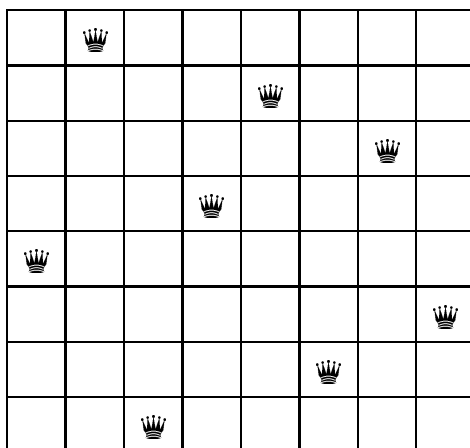


## Le problème des $n$ reines

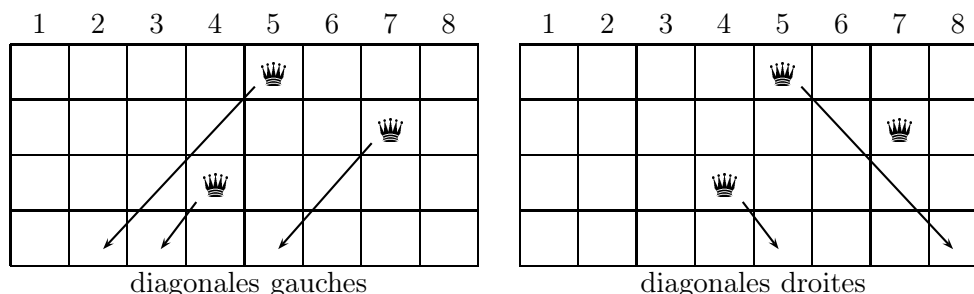
Le problème des  $n$  reines consiste à placer  $n$  reines sur un échiquier de taille  $n \times n$  de telle sorte qu'aucune ne soit en prise : il ne faut donc pas plus d'une reine par ligne, par colonne et par diagonale. Voici un exemple pour  $n = 8$ .



Le but de cet exercice est d'écrire un programme en Python qui permette de calculer le nombre de manières différentes de résoudre ce problème pour un nombre  $n$  quelconque de reines. Pour cela, nous allons utiliser un algorithme de recherche avec retour arrière (ou *backtracking* en anglais) qui va remplir les lignes de l'échiquier une à une. Afin de remplir une ligne, l'algorithme maintient 3 ensembles :

- $a$  : contient les numéros des colonnes où il n'y a encore aucune reine de placée ;
- $b$  : contient les numéros des colonnes sur lesquelles il n'est pas possible de placer une reine car ces positions se trouvent sur la diagonale *gauche* d'une reine déjà placée ;
- $c$  : joue le même rôle que  $b$  mais pour les diagonales *droites*.

Par exemple, après avoir placé les 3 premières reines sur les colonnes 5, 7 et 4 (dans cet ordre) d'un échiquier  $8 \times 8$ , on a la situation suivante



qui est représentée par les trois ensembles  $a = \{1, 2, 3, 6, 8\}$ ,  $b = \{2, 3, 5\}$  et  $c = \{5, 8\}$ . L'ensemble des colonnes où il est encore possible de placer une reine pour la 4<sup>e</sup> ligne est donc tout simplement  $(a \setminus b) \setminus c$ , soit ici  $\{1, 6\}$ . L'algorithme consiste alors à essayer *une à une* ces positions : récursivement, on cherche les solutions pour lesquelles la 4<sup>e</sup> reine est placée sur la colonne 1 puis, en revenant en arrière, récursivement celles pour lesquelles

la reine est placée sur la colonne 6. À chaque appel récursif, les ensembles  $a$ ,  $b$  et  $c$  sont modifiés en fonction de la colonne  $i$  choisie :  $i$  est supprimée de  $a$  et ajoutée aux ensembles  $b$  et  $c$ ; ces ensembles  $b$  et  $c$  sont alors mis à jour en décrémentant (resp. incrémentant) les éléments qu'ils contiennent afin de calculer les nouvelles diagonales des reines placées sur l'échiquier.

Dans la suite, nous allons simplement représenter les ensembles  $a$ ,  $b$  et  $c$  par des listes d'entiers.

**Question 1** Écrire une fonction `list_of_int` telle que `list_of_int(n)` renvoie la liste  $[1, 2, \dots, n]$  si  $n$  est positif et la liste vide sinon.

**Question 2** En utilisant des itérateurs sur les listes, écrire les fonctions `succ_list` et `pred_list` telles que `succ_list([a1, a2, ..., an])` renvoie la liste  $[a1+1, a2+1, \dots, an+1]$  et `pred_list([a1, a2, ..., an])` renvoie la liste  $[a1-1, a2-1, \dots, an-1]$ .

**Question 3** En utilisant un itérateur sur les listes, écrire la fonction `diff` telle que `diff(l1, l2)` renvoie une liste contenant les éléments de  $l1$  qui ne sont pas éléments de  $l2$ .

**Question 4** En utilisant un itérateur sur les listes, écrire une fonction `remove` telle que `remove(l, x)` renvoie une liste contenant les éléments de  $l$  sauf  $x$ .

La question suivante est l'algorithme principal du problème des  $n$  reines. Il s'agit d'écrire une fonction récursive prenant en arguments les 3 ensembles  $a$ ,  $b$  et  $c$  décrits précédemment. Cette fonction renvoie le nombre de solutions qui prolongent la solution partielle décrite par  $a$ ,  $b$  et  $c$ . En particulier, elle renvoie l'entier 1 quand l'ensemble  $a$  est vide (puisque'il n'y a plus de reines à placer c'est qu'une solution a été trouvée).

**Question 5** À l'aide des fonctions précédentes, et en utilisant un itérateur sur les listes, écrire une fonction récursive `nb_solutions` telle que `nb_solutions(a, b, c)` renvoie le nombre de solutions correspondant aux ensembles  $a$ ,  $b$  et  $c$  donnés en paramètres.

Enfin, la question suivante consiste à écrire la fonction principale du programme qui se contente d'appeler la fonction `nb_solutions` avec comme paramètres les ensembles  $a = \{1, 2, \dots, n\}$  et  $b = c = \emptyset$ .

**Question 6** Écrire une fonction `reines` telle que `reines(n)` renvoie le nombre de solutions au problème des  $n$  reines.