Artificial Intelligence HW2
Better Evaluation Function Report

EE2 B05901172 黃禹靖

1. Food eating:

```
    if not foodlist:
        dist_greedyeat=0
    else:
        dist_greedyeat=min([ util.manhattanDistance(position,i) for i
    in foodlist ])
    eatingfood =-50*len(foodlist)-dist_greedyeat+random.randint(0,2)
```

I. Pacman greedy find the nearest food. Because pacman should know where the nearest food is, so the evaluation value is minus ‘dist_greedyeat’, where ‘dist_greedyeat’ is the manhattan distance from pacman’s current position to the position of the nearest food, so that the value will become greater as pacman approaching the nearest food.

II. However, after swallowing a food, the value would become lesser suddenly because ‘dist_greedyeat’ now is the distance of next nearest food. It would cause pacman to stand still in front of a food, in where the value of minus ‘dist_greedyeat’ is the greatest. So I simply add a new variable, the amount of food to ‘eatingfood’ to avoid this problem.

III. In addition, because I only use manhattandistance to evaluate distance from food, which have some error to the real condition, so I add some random number to avoid stuck in some position.

2. Capsule eating

```
 if not capsulelist:
    dist_greedycapsule=0
else:
    dist_greedycapsule=min([ util.manhattanDistance(position,i) for i
 in capsulelist ])
eatingcapsule = -50*len(capsulelist)-dist_greedycapsule
ischasing=False
for i in GhostStates:
    if i.scaredTimer>util.manhattanDistance(position,i.getPosition()):
        ischasing=True
        break
if ischasing and len(capsulelist)==1 :
    eatingcapsule+=2
```

I. Capsule eating is almost same as food eating.

II. The only thing different from food eating is that when pacman is chasing ghosts, I don't want him to eating another capsule, which is wasteful. So I add some variable to keep him from this problem.

3. Ghost chasing

```
 nearghost,dist_chase=0,0
for i in GhostStates:
    dist_ghost = util.manhattanDistance(position,i.getPosition())
    if i.scaredTimer > dist_ghost:
        dist_chase+=dist_ghost
    elif dist_ghost < 2:
        nearghost+=(2-dist_ghost)
ghostchase = -dist_chase-5*nearghost
```

I. This part includes chasing ghost and escaping from ghost.

II. When pacman see ghost in certain distance, the evaluation value would drop as getting closer to ghosts.

III. When a ghost is in scared time, and it is possible for pacman to eat the ghost before scared time ends, pacman would start chasing it. Similar to food eating, pacman finds the direction of the ghosts by manhattan distance from pacman's current position to ghosts' position.

4. **Special case**

```
 specialcase=0
deadzone = [ (9,5), (12,5), (10,5), (11,5) ]
if position in deadzone:
    specialcase-=1
if position in deadzone[:2]:
    specialcase-=1

 islosing=0
if currentGameState.isLose():
    islosing=1
```

I. This is part is heavily maze dependent.

II. I found that it is very likely to lose when pacman is in certain area (deadzone in code). First, pacman may stuck in that area in smallClaasic maze. Second, more important, the area is where the eaten ghost reborn. If pacman eats ghost in the area, he is very likely to bump into the reborn ghost right away. So I don't want pacman to stay in the area in any case.

III. The evaluation value drops dramatically if losing. I just want pacman struggle to live.

5. Final evaluation value

```
Return eatingfood+40*eatingcapsule+100*ghostchase+99999*specialcase-
9999999*islosing
```

I. The final value of evaluation function is linear combination of the variable discuss above.