      0      0      →  1e+03
2      0 -1e+02  →      →      →      →      →      →      →      →
3      0      →  →      →  1e+03      → -1e+04      → -1e+04      →
4      →      →  →      →      →      → -1e+02 -1e+03      0      →
5      0      →  → -1e+02      → -1e+02      →      →      → -1e+02
6 -1e+03 -1e+02  → -1e+03      →      0      →      →      0      0
7      →      →  →      →      → -1e+02  1e+03      0      →      →
8      →      →  0      →      0      →      →      →      →      →
9      →      →  0      →      →      →      →      →      0      →
------------------------------------------------------------
Policy iteration: 0
Solving linear equations...
------------------------------------------------------------
Policy iteration: 1
Solving linear equations...
------------------------------------------------------------
Policy iteration: 2
Solving linear equations...
------------------------------------------------------------
Policy iteration: 3
Solving linear equations...
------------------------------------------------------------
Policy iteration: 4
Solving linear equations...
------------------------------------------------------------
Final values:
         0       1       2       3       4       5       6       7       8       9
0 -1000.00  310.27  392.14  310.27 -100.00 -100.00  482.58  609.93  770.89  867.45
1     0.00    0.00  467.08    0.00  669.09  596.92    0.00    0.00  854.91 1000.00
2     0.00 -100.00  648.72  751.02  854.68  745.43  101.95  387.32  728.08  863.22
3     0.00  569.03  732.73  855.16 1000.00  848.14 -10000.00 -1521.13 -10000.00  372.09
4   403.69  560.69  651.51  750.72  848.62  678.34 -100.00 -1000.00    0.00  294.40
5     0.00  448.74  500.47 -100.00  593.00 -100.00  595.63  522.82  376.43 -100.00
6 -1000.00 -100.00  261.34 -1000.00  336.96    0.00  774.42  604.63    0.00    0.00
7   104.70  166.03  226.90  264.37  257.41 -100.00 1000.00    0.00  402.91  383.07
8   111.58  129.58    0.00  306.64    0.00  651.26  838.04  660.34  511.71  433.80
9    98.39  102.16    0.00  425.89  499.94  631.87  717.21  632.77    0.00  343.22
Final policy:
       0      1  2      3      4      5      6      7      8      9
0 -1e+03      →  ↓      ← -1e+02 -1e+02      →      →      →      ↓
1      ↓      0  ↓      0      ↓      ↓      0      0      →  1e+03
2      0 -1e+02  →      ↓      ↓      ←      ↑      →      ↑      ↑
3      0      →  →      →  1e+03      ← -1e+04      ↑ -1e+04      →
4      →      →  →      ↑      ↑      ← -1e+02 -1e+03      0      ↑
5      0      ↑  ↑ -1e+02      ↑ -1e+02      ↓      ↓      ← -1e+02
6 -1e+03 -1e+02  ↑ -1e+03      ↑      0      ↓      ←      0      0
7      ↓      →  ↑      ↓      ↑ -1e+02  1e+03      0      ↓      ↓
8      →      ↑  0      ↓      0      →      ↑      ←      ←      ←
9      ↑      ↑  0      →      →      →      ↑      ←      0      ↑
Total value iterations: 0
Total policy iterations: 4
Runtime:0.03352618217468262s
```

4. Experiments (Runtime)

1) Value Iteration vs. Policy Iteration vs. Policy Iteration based on Linear Equations Solving

board_size	v_n_v_iter	v_runtime	p_n_v_iter	p_n_p_iter	p_runtime	pl_n_v_iter	pl_n_p_iter	pl_runtime
6	35	0.01527	19	5	0.01469	0	1	0.00413
7	17	0.00915	52	5	0.02330	0	5	0.01369
10	44	0.04054	151	4	0.08571	0	4	0.02571
20	55	0.33576	184	8	0.65270	0	10	0.36991
30	62	0.83874	218	10	1.78188	0	10	0.95684
100	82	12.76358	621	17	45.94693	0	18	98.81403



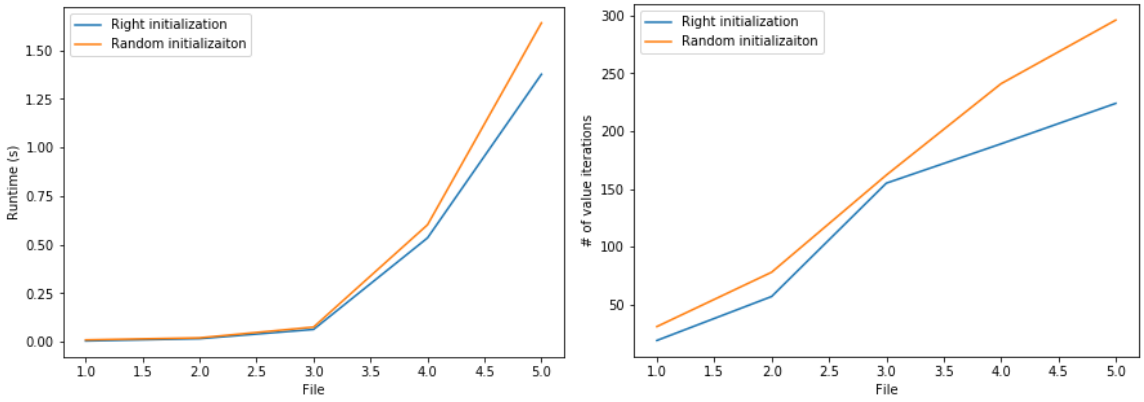
Conclusion:
In value iteration, the algorithm needs to check all of the possible actions. It works pretty good when the board is large.

