ENSAE TD noté, mardi 22 octobre 2019

Le programme devra être envoyé par mail au chargé de TD et au professeur. Toutes les questions valent 2 points.

1

1) C'est un mariage dans une salle ronde. On veut disposer les tables de sortes qu'elles soient éloignées le plus possible les unes des autres et du bord de la salle. Les tables ont toutes le même rayon. Ecrire une fonction qui calcule la distance entre deux tables rondes dont on connaît le centre. Et comme ce sont des tables rondes, on considère que la distance euclidienne entre leurs centres. **Fonction suggérée :** `def distance_table(x1,y1,x2,y2)` :

2) Ecrire une fonction qui calcule la distance entre une table (son centre) et le bord de la salle de rayon R. **Fonction suggérée :** `def distance_bord(x1,y1,R)` :

3) Ecrire une fonction qui tire aléatoirement une table dans le cercle de rayon R. **Fonction suggérée :** `def table_alea(R)` :

4) Ecrire une fonction qui tire aléatoirement N tables dans le cercle de rayon R. **Fonction suggérée :** `def n_table_alea(N,R)` :

5) Ecrire une fonction qui retourne la table la plus proche d'une table ou du bord. La fonction doit retourner l'indice de la table la plus proche ou -1 si c'est le bord, puis la distance associée. On ajoute un paramètre *skip_i* pour éviter une table. **Fonction suggérée :** `def table_proches(x1,y1,list_tables,R,skip_i)` :

6) Ecrire une fonction qui tire N tables aléatoirement dans la salle ronde et qui retourne la distance minimum entre deux tables ou le mur et les tables. **Fonction suggérée :** `def distance_table_alea(N,R)` :

7) Ecrire une fonction qui appelle k fois la fonction précédente et qui retourne la meilleure distribution aléatoire des tables. **Fonction suggérée :** `def meilleur_table_alea(k,N,R)` :

8) Ecrire une fonction qui retourne le résultat pour 11 tables et une salle de diamètre 1.

9) Ecrire une fonction qui représente la solution avec matplotlib en partant de l'exemple donné. **Fonction suggérée :** `def plot_tables(R,list_tables)` :

```
import matplotlib.pyplot as plt
from matplotlib.patches import Circle
fig, ax = plt.subplots(1, 1, figsize=(4, 4))
ax.set_xlim([-1, 1])
ax.set_ylim([-1, 1])
ax.add_artist(Circle((0, 0), 1, alpha=0.2))
ax.plot([-0.5, 0.2], [-0.2, 0.3], 'o');
```

10) Imaginer un algorithme qui soit meilleur que le hasard pour répartir les tables... Le quadrillage idéal serait sans doute que toutes les tables forment un quadrillage fait de triangles équilatéraux. Avez-vous déjà entendu parler des diagrammes de Voronoï ?

ENSAE TD noté, mardi 22 octobre 2019

Le programme devra être envoyé par mail au chargé de TD et au professeur. Toutes les questions valent 2 points.

2

1) C'est un mariage dans une salle carrée. On veut disposer les tables de sortes qu'elles soient éloignées le plus possible les unes des autres et du bord de la salle. Les tables sont toutes carrées et ont toutes la même taille. Ecrire une fonction qui calcule la distance entre deux tables carrées dont on connaît le centre. Et comme ce sont des tables carrées, on considère que la distance entre deux tables et la plus grande des valeurs absolues des différences de coordonnées des centres. **Fonction suggérée :** `def distance_table(x1, y1, x2, y2) :`

2) Ecrire une fonction qui calcule la distance entre une table (son centre) et le bord de la salle de côté $2C$ et dont le centre est placé aux coordonnées *(0, 0)*. La fonction retourne une valeur négative si la table n'est pas dans la salle. Le bord le plus proche est celui qui est horizontalement ou verticalement le plus proche. **Fonction suggérée :** `def distance_bord(x1, y1, C) :`

3) Ecrire une fonction qui tire aléatoirement une table dans le carré de côté $2C$. **Fonction suggérée :** `def table_alea(C) :`

4) Ecrire une fonction qui tire aléatoirement N tables dans le carré de côté $2C$. **Fonction suggérée :** `def n_table_alea(N, C) :`

5) Ecrire une fonction qui retourne la table la plus proche d'une table ou du bord. La fonction doit retourner l'indice de la table la plus proche ou -1 si c'est le bord, puis la distance associée. On ajoute un paramètre *skip_i* pour éviter une table. **Fonction suggérée :** `def table_proches(x1, y1, list_tables, C, skip_i) :`

6) Ecrire une fonction qui tire N tables aléatoirement et qui retourne la distance minimum entre deux tables ou le mur et les tables. **Fonction suggérée :** `def distance_n_tables_alea(N, C) :`

7) Ecrire une fonction qui appelle k fois la fonction précédente et qui retourne la meilleure distribution aléatoire des tables. **Fonction suggérée :** `def meilleur_table_alea(k, N, C) :`

8) Ecrire une fonction qui retourne le résultat pour 11 tables et une salle de diamètre 1 et 10 essais.

9) Ecrire une fonction qui représente la solution avec matplotlib en partant de l'exemple donné. **Fonction suggérée :** `def plot_tables(C, list_tables) :`

```
import matplotlib.pyplot as plt
from matplotlib.patches import Rectangle
fig, ax = plt.subplots(1, 1, figsize=(4, 4))
ax.set_xlim([-1, 1])
ax.set_ylim([-1, 1])
ax.add_artist(Rectangle((-1, -1), 2, 2, alpha=0.2))
ax.plot([-0.5, 0.2], [-0.2, 0.3], 'o');
```

10) Imaginer un algorithme qui soit meilleur que le hasard pour répartir les tables... Le quadrillage idéal serait sans doute que toutes les tables forment un quadrillage fait de triangles rectangles. Avez-vous déjà entendu parler des diagrammes de Voronoï ?