```python
from doctest import testmod


def propager(M, i, j, val):
    """ Met tous les pixels de la composante de M[i][j]
    * à la valeur val si M[i][j] vaut 1 et
    * à 0 sinon.

    Args:
        M (list): matrice de l'image
        i (int): abscisse du pixel de référence
        j (int): ordonnée du pixel de référence
        val (int): valeur de seuil différente de 1

    Tests et Exemples:
    >>> M = [[0, 0, 1, 0], [0, 1, 0, 1], [1, 1, 1, 0], [0, 1, 1, 0]]
    >>> propager(M, 2, 1, 3)
    >>> M
    [[0, 0, 1, 0], [0, 3, 0, 1], [3, 3, 3, 0], [0, 3, 3, 0]]
    """
    # cas où il n'y a pas de composante
    # et donc rien à propager
    # et donc l'image M est inchangée
    if M[i][j] == 0:
        return

    M[i][j]=val

    # l'élément en haut fait partie de la composante
    # car il fait parti de l'image (le pixel ne déborde pas)
    # car sa valeur vaut 1
    if ( (i-1) >= 0 and M[i-1][j] == 1 ):
        propager(M, i-1, j, val)

    # l'élément en bas fait partie de la composante
    # car il ne déborde par de l'image
    # et car sa valeur vaut 1
    if ((i+1) < len(M) and M[i+1][j] == 1):
        # appel récursif à propager sur le pixel du bas
        propager(M, i+1, j, val)

    # l'élément à gauche fait partie de la composante
    # car il ne déborde par de l'image
    # et car sa valeur vaut 1
    if ((j-1) >= 0 and M[i][j-1] == 1):
        # appel récursif à propager sur le pixel de gauche
        propager(M, i, j-1, val)

    # l'élément à droite fait partie de la composante
    # car il ne déborde par de l'image
    # et car sa valeur vaut 1
    if ((j+1) < len(M) and M[i][j+1] == 1):
        # appel récursif à propager sur le pixel de droite
        propager(M, i, j+1, val)


M = [[0,0,1,0],
     [0,1,0,1],
     [1,1,1,0],
     [0,1,1,0]]

propager(M,2,1,3)
assert M == [[0, 0, 1, 0],
             [0, 3, 0, 1],
             [3, 3, 3, 0],
             [0, 3, 3, 0]]


testmod()
```