Q3

May 17, 2023

[1]:

```
import random
```

[38]:

```
class chromosome:
    def __init__(self,n):
        self.n = n
        self.p_values = []                                                  #␣
 ↪Possible Values in a puzzle
        for i in range(n*n):
            self.p_values.append(i)

    def generate_chromosome(self):
        self.p = []
        temp = self.p_values[:]
        for i in range(self.n):
            self.p.append([])
            for j in range(self.n):
                temp1 = temp[random.randint(0,len(temp)-1)]
                self.p[i].append(temp1)
                temp.remove(temp1)

    def chk_greater(self,row,column):                          # This will check␣
 ↪the values that are smaller then the value at position row,column
        count = 0
        for i in range(row,self.n):
            for j in range(self.n):
                if(i == row and j <= column):
                    continue
                if(self.p[i][j] == 0):
                    continue
                if(self.p[i][j] < self.p[row][column]):
                    count += 1
        return count

    def fitness(self):
        self.fit = 0
        for i in range(self.n):
            for j in range(self.n):
                self.fit += self.chk_greater(i,j)
```

[39]:

```
c = chromosome(3)
```

[48]:

```
c.generate_chromosome()
```

[49]:

```
c.p
```

[49]:

```
[[3, 7, 5], [1, 2, 0], [8, 6, 4]]
```

[28]:

```
# 0 Means empty space
```

[50]:

```
c.fitness()
```

[51]:

```
c.fit
```

[51]:

```
13
```

[94]:

```
# Fitness should be zero for reaching the goal
```

[37]:

```
c.p = [[0,1,2],[3,4,5],[6,7,8]]
c.fitness()
c.fit
```

[37]:

```
0
```

[ ]: