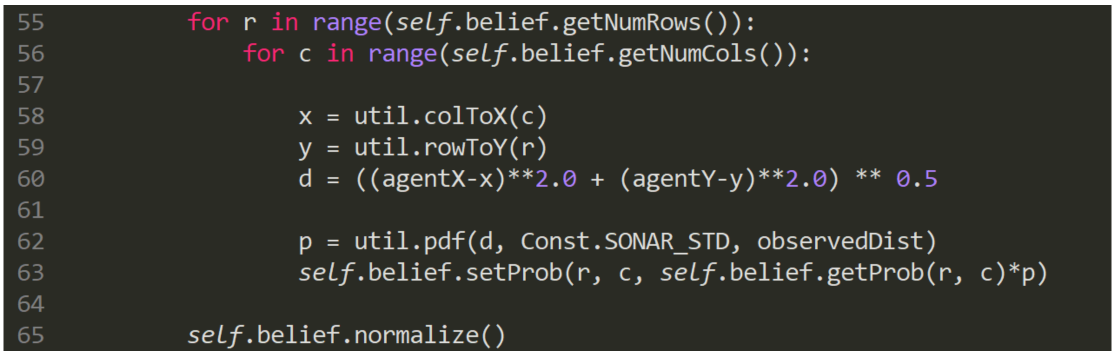
Homework 5: Car Tracking

109550112 林秉

Part I. Implementation (20%):

Part1:



```
55          for r in range(self.belief.getNumRows()):
56              for c in range(self.belief.getNumCols()):
57
58                  x = util.colToX(c)
59                  y = util.rowToY(r)
60                  d = ((agentX-x)**2.0 + (agentY-y)**2.0) ** 0.5
61
62                  p = util.pdf(d, Const.SONAR_STD, observedDist)
63                  self.belief.setProb(r, c, self.belief.getProb(r, c)*p)
64
65          self.belief.normalize()
```

In line 55 and 56, we locate values in matrix one-by-one by for loop.

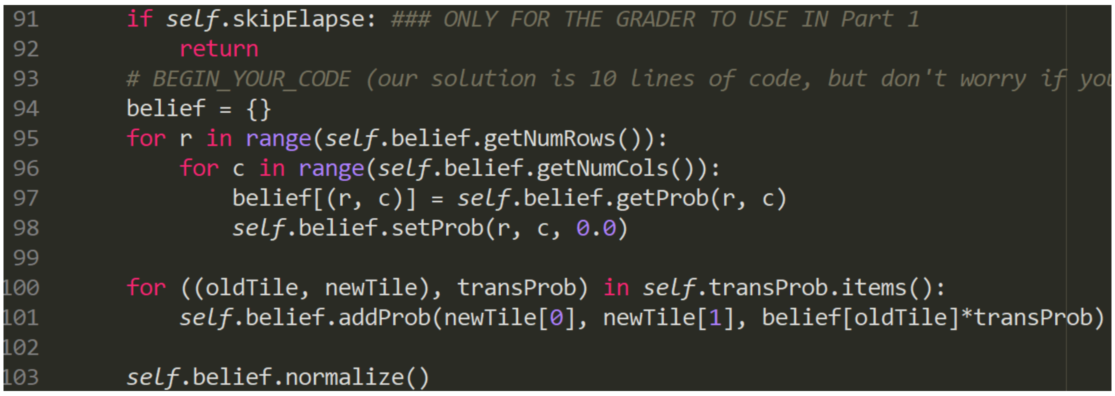
In line 58 to 60, we get the value from matrix and calculate the distance.

In line 62, since we now have the distance and the variance, we can sample the probability of the current car's position by the distribution function of the value observedDist.

In line 63, after we get the probability, we update the value in matrix by multiplying the probability with the old value.

In line 65, we normalize the matrix since the total probability value is not equal to 1.

Part2:

