Problème des reines sur l'échiquier [hs04] - Exercices

Karine Zampieri, Stéphane Rivière

Unisciel algoprog Version 21 mai 2018

Table des matières

C++ - Problème des reines sur l'échiquier (TP)

Mots-Clés Backtraking, Heuristique ■
Requis Axiomatique objet, Récursivité des actions, Heuristique ■
Difficulté ●●○ (4 h) ■

Objectif
Cet exercice résout le problème des reines sur l'échiquier : résolution récursive, recherche particulière et heuristique des reines.

Remarque
Dans le même ordre d'idées, l'exercice @[Problèmes du parcours du cavalier] résout celui du cavalier.

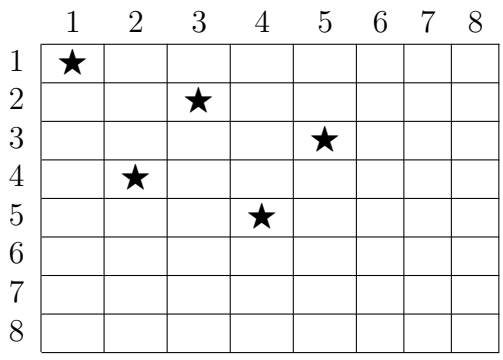
...(énoncé page suivante)...

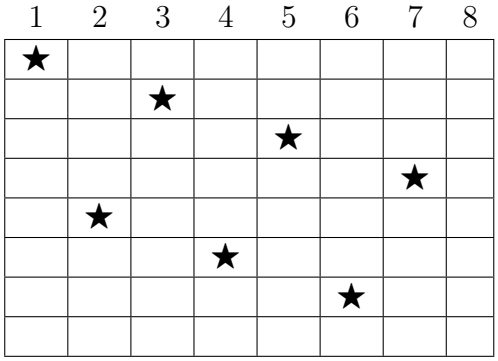
1 Le problème des reines

Voici un autre casse-tête pour les mordus des échecs : il consiste à placer n reines sur un échiquier $n \times n$ sans qu'elles ne se prennent l'une l'autre. Une reine peut prendre toutes les pièces se trouvant sur la même ligne, sur la même colonne ou sur les mêmes diagonales qu'elle-même.

Exemple

Sur la figure de gauche, cinq reines sont placées mais la sixième est bloquée. Sur la partie droite, sept reines sont placées mais la huitième est bloquée.





Historique

Ce problème fut posé pour la première fois en 1848 par Max Bessel dans un journal d'échecs. Cette publication donna lieu à un engouement extraordinaire. Le 21 septembre 1850, le Dr. Nauck donna pour $n = 8$ toutes les 92 solutions alors que le mathématicien Gauss n'en trouva que 72. D'après Kraitchick, le nombre s de solutions en fonction de l'ordre n de l'échiquier est :

n	1	2	3	4	5	6	7	8	9	10	11	12
s	1	0	0	2	10	4	40	92	352	720	2680	14200

...(suite page suivante)...

2 Algorithmique, Programmation

2.1 Représentation du problème

Structures de données

Deux reines ne pouvant être sur la même ligne et puisqu'il faut placer n reines et qu'il y a n lignes, il y aura une reine par ligne. De même, il y a une reine par colonne. Ainsi si les reines sont placées sur les lignes 1 à n, il suffit d'un tableau unidimensionnel pour mémoriser l'état de l'échiquier d'où :

- « La reine k » est une abréviation de « la reine qui est sur la ligne k »
- On peut modéliser l'échiquier par un vecteur.

Définissez :

- Le type `Plateau` comme étant un `Vecteur` d'entiers.
- Le type `Solutions` comme une `Liste` de `Plateau`.

Outil C++

Le modèle de classes `vector<T>` est défini dans la bibliothèque `<vector>` et celui de `list<T>` dans la bibliothèque `<list>`.

Test de case libre

Pour savoir si une case est libre il faut regarder si la colonne et les diagonales qui y passent sont libres. Si $R = r[k]_{1..n}$ est le vecteur qui indique les numéros des colonnes des reines :

- Alors $r[k] = 0$ si la k-ème reine est libre.
- Sinon $r[k]$ donne l'indice de la colonne où elle est posée.

Les diagonales sont caractérisées par le fait que la somme ou la différence des numéros de ligne et de colonne sont constantes. Ainsi si i désigne la colonne de la reine à placer et $j = 1, 2, \ldots, i-1$ une des colonnes précédentes alors :

- Deux reines sont sur une même ligne si $r[i] = r[j]$.
- Deux reines sont sur une même diagonale si le coefficient angulaire de la droite qui les relie est ± 1 (angle d'une diagonale d'un carré de 45°).

