

Programmation Fonctionnelle Avancée

Cours 11

kn@lri.fr

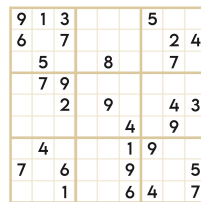
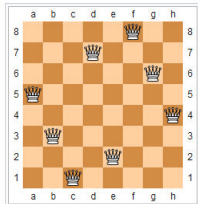
<http://www.lri.fr/~kn>

Problèmes de satisfaction



De nombreux problèmes algorithmiques peuvent être vus comme de problèmes de recherche d'une « solution » parmi un espace de valeur potentiellement grand

- ◆ Placer N reines sur un échiquier
- ◆ Compléter une grille de sudoku
- ◆ Trouver tous les mots sur une grille de Boggle
- ◆ Résoudre le problème du « compte est bon »
- ◆ Résoudre un casse tête (comme Rush Hour™)



Trop de solutions ?



Pour tout ces problèmes, énumérer de façon exhaustive toutes les configurations possibles, puis vérifier pour chaque configuration si celle-ci est une solution est trop coûteux

On présente une méthode systématique qui permet, dans de nombreux cas, de trouver une solution « rapidement ».

Pour chaque problème précis, on peut ensuite optimiser cette solution systématique.

Retour arrière



La technique de recherche avec retour arrière (*backtracking*) consiste à partir d'un du problème initial et à s'en rapprocher récursivement (en faisant diminuer une certaine mesure, ce qui va assurer que l'on termine).

Pour se « rapprocher », on essaye toutes les façons d'avancer « en un coup » vers la solution

Pour chacune d'elle, si c'est un morceaux de solution valide, alors on s'appelle récursivement sur la nouvelle configuration, sinon on passe à la suivante sans explorer plus ce début de solution.

Retour arrière (2)



La recherche avec retour arrière consiste à parcourir l'arbre de toutes les configurations possibles en profondeur. Cependant, lorsque l'on détecte que la configuration courante ne peut déjà pas mener à une solution, on évite de parcourir le sous-arbre correspondant.

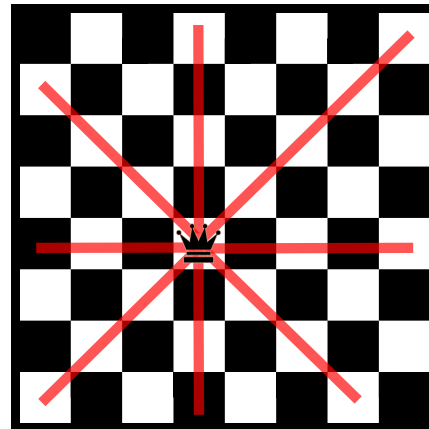
On illustre cette technique avec deux problèmes différents

Étude de cas 1 : problème des N reines



Étant donné un échiquier de taille $N \times N$, comment placer dessus N reines de façon à ce qu'aucune ne soit en prise.

Rappel: aux échecs, une reine peut capturer à n'importe quelle distance en ligne, en colonne et en diagonale.



Méthode générale



On utilise l'approche récursive suivante. Pour placer N reines

- ◆ Si on a posé toutes les reines, alors l'échiquier est une solution
- ◆ Sinon, *pour toute case libre c*
 - ◆ si c n'est pas une case capturée^{*}:
 - ◆ placer une reine sur c
 - ◆ se rappeler récursivement avec (N-1) reine à placer
 - ◆ sinon, considérer la case suivante^{**}

La ligne ^{**} effectue le retour arrière.

La ligne ^{*} est un endroit où on pourra mettre une optimisation spécifique

Illustration (1/2)

