ISM MPSI/PCSI, Informatique
Année 2024/2025

Corrigé du TP Informatique 19

Exercice 1

1. On saisit :

```
h=lambda pos:[pos[0]-1,pos[1]]
b=lambda pos:[pos[0]+1,pos[1]]
g=lambda pos:[pos[0],pos[1]-1]
d=lambda pos:[pos[0],pos[1]+1]
```

2. On saisit :

```
def visite(laby,pos):
    """Bascule pos dans laby en état visité"""
    laby[pos[0]][pos[1]]=1
```

3. On saisit :

```
def etat(laby,pos):
    # prise en compte des bords
    L,C=len(laby),len(laby[0])
    if pos[0]<0 or pos[0]>=L or pos[1]<0 or pos[1]>=C:
        return 2
    else:
        return laby[pos[0]][pos[1]]
```

4. On saisit :

```
def voisinage(laby,pos):
    """Indique si pos dans laby admet un voisin pas encore visité"""
    return etat(laby,g(pos))*etat(laby,d(pos))*etat(laby,h(pos))*etat(laby,b(pos))==0
```

5. On saisit :

```
def voisin(laby,pos):
    """Renvoie un voisin de pos dans laby pas encore visité"""
    if etat(laby,g(pos))==0:
        return g(pos)
    elif etat(laby,d(pos))==0:
        return d(pos)
    elif etat(laby,h(pos))==0:
        return h(pos)
    elif etat(laby,b(pos))==0:
        return b(pos)
```

ou une version avec boucle :

```
def voisin(laby,pos):
    Lpos=[g,d,h,b]
    for f in Lpos:
        if etat(laby,f(pos))==0:
            return f(pos)
```

6. On saisit :

```
def dedale(laby,deb,fin):
    """Résout le labyrinthe laby en partant de la position deb"""
    chemin=Pile()
    chemin.empiler(deb)
    pos=deb
    while pos!=fin and not chemin.vide():
        visite(laby,pos)
        if voisinage(laby,pos):
            chemin.empiler(pos)
            pos=voisin(laby,pos)
        else:
            pos=chemin.depiler()
    if pos==fin:
        visite(laby,pos)
    return not chemin.vide(),chemin
```

On teste :

```
plt.imshow(np.array(laby),cmap='Greys',interpolation='nearest')
plt.show()
deb=[0,0]
fin=[3,6] # ou [1,6] pour pas de chemin
print("Recherche de chemin - origine :", deb)
print("priorité de transition : gauche/droite/haut/bas")
res,chemin=dedale(laby,deb,fin)
print("Résultat :")
print(res)
plt.imshow(np.array(laby),cmap='Greys',interpolation='nearest')
plt.show()
```

La pile `chemin` contient un chemin du début à la sortie.

Exercice 2

1. On saisit :

```
def creuse(laby,pos):
    """Creuse la case en position pos dans laby"""
    laby[pos[0]][pos[1]]=0

def mur(laby,pos):
    """Mure la case en position pos dans laby"""
    laby[pos[0]][pos[1]]=2
```

2. On saisit :

```
def avance(laby,pos):
    """Liste des cases creusables depuis pos dans laby"""
    L,C=len(laby),len(laby[0])
    res=[]
    if pos[0]>0 and etat(laby,h(pos))==2 and not voisinage(laby,h(pos)):
        res.append(h(pos))
    if pos[0]<L-1 and etat(laby,b(pos))==2 and not voisinage(laby,b(pos)):
        res.append(b(pos))
    if pos[1]>0 and etat(laby,g(pos))==2 and  not voisinage(laby,g(pos)):
        res.append(g(pos))
    if pos[1]<C-1 and etat(laby,d(pos))==2 and not voisinage(laby,d(pos)):
        res.append(d(pos))
    return res
```

3. On saisit :

```
def gener(laby,deb):
    """Génération d'un labyrinthe dans laby en partant de deb"""
    chemin=Pile()
    pos=deb
    chemin.empiler(pos)
    fin=False
    while not fin:
        liste=avance(laby,pos)
        creuse(laby,pos)
        n=len(liste)
        if n==0:
            pos=chemin.depiler()
            mur(laby,pos)
            fin=pos==deb
        else:
            pos=liste[rd.randint(n)]
            chemin.empiler(pos)
    creuse(laby,pos)
```

On teste :

```
L,C=8,15
laby1=[[2 for j in range(15)] for i in range(8)]
print("Génération de labyrinthe")
gener(laby1,[0,0])
plt.imshow(np.array(laby1),cmap='Greys',interpolation='nearest')
plt.show()
```

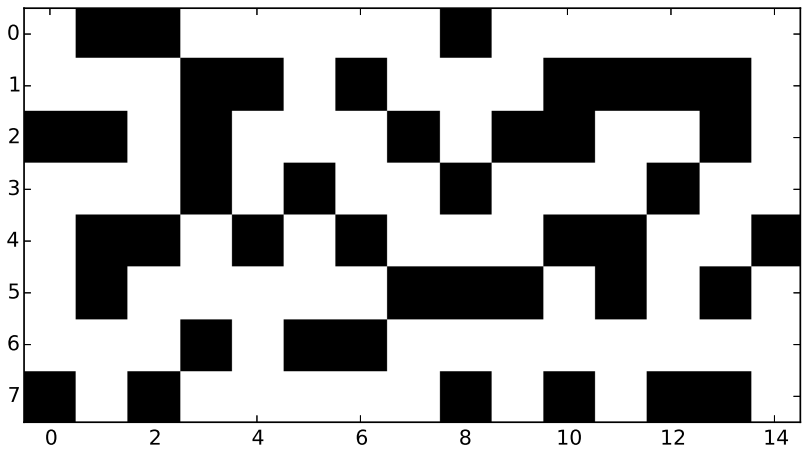

FIGURE 1 – Un labyrinthe parfait