Joel Doumit
9/4/18
CS472
Assignment 1a

Abstract

This project was intended to lay the foundation for later evolutionary algorithmic work. In this project, we simply created a population consisting of 100 8-entry arrays, filled with a permutation of the numbers 0 through 7, which represented not only the number of the queen to be placed, but both its position vertically via the number, and horizontally via the column position. In addition, we had to provide fitness scores for each of the arrays. That is, the number of times the queens are in conflict with one another. My solution for that will be discussed in a later section.

Algorithm

By nature of permutations, we didn't have to worry about any vertical or horizontal conflicts between queens. That only left diagonal conflicts, which I determined were calculable by adding or subtracting the number of the column a queen was in to its value (which represented its horizontal position). By checking if any of these values matched in a given array, I was able to determine if there were any conflicts, and if there were, how many. Adding up these conflicts is what determines the fitness of a particular individual. The closer the score is to 0, the better the board state.

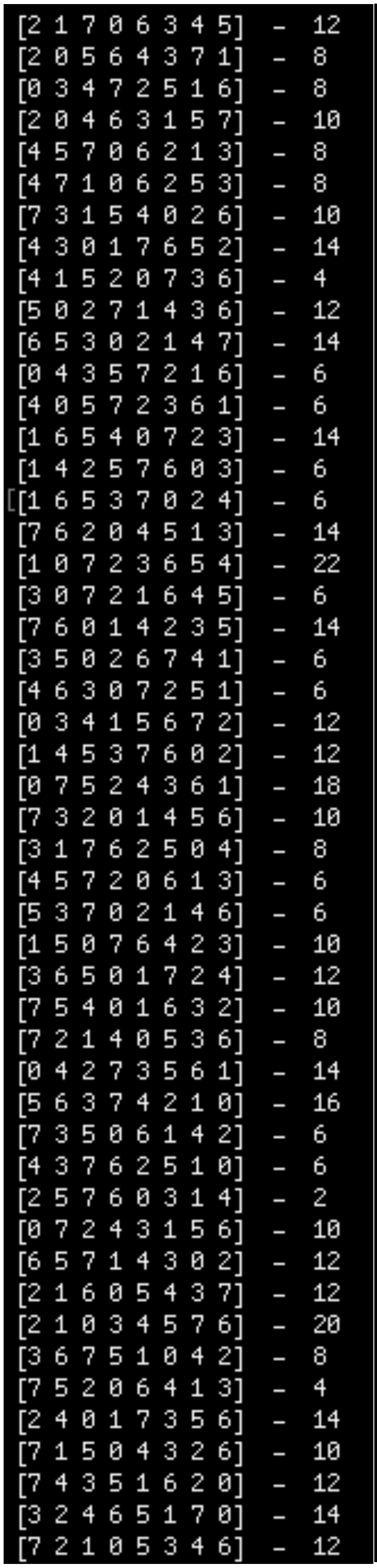
Output

```
[2 1 7 0 6 3 4 5]  -  12     [6 5 3 0 2 1 4 7]  -  14     [3 5 0 2 6 7 4 1]  -  6
[2 0 5 6 4 3 7 1]  -  8      [0 4 3 5 7 2 1 6]  -  6      [4 6 3 0 7 2 5 1]  -  6
[0 3 4 7 2 5 1 6]  -  8      [4 0 5 7 2 3 6 1]  -  6      [0 3 4 1 5 6 7 2]  -  12
[2 0 4 6 3 1 5 7]  -  10     [1 6 5 4 0 7 2 3]  -  14     [1 4 5 3 7 6 0 2]  -  12
[4 5 7 0 6 2 1 3]  -  8      [1 4 2 5 7 6 0 3]  -  6      [0 7 5 2 4 3 6 1]  -  18
[4 7 1 0 6 2 5 3]  -  8      [1 6 5 3 7 0 2 4]  -  6      [7 3 2 0 1 4 5 6]  -  10
[7 3 1 5 4 0 2 6]  -  10     [7 6 2 0 4 5 1 3]  -  14     [3 1 7 6 2 5 0 4]  -  8
[4 3 0 1 7 6 5 2]  -  14     [1 0 7 2 3 6 5 4]  -  22     [4 5 7 2 0 6 1 3]  -  6
[4 1 5 2 0 7 3 6]  -  4      [3 0 7 2 1 6 4 5]  -  6      [5 3 7 0 2 1 4 6]  -  6
[5 0 2 7 1 4 3 6]  -  12     [7 6 0 1 4 2 3 5]  -  14     [1 5 0 7 6 4 2 3]  -  10
```