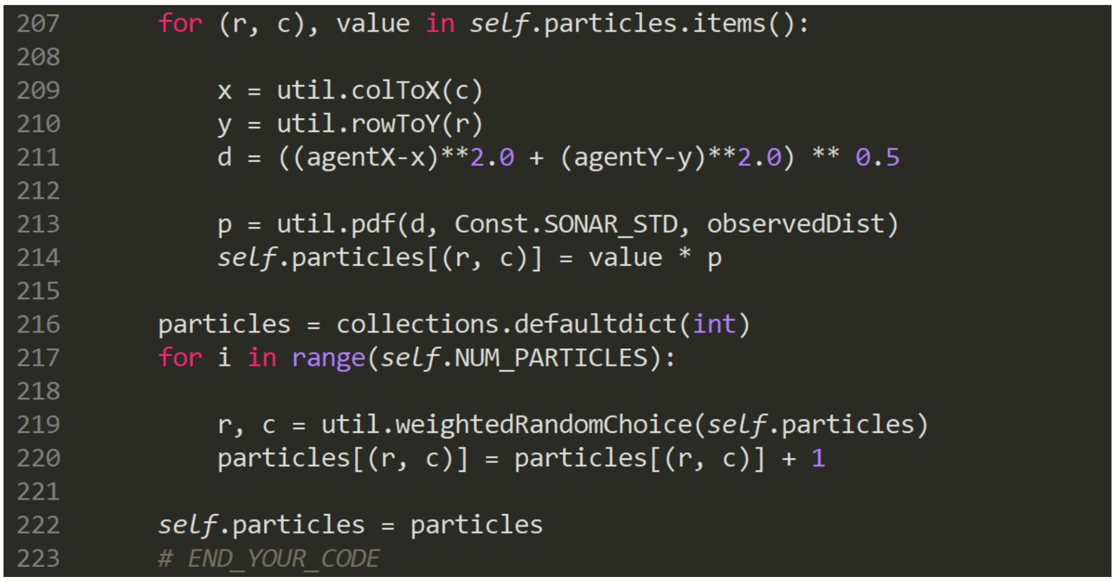


```
91          if self.skipElapse: ### ONLY FOR THE GRADER TO USE IN Part 1
92              return
93          # BEGIN_YOUR_CODE (our solution is 10 lines of code, but don't worry if yo
94          belief = {}
95          for r in range(self.belief.getNumRows()):
96              for c in range(self.belief.getNumCols()):
97                  belief[(r, c)] = self.belief.getProb(r, c)
98                  self.belief.setProb(r, c, 0.0)
99
100         for ((oldTile, newTile), transProb) in self.transProb.items():
101             self.belief.addProb(newTile[0], newTile[1], belief[oldTile]*transProb)
102
103         self.belief.normalize()
```

In line 94 to 98, we want another dictionary to store the original data, we create a new dictionary and copy everything from the old one to new one, then set the old dictionary with value 0.

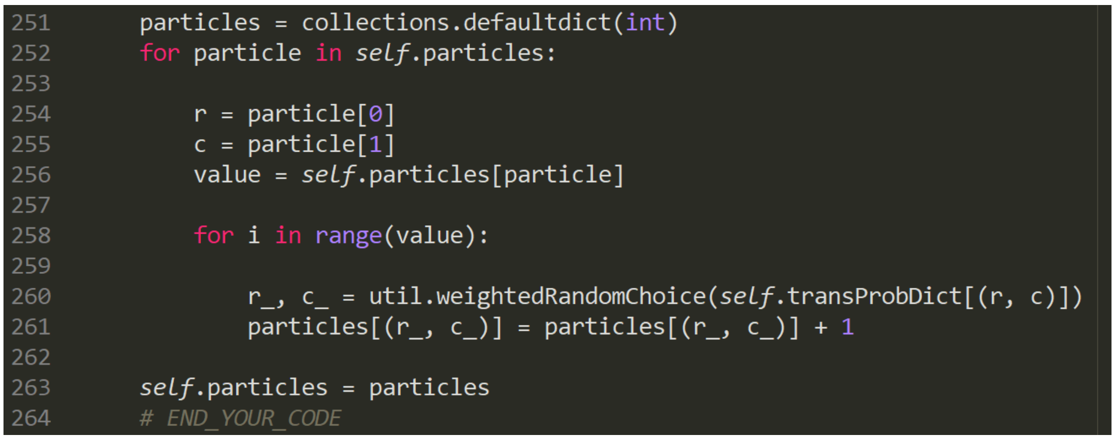
In line 100 to 101, since we want to predict the future position, for every key pair of r and c, we update the new value my multiplying the value of the old key pair and the transition probability, which is given in the transProb.items().

Part3-1:



```
207         for (r, c), value in self.particles.items():
208 
209             x = util.colToX(c)
210             y = util.rowToY(r)
211             d = ((agentX-x)**2.0 + (agentY-y)**2.0) ** 0.5
212 
213             p = util.pdf(d, Const.SONAR_STD, observedDist)
214             self.particles[(r, c)] = value * p
215 
216         particles = collections.defaultdict(int)
217         for i in range(self.NUM_PARTICLES):
218 
219             r, c = util.weightedRandomChoice(self.particles)
220             particles[(r, c)] = particles[(r, c)] + 1
221 
222         self.particles = particles
223         # END_YOUR_CODE
```

In line 207 to 211, we again calculate the car distance like we did in part1.
In line 213, we sample the probability of the observedDist by the distance and the variance.
In line 214, we update the value of the key pair value by multiplying the dot count and the probability in line 213.
After line 214, we get a new probability distribution, but the matrix's value type has to be int since it stores the number of the dot count, so we have a new matrix in line 216, then for the total dot count, we sample a dot's position by the new probability distribution, after that, we update the matrix value with the sampled row and column.
Part3-2:



```
251         particles = collections.defaultdict(int)
252         for particle in self.particles:
253 
254             r = particle[0]
255             c = particle[1]
256             value = self.particles[particle]
257 
258             for i in range(value):
259 
260                 r_, c_ = util.weightedRandomChoice(self.transProbDict[(r, c)])
261                 particles[(r_, c_)] = particles[(r_, c_)] + 1
262 
263         self.particles = particles
264         # END_YOUR_CODE
```

In line 251, we have a new dictionary to store the prediction.
In line 254 to 256, we have the dot count of the given row and column.
In line 258 to 261, for every dots in the row column pair, we sample the new dots position by the given transition function with the row column pair, then we update the dictionary in line 261, after the process, the predicted dot position is stored in the new dictionary.