ISM MP, Informatique
Année 2025/2026

Corrigé du TP Informatique 16

Exercice 1

1. On modifie :

```
decrease=True
```

2. On complète :

```
# Initialisation de la partition
S=[[] for i in range(k)]
for pt in pop:
    tirage=rd.randint(0,k)
    S[tirage].append(pt)
```

3. On complète :

```
while decrease:

    # Affichage des k classes de points au cours de la boucle
    for i in range(k):
        tx,ty=[pt[0] for pt in S[i]],[pt[1] for pt in S[i]]
        plt.plot(tx,ty,'o')
    plt.show()

    # Calcul des centroïdes
    for i in range(k):
        if len(S[i])==0:
            centr[i]=pop[rd.randint(len(pop))]
        else:
            centr[i]=sum(S[i])/len(S[i])

    # Mise à jour de la partition
    S,V_aux=[[] for i in range(k)],0
    for pt in pop:
        dmin=float('inf')
        for i in range(k):
            d=alg.norm(pt-centr[i])
            if d<dmin:
                dmin,ind=d,i
        S[ind].append(pt);V_aux+=dmin**2

    # Gestion du booléen decrease
    if V_aux<V:
        V=V_aux
    else:
        decrease=False
```

On prévoit une affectation d'un poids au hasard sur un centroïde au cas où le cluster associé serait vide.

Exercice 2

1. On modifie :

```
decrease=True
```

2. On complète :

```
while decrease:

    # Calcul des centroïdes
    for q in range(k):
        if len(S[q])==0:
            centr[q]=np.array(tab[rd.randint(L),rd.randint(C),:])
        else:
            centr[q]=sum([tab[i,j,:] for i,j in S[q]])/len(S[q])


    # Mise à jour de la partition
    S=[[] for i in range(k)]
    V_aux=0
    for i in range(L):
        for j in range(C):
            pt=tab[i,j,:]
            dmin=float('inf')
            for q in range(k):
                d=alg.norm(pt-centr[q])
                if d<dmin:
                    dmin,ind=d,q
            S[ind].append([i,j])
            V_aux+=dmin**2

    # Gestion du booléen decrease
    if V_aux<V:
        V=V_aux
    else:
        decrease=False
```

3. Une fois que la classification de l'ensemble des pixels a été faite, on choisit pour chaque cluster son centroïde comme représentant et on affecte la couleur codée par un centroïde à tous les points d'un même cluster pour construire l'image « compressée ». En réalité, l'image ainsi construite pèse aussi lourd que l'image d'origine mais si on le souhaitait, on pourrait la stocker dans une structure ne codant que $\mathrm{L} \times \mathrm{C} \times k$ au lieu de $\mathrm{L} \times \mathrm{C} \times 2^{24}$ bits.