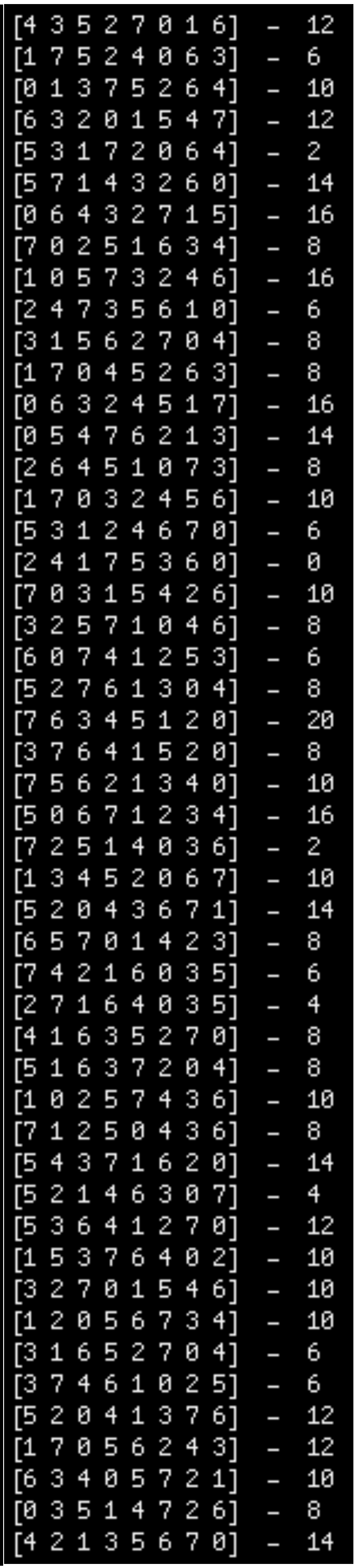
[3 6 5 0 1 7 2 4] - 12
[7 5 4 0 1 6 3 2] - 10
[7 2 1 4 0 5 3 6] - 8
[0 4 2 7 3 5 6 1] - 14
[5 6 3 7 4 2 1 0] - 16
[7 3 5 0 6 1 4 2] - 6
[4 3 7 6 2 5 1 0] - 6
[2 5 7 6 0 3 1 4] - 2
[0 7 2 4 3 1 5 6] - 10
[6 5 7 1 4 3 0 2] - 12
[2 1 6 0 5 4 3 7] - 12
[2 1 0 3 4 5 7 6] - 20
[3 6 7 5 1 0 4 2] - 8
[7 5 2 0 6 4 1 3] - 4
[2 4 0 1 7 3 5 6] - 14
[7 1 5 0 4 3 2 6] - 10
[7 4 3 5 1 6 2 0] - 12
[3 2 4 6 5 1 7 0] - 14
[7 2 1 0 5 3 4 6] - 12
[4 3 5 2 7 0 1 6] - 12
[1 7 5 2 4 0 6 3] - 6
[0 1 3 7 5 2 6 4] - 10
[6 3 2 0 1 5 4 7] - 12
[5 3 1 7 2 0 6 4] - 2

[5 7 1 4 3 2 6 0] - 14
[0 6 4 3 2 7 1 5] - 16
[7 0 2 5 1 6 3 4] - 8
[1 0 5 7 3 2 4 6] - 16
[2 4 7 3 5 6 1 0] - 6
[3 1 5 6 2 7 0 4] - 8
[1 7 0 4 5 2 6 3] - 8
[0 6 3 2 4 5 1 7] - 16
[0 5 4 7 6 2 1 3] - 14
[2 6 4 5 1 0 7 3] - 8
[1 7 0 3 2 4 5 6] - 10
[5 3 1 2 4 6 7 0] - 6
[2 4 1 7 5 3 6 0] - 0
[7 0 3 1 5 4 2 6] - 10
[3 2 5 7 1 0 4 6] - 8
[6 0 7 4 1 2 5 3] - 6
[5 2 7 6 1 3 0 4] - 8
[7 6 3 4 5 1 2 0] - 20
[3 7 6 4 1 5 2 0] - 8
[7 5 6 2 1 3 4 0] - 10
[5 0 6 7 1 2 3 4] - 16
[7 2 5 1 4 0 3 6] - 2
[1 3 4 5 2 0 6 7] - 10
[5 2 0 4 3 6 7 1] - 14