Ces deux conditions s'écrivent (les reines des colonnes i et j se menacent) :

$$\frac{r[i] - r[j]}{i - j} = \pm 1$$

ce qui s'écrit aussi :

$$|r[i] - r[j]| = |i - j|$$

Écrivez une fonction `issafe(cb,k)` qui, sur un `Plateau cb|, teste et renvoie \lstinline`Vrai@ s'il n'y a pas de conflit de mettre la reine en colonne `k`, ligne `cb[k]` avec toutes les `(k-1)`-ème reines déjà placées.

2.2 Résolution récursive

Dans ce problème la complexité n'est pas à craindre car le plus mauvais des programmes récursifs est encore raisonnablement rapide.

Stratégie récursive
Elle s'énonce : « Sachant qu'il y a des reines bien placées dans les lignes 1 à $k-1$ et connaissant leur place, trouver toutes les solutions possibles en plaçant convenablement les reines de k à n. »

Algorithme de placement
L'algorithme `placerRN(k)` qui tente de placer la `k`-ème reine est alors :

- Si `k=n` alors écrire la solution obtenue.
- Sinon Parcourir toutes les cases de la ligne `k` :
  - Si une case est libre (c.-à-d. non condamnée par les reines déjà en place) alors :
    - Y mettre la reine `k` puis
    - Résoudre le problème pour `k+1`.
  - Libérer la reine `k`.

Écrivez la procédure `placerRN(cb,sols,k)` qui réalise l'algorithme de placement de la reine `k` sur un `Plateau cb` comme décrit ci-dessus. Dans le cas d'une solution, elle l'enregistre dans `Solutions sols`.

Déduisez une procédure `dfsRN(cb,sols)` qui réalise la résolution récursive sur un `Plateau cb` et mémorise toutes les solutions dans `Solutions sols`.

Écrivez une procédure `afficherSolutions(sols)` qui affiche les `Solutions sols`.

Écrivez une procédure `test_dfsRN` qui saisit un entier représentant l'ordre de l'échiquier, instancie un échiquier de problème des reines, puis cherche et affiche l'ensemble des solutions au problème du placement des reines.

Testez. Exemple d'exécution :

```
Dimension n de l'echiquier? 8
Il y a 92 solutions. On en affiche? 5
Sol #  1: [0 4 7 5 2 6 1 3 ]
Sol #  2: [0 5 7 2 6 3 1 4 ]
Sol #  3: [0 6 3 5 7 1 4 2 ]
Sol #  4: [0 6 4 7 1 3 5 2 ]
Sol #  5: [1 3 5 7 2 0 6 4 ]
```

2.3 Résolution à partir d'une position

Les solutions ne sont pas toutes distinctes car l'échiquier possède des symétries. Ainsi pour optimiser le programme récursif, on ne cherchera que les solutions principales au problème des reines c.-à-d. celles à partir desquelles elles peuvent toutes être trouvées en utilisant ces symétries (Sont considérées comme équivalentes des solutions déduites l'une de l'autre, par symétrie par rapport aux droites joignant les milieux de l'échiquier ou par rotation de l'échiquier).

La procédure actuelle fait parcourir à la première reine toute la première ligne. Or il suffit d'en parcourir la moitié et de compléter par symétrie. Cependant il n'est pas facile d'arrêter le parcours lorsque $r[1]$ a atteint la valeur $\frac{n}{2}$ mais il est facile de commencer le parcours à $\frac{n}{2}+1$.

Copiez/collez la procédure `placerRN(cb,sols,k)` en la procédure `placerPosRN(cb,sols,k)`. Modifiez-la de sorte à lancer la recherche à partir de la ligne 0.

De même **copiez/collez** la procédure `dfsRN(cb,sols)` en la procédure `dfsPosRN(cb,sols,pos)`. Modifiez-la de sorte à placer la première reine en la ligne numéro `pos`.

Écrivez une procédure `dessinerEchiquier(cb)` qui « dessine » un `Plateau cb` des reines posées en utilisant les symboles `'Q'` ou `'-'`.

Déduisez une procédure `dessinerSolutions(sols)` qui « dessine » les `Solutions sols`.

Copiez/collez la procédure `test_dfsRN` en la procédure `test_dfsPosRN` puis complétez-la et modifiez-la de sorte à utiliser les procédures de ce problème.