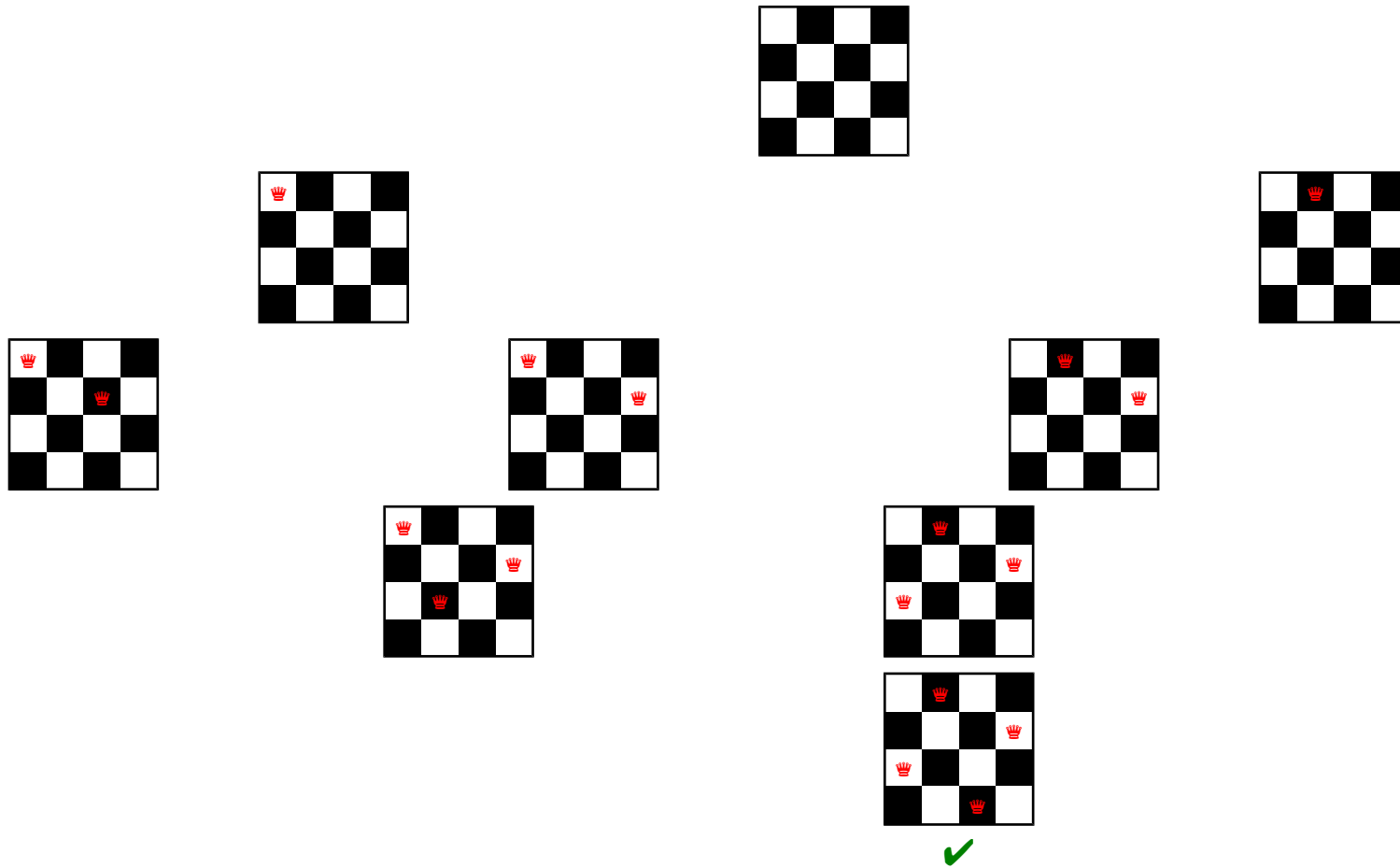
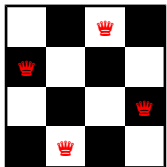
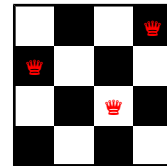
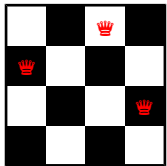
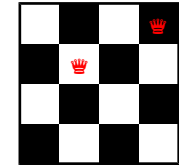
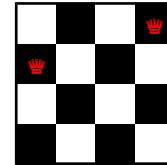
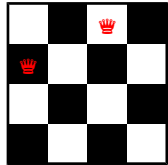
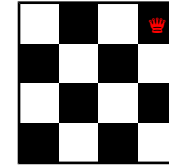
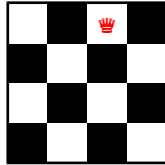
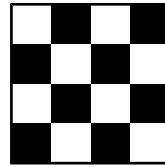


Illustration (2/2)



On écrit le code maintenant



On réfléchit ensemble à l'implémentation.

On finira en TP et on verra un deuxième exemple.

