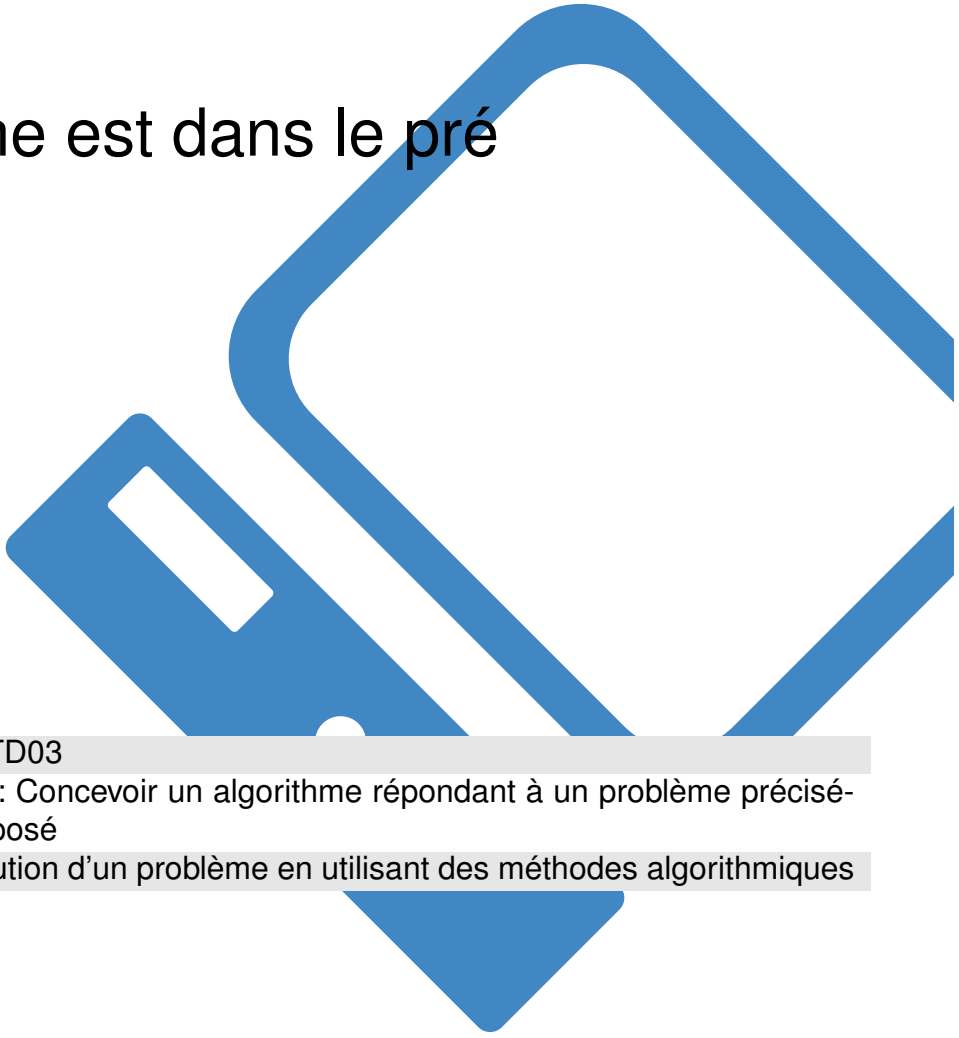
Séquence : 03

Document : TD03

Lycée Dorian
Juliette Genzmer
Willie Robert
Renaud Costadoat

Avec Correction

L'âne est dans le pré



Référence	S03- TD03
Compétences	Alt-C3: Concevoir un algorithme répondant à un problème précisément posé
Description	Résolution d'un problème en utilisant des méthodes algorithmiques

1 Présentation

Un paysan possède un âne pour lequel il réserve un pré de longueur $L = 20m$ et de largeur $l = 20m$. L'âne est attaché à un poteau par une corde de longueur $c = 2m$. De temps en temps, le propriétaire de l'âne est obligé de venir déplacer le poteau car l'âne a mangé toute l'herbe qu'il pouvait atteindre.

Il se rend compte que le choix de l'emplacement du poteau est important car parfois, après avoir effectué plusieurs déplacements, les zones où il reste de l'herbe sont éparpillées et de faible taille ce qu'il fait qu'il doit venir déplacer souvent l'âne.

Enfin, il est absolument interdit que l'âne ait la possibilité de manger l'herbe du pré voisin car le paysan n'en est pas le propriétaire.

Résumé des contraintes :

— L'âne ne doit pas aller en dehors de ce pré,
— Le propriétaire doit effectuer le minimum de déplacements de poteau,
— La surface d'herbe non mangée par l'âne ne doit pas excéder 15% de la surface du terrain.

Objectif : L'objectif de cette étude va être de proposer les positions des poteaux au paysan afin de respecter ces contraintes.

Le DM devra être rendu au format numérique, par mail ou sur une clef USB.