The normal policy iteration, which uses iteration to evaluate the policy, only needs to check one fixed action. Therefore, policy iteration will take less time for each step of calculating q values. However, the experiment shows that normal policy iteration needs more iterations to converge, which results in costing more runtime.

The policy iteration based on linear equations solving performs quite good in the experiment when the size of the board is small. However, when the board size is larger than 100, it takes forever for solving the equations.

2) Compare different initialization method

board_size	p_i_n_v_iter	p_i_n_p_iter	p_i_runtime	p_n_v_iter	p_n_p_iter	p_runtime
6	19	1	0.00431	31	5	0.00966
7	57	5	0.01588	78	5	0.02025
10	155	4	0.06356	162	4	0.07570
20	189	10	0.53469	241	8	0.60211
30	224	11	1.37734	296	11	1.64174

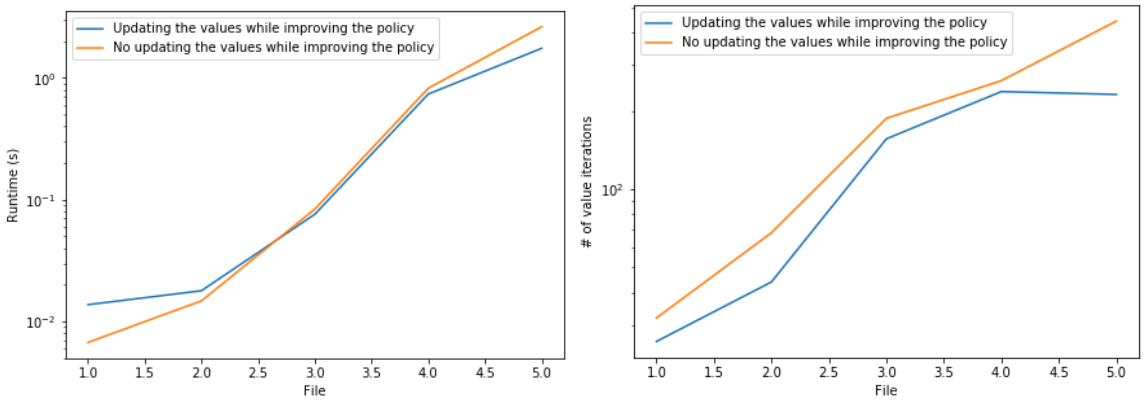


Conclusion:
I tried different initialization methods for policy, both random and choosing a direction for all states.

As shown in the experiment above, in some cases, a good policy initialization can reduce the runtime as well as the number of iterations needed.

3) Improve the values while improving the policy

board_size	p_u_n_v_iter	p_u_n_p_iter	p_u_runtime	p_n_v_iter	p_n_p_iter	p_runtime
6	26	4	0.01373	32	2	0.00672
7	44	3	0.01787	68	4	0.01475
10	156	4	0.07598	187	4	0.08377
20	237	8	0.73735	261	25	0.82293
30	231	9	1.75129	442	21	2.63338



Conclusion:
When I was using the normal policy iteration, I tried to improve the values while improving the policy. As shown in the experiment above, it can reduce the runtime as well as the number of iterations needed. This is because in this way the time for the first value iteration in evaluating policy can be saved.

5) Summary

On the one hand, the value iteration is faster than policy iteration when the scale of the task is large. On the other, the policy iteration based on linear equations solving works better when the size of the task is small. In conclusion, it is a trade-off to decide which algorithm we should use, and it also depends on the case.