Partiel de robotique
25 juin 2014 session 2

1 Lecture et amélioration de programme

Il y a un mois, Pippo a écrit un programme qui simule de façon probabiliste le déplacement d'un robot dans un couloir. Plus précisément le problème résolu est le suivant.

> Un robot se déplace dans un couloir de N cases, en avançant case par case dans l'une ou l'autre des deux directions. À chaque fois que le robot se déplace d'une case il y a une probabilité `p[0]` qu'il ne se soit pas déplacé (il reste sur sa case d'origine), une probabilité `p[1]` qu'il se soit déplacé d'exactement une case dans la bonne direction, une probabilité `p[2]` qu'il soit allé trop loin d'une case, etc. Le nombre de possibilités d'erreurs peut varier, mais la somme des probabilités fait toujours 1. Écrire un programme qui simule le déplacement du robot à l'aide d'actions au clavier lui ordonnant de prendre l'une ou l'autre des deux directions, et, pour chaque, case afficher sa probabilité de présence en cette case.

Pippo a construit un programme complet et correct que voici.

```
 1  # Paramètres ####
 2  N = 10
 3  p = [0., 0.7, 0.2, 0.1]
 4  K = len(p)
 5
 6  def un_char():
 7      # Cette fonction récupère un caractère auprès de l'utlisateur sans
 8      # qu'il soit nécessaire que celui-ci tape <entrée>.
 9      import sys, tty, termios
10      fd = sys.stdin.fileno()
11      old_settings = termios.tcgetattr(fd)
12      try:
13          # passage en mode caractère
14          tty.setraw(sys.stdin.fileno())
15          ch = sys.stdin.read(1)
16      finally:
17          # retour en mode ligne
18          termios.tcsetattr(fd, termios.TCSADRAIN, old_settings)
19          return ch
20
21  def deplacer_est(t): # une case vers la droite ->
22      res = [0 for i in range(N)]
23      for i in range(N):
24          for j in range(0, min(i + 1, K)):
25              res[i] += t[i - j] * p[j]
26      return res
```

```
27
28  def deplacer_ouest(t): # une case vers la gauche <-
29      res = [0 for i in range(N)]
30      for i in range(N):
31          for j in range(0, min(N - i, K)):
32              res[i] += t[i + j] * p[j]
33      return res
34
35  def main():
36      t = [0 for i in range(N)] #terrain (un couloir)
37      t[0] = 1. #position initiale
38      continuer = True
39      while (continuer):
40            affichage_joli = [round(e, 2) for e in t]
41            print affichage_joli # affichage avec moins de décimales
42            c = un_char() # saisie utilisateur
43            if ((c == 'j')  or (c == 'J')):
44                t = deplacer_ouest(t)
45            if ((c == 'l')  or (c == 'L')):
46                t = deplacer_est(t)
47            if ((c == 'q') or (c == 'Q')):
48                continuer = False
49
50  main()
```

Comportement observé du programme :

```
1  RobotiqueL1$ python couloir.py
2  [1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
3  <l'utilisateur tape L (déplacement vers l'Est)>
4  [0.0, 0.7, 0.2, 0.1, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
5  <l'utilisateur tape J (déplacement vers l'Ouest)>
6  [0.54, 0.16, 0.07, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
7  <l'utilisateur tape J (déplacement vers l'Ouest)>
8  [0.13, 0.05, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
9  <l'utilisateur tape J (déplacement vers l'Ouest)>
10 [0.03, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
11 <l'utilisateur tape J (déplacement vers l'Ouest)>
12 [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
13 <l'utilisateur tape Q>
```

1.1 Mettre en rapport le code et le comportement (1.5 pt)

Pour chacune des questions suivantes répondre en donnant les numéros des lignes concernées du programme de Pippo.

1. Dans quelle case se trouve initialement le robot ?
2. Quelle est la probabilité que le robot ne se déplace pas lorsqu'il en a reçu l'ordre ?
3. À chaque ordre de déplacement d'une case dans une direction de combien de cases au maximum peut avancer le robot ?

1.2 Expliquer le comportement observé (1.5 pt)

Dans le comportement observé à la fin juste avant de quitter, où se trouve le robot et pourquoi ?

1.3 Pippo modifie le programme (2 pt)

Pippo ajoute les trois lignes suivantes à la fonction `deplacer_est`, juste avant le `return res`:

```
for i in range(0, K):
    for j in range(i + 1, K):
        res[N - 1] += t[N - 1 - i] * p[j]
```

ainsi que les trois lignes suivantes à la fin de la fonction `deplacer_ouest`, juste avant le `return res`:

```
for i in range(0, K):
    for j in range(i  + 1, K):
        res[0] += t[i] * p[j]
```

1. Expliquer ce que cela va changer au comportement général du robot.
2. Si l'on reprend l'exemple utilisé pour le comportement observé, c'est à dire si on tape L, puis quatre fois J, où se trouve le robot à la fin (donner les valeurs du tableau) ?

1.4 Obstacle : à vous de modifier le programme (5 pt)

On place un obstacle au milieu du couloir (case `t[N/2]`) que le robot ne peut pas franchir lorsqu'il vient de l'Ouest et se déplace vers l'Est (il y a, par exemple une marche que le robot peut descendre mais ne peut pas monter). Lorsqu'il butte dans l'obstacle le robot reste dans sa case actuelle (la case `t[N/2 - 1]`).

1. Modifier le programme pour tenir compte de cet obstacle.
2. Au bout d'un certain nombre de déplacements, y a t'il plus de chances de trouver le robot à gauche (Ouest) ou à droite (Est) de l'obstacle ?
3. On considère maintenant que l'obstacle interdit également les déplacements en provenance de l'Ouest (le robot reste alors en `t[N/2 + 1]`). Que devait vous également modifier dans le programme ? (l'obstacle est un mur qui occupe toute la case t[N/2]).
4. Au départ le robot est placé dans l'une ou l'autre des deux cases qui entourent l'obstacle avec une probabilité équivalente. Quel doit être le tableau de probabilité initial ? Que faut-il modifier dans le programme pour en tenir compte ?
5. Au bout d'un certain nombre de déplacements, y a t'il plus de chances de trouver le robot à gauche (Ouest) ou à droite (Est) de l'obstacle ?

2 Élaboration de programmes complets

2.1 NXT (5 pt)

Vous disposez d'un robot NXT équipé de deux roues motorisées reliées en A et B et d'un capteur de lumière dirigé vers le sol entre ses deux roues et branché sur le port 3. Ce robot se trouve actuellement sur une tâche sombre (intensité < 100). Droit devant lui se trouve une zone claire puis une nouvelle tâche sombre, à une distance inconnue.
Écrire un programme pour le robot qui l'amène et le stoppe exactement à mi-distance entre les deux tâche sombres. Attention le terrain peut-être en pente (mais avec une déclivité constante).
Vous pouvez répondre en donnant du pseudo-code, ça ne vaudra pas moins de points.

2.2 Boids (5 pts)

Proposer un programme de boids, en énonçant au préalable les règles utilisées et en montrant à quelles parties de votre code elles correspondent.
Vous pouvez répondre en donnant du pseudo-code, ça ne vaudra pas moins de points.