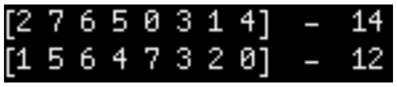
[6 5 7 0 1 4 2 3] - 8
[7 4 2 1 6 0 3 5] - 6
[2 7 1 6 4 0 3 5] - 4
[4 1 6 3 5 2 7 0] - 8
[5 1 6 3 7 2 0 4] - 8
[1 0 2 5 7 4 3 6] - 10
[7 1 2 5 0 4 3 6] - 8
[5 4 3 7 1 6 2 0] - 14
[5 2 1 4 6 3 0 7] - 4
[5 3 6 4 1 2 7 0] - 12
[1 5 3 7 6 4 0 2] - 10
[3 2 7 0 1 5 4 6] - 10
[1 2 0 5 6 7 3 4] - 10
[3 1 6 5 2 7 0 4] - 6
[3 7 4 6 1 0 2 5] - 6
[5 2 0 4 1 3 7 6] - 12
[1 7 0 5 6 2 4 3] - 12
[6 3 4 0 5 7 2 1] - 10
[0 3 5 1 4 7 2 6] - 8
[4 2 1 3 5 6 7 0] - 14
[2 7 6 5 0 3 1 4] - 14
[1 5 6 4 7 3 2 0] - 12



```
[2 1 7 0 6 3 4 5]  -  12
[2 0 5 6 4 3 7 1]  -  8
[0 3 4 7 2 5 1 6]  -  8
[2 0 4 6 3 1 5 7]  -  10
[4 5 7 0 6 2 1 3]  -  8
[4 7 1 0 6 2 5 3]  -  8
[7 3 1 5 4 0 2 6]  -  10
[4 3 0 1 7 6 5 2]  -  14
[4 1 5 2 0 7 3 6]  -  4
[5 0 2 7 1 4 3 6]  -  12
[6 5 3 0 2 1 4 7]  -  14
[0 4 3 5 7 2 1 6]  -  6
[4 0 5 7 2 3 6 1]  -  6
[1 6 5 4 0 7 2 3]  -  14
[1 4 2 5 7 6 0 3]  -  6
[[1 6 5 3 7 0 2 4]  -  6
[7 6 2 0 4 5 1 3]  -  14
[1 0 7 2 3 6 5 4]  -  22
[3 0 7 2 1 6 4 5]  -  6
[7 6 0 1 4 2 3 5]  -  14
[3 5 0 2 6 7 4 1]  -  6
[4 6 3 0 7 2 5 1]  -  6
[0 3 4 1 5 6 7 2]  -  12
[1 4 5 3 7 6 0 2]  -  12
[0 7 5 2 4 3 6 1]  -  18
[7 3 2 0 1 4 5 6]  -  10
[3 1 7 6 2 5 0 4]  -  8
[4 5 7 2 0 6 1 3]  -  6
[5 3 7 0 2 1 4 6]  -  6
[1 5 0 7 6 4 2 3]  -  10
[3 6 5 0 1 7 2 4]  -  12
[7 5 4 0 1 6 3 2]  -  10
[7 2 1 4 0 5 3 6]  -  8
[0 4 2 7 3 5 6 1]  -  14
[5 6 3 7 4 2 1 0]  -  16
[7 3 5 0 6 1 4 2]  -  6
[4 3 7 6 2 5 1 0]  -  6
[2 5 7 6 0 3 1 4]  -  2
[0 7 2 4 3 1 5 6]  -  10
[6 5 7 1 4 3 0 2]  -  12
[2 1 6 0 5 4 3 7]  -  12
[2 1 0 3 4 5 7 6]  -  20
[3 6 7 5 1 0 4 2]  -  8
[7 5 2 0 6 4 1 3]  -  4
[2 4 0 1 7 3 5 6]  -  14
[7 1 5 0 4 3 2 6]  -  10
[7 4 3 5 1 6 2 0]  -  12
[3 2 4 6 5 1 7 0]  -  14
[7 2 1 0 5 3 4 6]  -  12
```



```
[4 3 5 2 7 0 1 6]  -  12
[1 7 5 2 4 0 6 3]  -  6
[0 1 3 7 5 2 6 4]  -  10
[6 3 2 0 1 5 4 7]  -  12
[5 3 1 7 2 0 6 4]  -  2
[5 7 1 4 3 2 6 0]  -  14
[0 6 4 3 2 7 1 5]  -  16
[7 0 2 5 1 6 3 4]  -  8
