Flood Fill

```python
from collections import deque

# Liste des mouvements possibles pour l'algorithme
deplacement = [
    (-1, 0),    # gauche
    (1, 0),     # droite
    (0, -1),    # haut
    (0, 1),     # bas
    (-1,-1),    # diagonale haut/gauche
    (-1, 1),    # diagonale bas/gauche
    (1, -1),    # diagonale haut/droite
    (1, 1)      # diagonale bas/droite
]

def pointValide(image, point, ancienneCouleur):
    """
    Un point est juge valide si :
    - Il est toujours dans les limites de notre tableau image
    - Il est de la couleur a modifier
    Les arguments de la fonction sont:
    - L'image sous forme de liste de Liste
    - Les coordonnees du point de depart de la diffusion
    - La couleur a remplacer
    """
    return 0 <= point[0] < len(image) and \
           0 <= point[1] < len(image[0]) and \
           image[point[0]][point[1]] == ancienneCouleur

def remplissageDiffusionIteratif(image, pointDepart, nouvelleCouleur):
    """
    Fonction de remplissage par diffusion iterative. Cette fonction utilise la
    methode de parcourt en largeur dans la theorie des graphes.
    Les arguments de la fonction sont:
    - L'image sous forme de liste de Liste
    - Les coordonnees du point de depart de la diffusion
    - La nouvelle couleur a utiliser
    """

    resteATraiter = deque()
    resteATraiter.append(pointDepart)

```

```
42      # Recuperation de la couleur au point origine
43      ancienneCouleur = image[pointDepart[0]][pointDepart[1]]
44
45      while resteATraiter:
46
47          pointCourant = resteATraiter.popleft()
48          image[pointCourant[0]][pointCourant[1]] = nouvelleCouleur
49
50          for depl in deplacement:
51              # Addition du point courant et d'un deplacement pour obtenir
52              # les coordonnees d'un point adjacent
53              pointVoisin = tuple(p+q for p, q in zip(pointCourant, depl))
54              if pointValide(image, pointVoisin, ancienneCouleur):
55                  resteATraiter.append((pointVoisin))
56
57  def remplissageDiffusionRecursif(image, pointDepart, nouvelleCouleur):
58      """
59       Fonction de remplissage par diffusion recursive. Cette fonction utilise la
60       methode de parcourt en profondeur dans la theorie des graphes.
61       Les arguments de la fonction sont:
62       - L'image sous forme de liste de Liste
63       - Les coordonnees du point de depart de la diffusion
64       - La nouvelle couleur a utiliser
65       """
66
67      # Recuperation de la couleur au point origine
68      ancienneCouleur = image[pointDepart[0]][pointDepart[1]]
69      image[pointDepart[0]][pointDepart[1]] = nouvelleCouleur
70
71      for depl in deplacement:
72          # Addition du point courant et d'un deplacement pour obtenir
73          # les coordonnees d'un point adjacent
74          pointVoisin = tuple(p+q for p, q in zip(pointDepart, depl))
75          if pointValide(image, pointVoisin, ancienneCouleur):
76              remplissageDiffusionRecursif(image, pointVoisin, nouvelleCouleur)
77
78
79
80  image = [
81          ['J', 'J', 'J', 'G', 'G', 'G', 'G', 'G', 'G', 'G'],
82          ['J', 'J', 'J', 'J', 'J', 'J', 'G', 'V', 'V', 'V'],
```

```
        ['G', 'G', 'G', 'G', 'G', 'G', 'G', 'V', 'V', 'V'],
        ['B', 'B', 'B', 'B', 'B', 'G', 'G', 'G', 'G', 'V'],
        ['B', 'R', 'R', 'R', 'R', 'R', 'G', 'V', 'V', 'V'],
        ['B', 'B', 'B', 'R', 'R', 'G', 'G', 'V', 'V', 'V'],
        ['B', 'M', 'B', 'R', 'R', 'R', 'R', 'R', 'R', 'V'],
        ['B', 'M', 'M', 'M', 'M', 'R', 'R', 'V', 'V', 'V'],
        ['B', 'M', 'M', 'V', 'M', 'M', 'M', 'M', 'V', 'V'],
        ['B', 'M', 'M', 'V', 'V', 'V', 'V', 'V', 'V', 'V']
]

pointDepart = (3, 9)
nouvelleCouleur = 'C'

print("-------------------------------------------------------")
for r in image:
    print(r)
print("-------------------------------------------------------")

#remplissageDiffusionIteratif(image, pointDepart, nouvelleCouleur)
remplissageDiffusionRecursif(image, pointDepart, nouvelleCouleur)

for r in image:
    print(r)

print("-------------------------------------------------------")

#pire des cas : tout le tableau est a remplir, complexite O(n^2)
```