Testez. Exemple d'exécution :

```
Dimension n de l'echiquier? 5
# de ligne de la reine en colonne 0? 3
Il y a 2 solutions
   0 1 2 3 4
0  - Q - - -
1  - - - - Q
2  - - Q - -
3  Q - - - -
4  - - - Q -

   0 1 2 3 4
0  - - - - Q
1  - Q - - -
2  - - - Q -
3  Q - - - -
4  - - Q - -
```

2.4 Blocage de l'échiquier des reines

Ce problème cherche quel est le plus petit nombre de reines nécessaire pour bloquer l'échiquier de sorte qu'il n'est plus possible de placer une pièce sur une case sans qu'elle ne soit prise par une reine, c.-à-d. « Comment bloquer l'échiquier avec le minimum de reines ? ». **Attention** Avec n reines il y avait une reine par ligne et une par colonne. Ici ceci n'est plus vrai.

Donnez les bornes inférieure et supérieure pour le nombre de reines nécessaire pour bloquer l'échiquier standard (taille 8×8).

Aide simple
Le nombre x de reines n'étant pas donné il faut essayer des x croissants ou décroissants. Il faut trouver un borne inférieure pour le nombre de reines. Sur l'échiquier standard vide une reine peut bloquer 28 cases : il faut donc au moins trois reines pour bloquer l'échiquier. Il en faut au plus sept car l'échiquier est souvent bloqué avant d'avoir pu mettre la 8-ème reine ; il est même probable que six suffissent. Il faut explorer la tranche de trois à six reines (cas de l'échiquier d'ordre 8).

Proposez une stratégie du blocage de l'échiquier.

Aide simple
Voici une idée. Placez k reines puis parcourez l'échiquier à la recherche d'une case libre. S'il n'y en a pas, affichez la solution. S'il y en a une, déplacez une reine et recommencez.

2.5 Heuristique des n reines

Ce problème tente l'heuristique d'accessibilité pour résoudre le problème des reines.

En vous inspirant de l'heuristique du problème du cavalier @[Heuristique du cavalier d'Euler], formulez une heuristique pour ce problème.

Pourquoi la stratégie suivante est-elle intéressante ? « Pour chaque case, on inscrit le nombre de cases éliminées dans le restant de l'échiquier lorsqu'une reine y est placée (par ex. sur l'échiquier standard, un coin possède la valeur 22). On place alors une reine dans la case qui **supprime le moins** de possibilités pour le placement des autres reines. »

Écrivez les définitions du delta de déplacements ainsi que l'opération `+| sur les \lstinline`Position|.

Écrivez une procédure `calculerHB(cb,hb,delta)` qui calcule le tableau des accessibilités `Echiquier hb` selon les `CVDeplacements delta` d'un `Echiquier cb`.

Écrivez une procédure `placerNaif(cb,hb,sols,pos0)` qui lance la pose des reines à partir d'une `Position pos0`. La recherche de la prochaine position de pose s'effectue en parcourant systématiquement tout l'`Echiquier hb` des accessibilités. En cas de solution, celle-ci est enregistrée dans les `Solutions sols`.

Écrivez une procédure `test_heuristiq` qui lance la recherche sur chacune des positions.

Testez. Exemple d'exécution :

```
Dimension n de l'echiquier?  12 12 12 12 12
 12 14 14 14 12
 12 14 16 14 12
 12 14 14 14 12
 12 12 12 12 12

Solution issue de (1,1)
[(1,1) (3,0) (4,2) (0,4) (2,3) ]
Solution issue de (2,2)
[(2,2) (1,0) (4,1) (0,3) (3,4) ]
Solution issue de (4,4)
[(4,4) (1,0) (0,2) (3,1) (2,3) ]
```

De même, écrivez une procédure `placerOptm(cb,hb,sols,pos0)` qui réalise une version améliorée de la recherche de pose en utilisant une liste de toutes les positions encore disponibles.

Testez en appelant cette procédure dans `test_heuristiq`.

2.6 Force brute des reines

Ce problème tente différentes approches par force brute pour résoudre le problème des reines.

Trouvez des solutions au problème des n reines à l'aide de l'approche par force brute aléatoire (voir l'approche par force brute dans le cas du cavalier).

Développez une approche exhaustive en traitant toutes les combinaisons possibles des n reines sur l'échiquier.

Comparez et faites ressortir les différences entre les approches par force brute aléatoire et exhaustive.