I Procédure de rendu

Une seule manière de rendre ce DM : se connecter au site de la classe et déposer votre fichier en cliquant à l'endroit adéquat.
Je ne prendrai pas en compte les fichiers envoyés par mail. Par contre, vous pouvez tout à fait poser vos questions sur le site ou par mail.

Tester vos fonctions

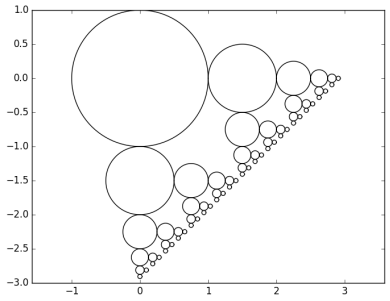
Il y a des tests gentiment fournis pour beaucoup de vos fonctions. En exécutant votre script via F5, vous aurez accès dans la console à la fonction `test_fonctions()` qui ne prend pas d'argument mais exécute des tests sur le comportement de vos fonctions. Utilisez-la !

Un rendu propre

Assurez-vous que votre feuille de script supporte une exécution via F5 avant de l'envoyer.

II Graphique

Dans ce premier exercice, le but est de reproduire la figure suivante :



Pour cela vous disposez d'une fonction `cercle(x, y, r)` qui trace un cercle dans la figure courante, centré au point de coordonnées $\begin{pmatrix} x \\ y \end{pmatrix}$ et de rayon r.

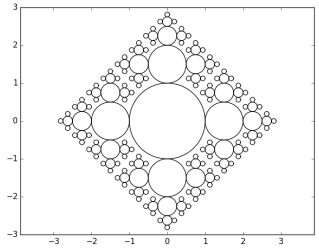
La fonction principale est

```
def bulles(n):
    plt.figure()
    plt.axis("equal")
    bulles_aux(0, 0, n, 1)
    plt.show()
```

et votre travail consiste à implémenter la fonction récursive `bulles_aux(x, y, n, r)` où n représente le nombre d'appel restant à effectuer. Pour réussir à implémenter la fonction `bulles_aux`, il vous faut tout d'abord réussir à décrire récursivement la figure, comme vu en TD.

Bonus

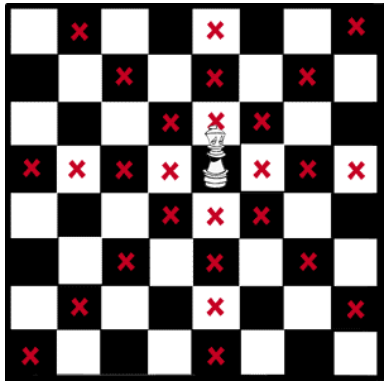
En question optionnelle, et si le graphique précédent était trop facile à obtenir, essayez avec :



Indication : votre fonction auxiliaire devra prendre en plus un paramètre représentant la direction de la provenance de l'appel.

III Echec à la dame

Aux échec une dame peut se déplacer de la manière suivante :



Toute pièce sur une des cases marquées en rouge est susceptible d'être "prise". La question abordée ici est de placer le maximum de dames sur un même échiquier de taille n de telle manière qu'aucune ne peut être prise par une autre.

Il se trouve que l'on peut toujours en placer n et pas plus (il est facile de montrer qu'on ne peut pas en placer plus). Ainsi nous allons essayer de placer une dame par ligne de l'échiquier, de toutes les manières répondant à notre problème : aucune ne peut être prise par une autre.

3.1 Créer et afficher un échiquier

On représente un échiquier par un tableau (liste de listes) de dimension $n \times n$ contenant des booléens (vrai si on a placé une reine à cet endroit, faux sinon).

Etape 1 Remplissez la fonction **echiquier_vide(n)**

Etape 2 : visualiser Remplissez la fonction **imprimer(T)** qui prend un échiquier comme argument et affiche (pas de valeur de retour) celui-ci. Le but est d'obtenir

```
>> T = echiquier_vide(4)
>> T[0][0] = True
>> T[2][3] = True
>> imprimer(T)
oxxx
xxxx
xxxo
xxxx

>>
```

Remarquez la ligne vide affichée en plus. Informatiquement un retour chariot (passage à la ligne, comme un appui sur Entrée) est représenté par le caractère "\n".

3.2 Se déplacer dans l'échiquier

Etape 3 Notre algorithme va consister à placer les reines une par une en testant toutes les cases de l'échiquier. On le parcourt de la gauche vers la droite puis du haut en bas.
Remplir la fonction **case_suivante(n, i, j)**

Etape 4 On considère maintenant une case d'indices i, j et un échiquier T. Le but des prochaines fonctions est de savoir si la case examinée est dans la ligne de mire d'une reine déjà présente.
Vu notre algorithme, une reine déjà présente est forcément sur une ligne au dessus (d'indice $\leqslant i$) ou à gauche.
Remplir les 4 fonctions **prise_...(i, j T)**. Tracer quelques schémas au brouillon pour comprendre les exemples de test et tester d'autres cas peut-être une bonne idée.
Pour jouer avec les booléens, on se souviendra des opérations **or** et **and** qui ont la même signification qu'en mathématiques.

```
>> b1 = True
>> b2 = False
>> b1 or b2
True
>> b1 and b2
False
>> b1 or b1
True
```

Compléter enfin la fonction **en_prise(i, j, T)**.

3.3 Les solutions

Etape finale Complétez la fonction récursive auxiliaire **aux(i, j, m)**.
Le principe de notre algorithme est le suivant :

- on part d'un échiquier vide.
- Pour chaque case on compte (récursivement) le nombre de solutions en plaçant ou non une dame à cet endroit. Plus précisément, le nombre de solutions à notre problème est la somme des nombres de solution en plaçant une dame dans la case courante et du nombre de solutions sans placer de dame dans la case courante.

Avec les notations de la feuille de script, il y a deux cas à considérer :

1. si la case i, j est en prise un seul appel récursif doit suffire
2. sinon il nous en faut 2, suivant qu'on place une reine en position i, j ou non.

Dernière indication : attention à la signification du paramètre m.