Dans un premier temps, nous allons suivre une grille afin de placer les poteaux.

2 Fonction $Manger$

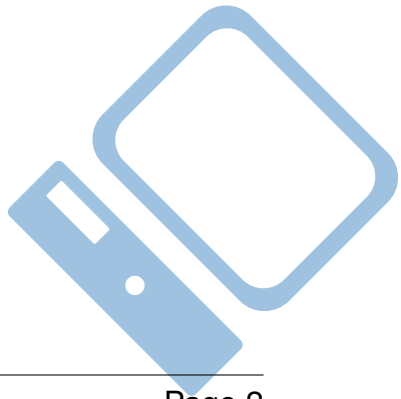
Soient (x, y), les coordonnées du poteau dans le repère $O(\overrightarrow{x}, \overrightarrow{y})$, liée au champ.
Soit (i, j) les coordonnées d'un point M du champ dans le repère $O(\overrightarrow{x}, \overrightarrow{y})$.

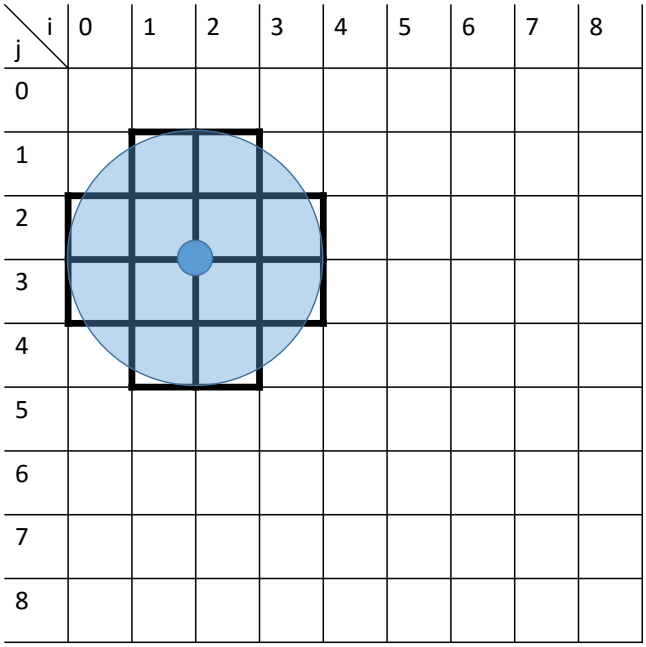
Question 1 : Donner l'inéquation vérifiée par i, j, x, y et c correspondant aux points $M(i, j)$ accessibles par l'âne lorsque le poteau est en (x, y) pour une corde de longueur c.

Pour simuler numériquement le problème, le champ va être découpé en petits carrés de $10cm$ de côté. Chaque carré a pour centre le point $M(i, j)$ et on considère que le carré est entièrement mangé si son centre $M(i, j)$ est accessible à l'âne.

La matrice `pre` va servir de map pour déterminer si une zone a été broutée ou pas. En effet, la composante `pre[i,j]` sera égale à :

— 0, si la zone n'a pas été parcourue par l'âne,
— 1, si la zone a été parcourue par l'âne une fois,
— 2, si la zone a été parcourue par l'âne plus d'une fois.





Question 2 : Définir une fonction `manger(x,y)` qui parcourt chacun des carrés de coordonnées (i, j), teste si il est accessible à l'âne et si il a déjà été mangé, et qui modifie la matrice `pre` . Les entrées de cette fonction sont les coordonnées (x, y) du poteau.

Question 3 : Proposer un algorithme qui fait parcourir au poteau le pré en commençant par la position $x_0 = 20dm$ et $y_0 = 20dm$ en se décalant à chaque fois d'un intervalle e et remplissant la matrice `pre`. On prendra pour commencer e égal à $35dm$ (on le fera varier par la suite).

3 Fonction *Analyse*

La fonction analyse va traiter la matrice `pre` générée précédemment. Si vous n'avez pas réussi à déterminer la matrice `pre` , le fichier `Pre.txt` , vous permet d'en générer une. Il est écrit sous la forme `i,j,pre[i,j] \n` . Il est disponible sur le forum.

Question 4 : Créer une fonction `analyse` qui en parcourant la liste `pre` , créer une liste :

- `x1` qui renvoie l'ensemble des i parcourus dans le sens des i croissants puis des j croissants lorsque `pre[i,j]=1` ,
- `y1` qui renvoie l'ensemble des j parcourus dans le sens des i croissants puis des j croissants lorsque `pre[i,j]=1` ,
- `x2` qui renvoie l'ensemble des i parcourus dans le sens des i croissants puis des j croissants lorsque `pre[i,j]=2` ,
- `y2` qui renvoie l'ensemble des j parcourus dans le sens des i croissants puis des j croissants lorsque `pre[i,j]=2` .

Cette fonction devra aussi retourner le nombre de 1 dans la matrice `pre` et le nombre de 2 dans la matrice `pre` .