[1 0 5 7 3 2 4 6]  -  16
[2 4 7 3 5 6 1 0]  -  6
[3 1 5 6 2 7 0 4]  -  8
[1 7 0 4 5 2 6 3]  -  8
[0 6 3 2 4 5 1 7]  -  16
[0 5 4 7 6 2 1 3]  -  14
[2 6 4 5 1 0 7 3]  -  8
[1 7 0 3 2 4 5 6]  -  10
[5 3 1 2 4 6 7 0]  -  6
[2 4 1 7 5 3 6 0]  -  0
[7 0 3 1 5 4 2 6]  -  10
[3 2 5 7 1 0 4 6]  -  8
[6 0 7 4 1 2 5 3]  -  6
[5 2 7 6 1 3 0 4]  -  8
[7 6 3 4 5 1 2 0]  -  20
[3 7 6 4 1 5 2 0]  -  8
[7 5 6 2 1 3 4 0]  -  10
[5 0 6 7 1 2 3 4]  -  16
[7 2 5 1 4 0 3 6]  -  2
[1 3 4 5 2 0 6 7]  -  10
[5 2 0 4 3 6 7 1]  -  14
[6 5 7 0 1 4 2 3]  -  8
[7 4 2 1 6 0 3 5]  -  6
[2 7 1 6 4 0 3 5]  -  4
[4 1 6 3 5 2 7 0]  -  8
[5 1 6 3 7 2 0 4]  -  8
[1 0 2 5 7 4 3 6]  -  10
[7 1 2 5 0 4 3 6]  -  8
[5 4 3 7 1 6 2 0]  -  14
[5 2 1 4 6 3 0 7]  -  4
[5 3 6 4 1 2 7 0]  -  12
[1 5 3 7 6 4 0 2]  -  10
[3 2 7 0 1 5 4 6]  -  10
[1 2 0 5 6 7 3 4]  -  10
[3 1 6 5 2 7 0 4]  -  6
[3 7 4 6 1 0 2 5]  -  6
[5 2 0 4 1 3 7 6]  -  12
[1 7 0 5 6 2 4 3]  -  12
[6 3 4 0 5 7 2 1]  -  10
[0 3 5 1 4 7 2 6]  -  8
[4 2 1 3 5 6 7 0]  -  14
```


```
[2 7 6 5 0 3 1 4]  -  14
[1 5 6 4 7 3 2 0]  -  12
```

Code

```
#!/Library/Frameworks/Python.framework/Versions/3.4/bin/python3
# Joel Doumit
# CS472 - Evolutionary Algorithms
# Assignment 1a - 8 Queens

import numpy as np

def initializeBoard():
	return np.random.permutation(8)

def initializePopulation(size):
	return [initializeBoard() for i in range(size)]

def checkMatch(pos):
	counts = (np.unique(pos, return_counts=True))[1]
	conflicts = [i * (i-1) for i in counts]
	return np.sum(conflicts)

def fitness(board):
	score = 0
	posdiag = [0, 1, 2, 3, 4, 5, 6, 7]
	negdiag = [0, -1, -2, -3, -4, -5, -6, -7]

	checkpos = [x + y for x, y in zip(board, posdiag)]
	checkneg = [x + y for x, y in zip(board, negdiag)]
```

```
        score = (checkMatch(checkpos) + checkMatch(checkneg))

        return score

if __name__ == "__main__":
        population = initializePopulation(100)
        for i in population:
                print(i, " - ", fitness(i))
```