Homework 5: Car Tracking

110550091 吳承瑀

Part I. Implementation (15%):

Part 1:

```python
    def observe(self, agentX: int, agentY: int, observedDist: float) -> None:
        # BEGIN_YOUR_CODE

        for r in range (self.belief.numRows):
            for c in range (self.belief.numCols): # for each place in grid
                # dist of the grid to my car: dist = sqrt((X-x)^2 + (Y-y)^2)
                MyCarDist = math.sqrt((util.colToX(c) - agentX) ** 2 + (util.rowToY(r) - agentY) ** 2)
                # pdf: mean = MyCarDist, std = Const.SONAR_STD, value = observeDist
                CalculatedPDF = util.pdf(MyCarDist, Const.SONAR_STD, observedDist)
                CurrentProbability = self.belief.getProb(r, c)
                self.belief.setProb(r, c, CurrentProbability * CalculatedPDF) # update probability
        self.belief.normalize() # normalize self.belief

        # END_YOUR_CODE
```

Part 2:

```python
    def elapseTime(self) -> None:
        if self.skipElapse: ### ONLY FOR THE GRADER TO USE IN Part 1
            return
        # BEGIN_YOUR_CODE

        # new belief for update with default all 0
        newBelief = util.Belief(self.belief.numRows, self.belief.numCols, value=0)
        for oldTile, newTile in self.transProb:
            oldr, oldc = oldTile # old col & row
            newr, newc = newTile # new col & row
            CurrentProb = self.belief.getProb(oldr, oldc) # get current probability of current(old) row & col
            TransProb = self.transProb[(oldTile, newTile)] # get transprob with (old,new) pair
            # update probability with new location and delta(cur_prob*trans_prob)
            newBelief.addProb(newr, newc, CurrentProb * TransProb)
        newBelief.normalize()
        self.belief = newBelief # update normalized belief

        # END_YOUR_CODE
```

Part 3-1:

```python
    def observe(self, agentX: int, agentY: int, observedDist: float) -> None:
        # BEGIN_YOUR_CODE

        # create new dictionary to store current particles
        tempParticles = collections.defaultdict(float)
        # create new dictionary for new particles
        newParticles = collections.defaultdict(int)
        for r, c in self.particles:
            # dist of the grid to my car: dist = sqrt((X-x)^2 + (Y-y)^2)
            MyCarDist = math.sqrt((util.colToX(c) - agentX) ** 2 + (util.rowToY(r) - agentY) ** 2)
            # pdf: mean = MyCarDist, std = Const.SONAR_STD, value = observeDist
            CalculatedPDF = util.pdf(MyCarDist, Const.SONAR_STD, observedDist)
            # update new dictionary with current particle*pdf
            tempParticles[(r, c)] = self.particles[(r, c)] * CalculatedPDF

        for _ in range(self.NUM_PARTICLES):
            # new NUM_PARTICLES sampled from the new re-weighted distribution
            particle = util.weightedRandomChoice(tempParticles)
            newParticles[particle] += 1 # dict : add 1 of val which index = particle

        self.particles = newParticles # update new particles

        # END_YOUR_CODE
```

Part 3-2:

```python
    def elapseTime(self) -> None:

        # BEGIN_YOUR_CODE

        # create new dictionary for new particles
        newParticles = collections.defaultdict(int)
        for p in self.particles: # loop over particles
            val = self.particles[p] # corresponding particles at the location
            for _ in range(val):  #call weightedRandomChoice for every particles at the location
                #self.transProbDict[oldTile][newTile], particle = oldtile; new_t = newtile(new weight dict)
                newTile = self.transProbDict[p]
                # weightedRandomChoice based on new weight dict
                tempParticles = util.weightedRandomChoice(newTile)
                # dict : add 1 of val which index = temp_particle
                newParticles[tempParticles] += 1
        self.particles = newParticles # update particles

        # END_YOUR_CODE
```

Part 2. Question answering (5%):

Understanding the functionality and interaction between different functions proved to be the most challenging aspect. It required extensive reading and comprehension of numerous notes and functions before I could commence the project. However, once I grasped the workings of all the functions in util.py, the actual code implementation did not pose much difficulty. Surprisingly, the code was shorter in length than I had initially anticipated.