4 Fonction $Pourcent$

Question 5 : Créer une fonction `pourcent` qui pour une entrée k (représentant le nombre de 1 plus le nombre de 2 dans la matrice `pre`) retourne le pourcentage que ce nombre k représentante par rapport à l'ensemble des éléments de la matrice `pre`

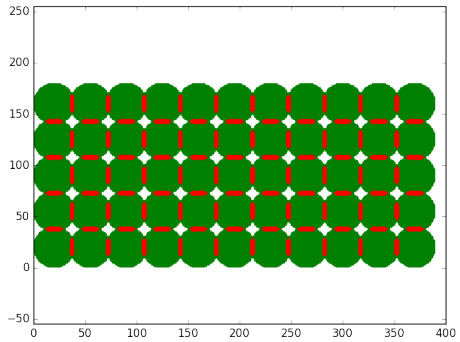
Question 6 : Quel résultat vous donne la fonction `pourcent` , ce résultat respecte-t-il le cahier des charges ?

5 Tracé de la réponse et optimisation

Le code suivant va vous permettre de tracer le résultat obtenu.

```
import matplotlib.pyplot as plt

plt.scatter(x1,y1,s=1,color='green')
plt.scatter(x2,y2,s=1,color='red')
plt.axis('equal')
plt.xlim(0, long)
plt.ylim(0, larg)
plt.show()
```



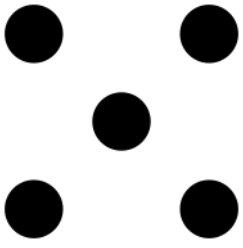
Sur cette figure, on trouve en vert l'herbe broutée une seule fois, en rouge, celle broutée deux fois et en blanc celle qui n'a pas été broutée.

Question 7 : Modifier la valeur de e afin de respecter le cahier des charges.

Question 8 : Donner le pourcentage de zones « rouges »par rapport à la surface du pré avec cette nouvelle valeur de e.

6 Nouvelle solution (facultatif)

Dans cette solution proposer une disposition en quinconce de la position du poteau et tenter d'améliorer les performances précédentes.



Question 9 : Donner les nouvelles valeurs trouvées pour les questions 6 et 8. Que concluez-vous quant à cette simulation.

1 Correction

Question 1 : $(i-x)^2+(j-x)^2 \leq c^2$

Question 2 :

```
def manger(x,y):
    for i in range(long):
        for j in range(larg):
            if (i-x)**2+(j-y)**2<corde**2 and pre[i,j]!=0:
                pre[i,j]=2
            if (i-x)**2+(j-y)**2<corde**2 and pre[i,j]==0:
                pre[i,j]=1
    return pre
```

Question 3 :

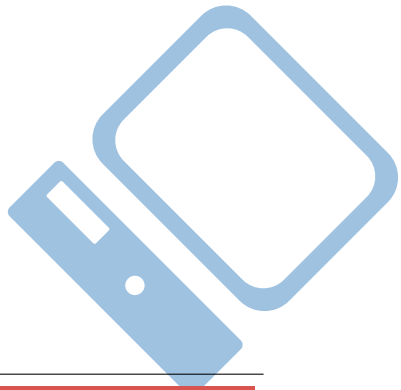
```
def calcul_classique(x0,y0,inc):
    x=x0
    y=y0
    while y+corde < larg:
        while x+corde < long:
            pre=manger(x,y)
            x=x+inc
        x=x0
        y=y+inc
    return pre
```

Question 4 :

```
def analyse(pre):
    k,h=0,0
    x1,y1,x2,y2 = [],[],[],[]
    for i in range(long):
        for j in range(larg):
            if pre[i,j] == 1:
                k=k+1
                x1.append(i)
                y1.append(j)
            if pre[i,j] == 2:
                h=h+1
                x2.append(i)
                y2.append(j)
    return x1,y1,x2,y2,k,h
```

Question 5 :

```
def pourcent(k):
    return 100*(k)/(long*larg)
```



Question 6 : La réponse est 78%, cela ne respecte pas le cahier des charges.

Question 7 : La valeur de $e = 29$ convient.

Question 8 : Pour $e = 29$, on obtient :
— Surface couverte 86%,
— Surface couverte 2 fois 34%.
Cela respecte le cahier des charges.

Question 9 :

```
def calcul_quinconce(x0,y0,inc):
    u=0
    x=x0
    y=y0
    while y+corde+inc/2 < larg:
        while x+corde < long:
            u=u+1
            if (u % 2 == 0):
                pre=manger(x,y+inc/2)
            else:
                pre=manger(x,y)
            x=x+inc
        x=x0
        u=0
        y=y+inc
    return pre
```

Ici le résultat obtenu pour $e = 34.9$ est :
— Surface couverte 81%,
— Surface couverte 2 fois 4%.
Même si la zone « rouge »a diminué, le cahier des charges pourra difficilement être respecté avec cette simulation.