TD 16

RETOUR SUR TRACE

1 Le problème des n dames

On considère un échiquier de taille $n \times n$ et l'on souhaite y placer n dames de manière à ce qu'aucune d'entre elles ne soit en prise, c'est-à-dire sur la même ligne, colonne ou diagonale qu'une autre dame.

Exercice 1

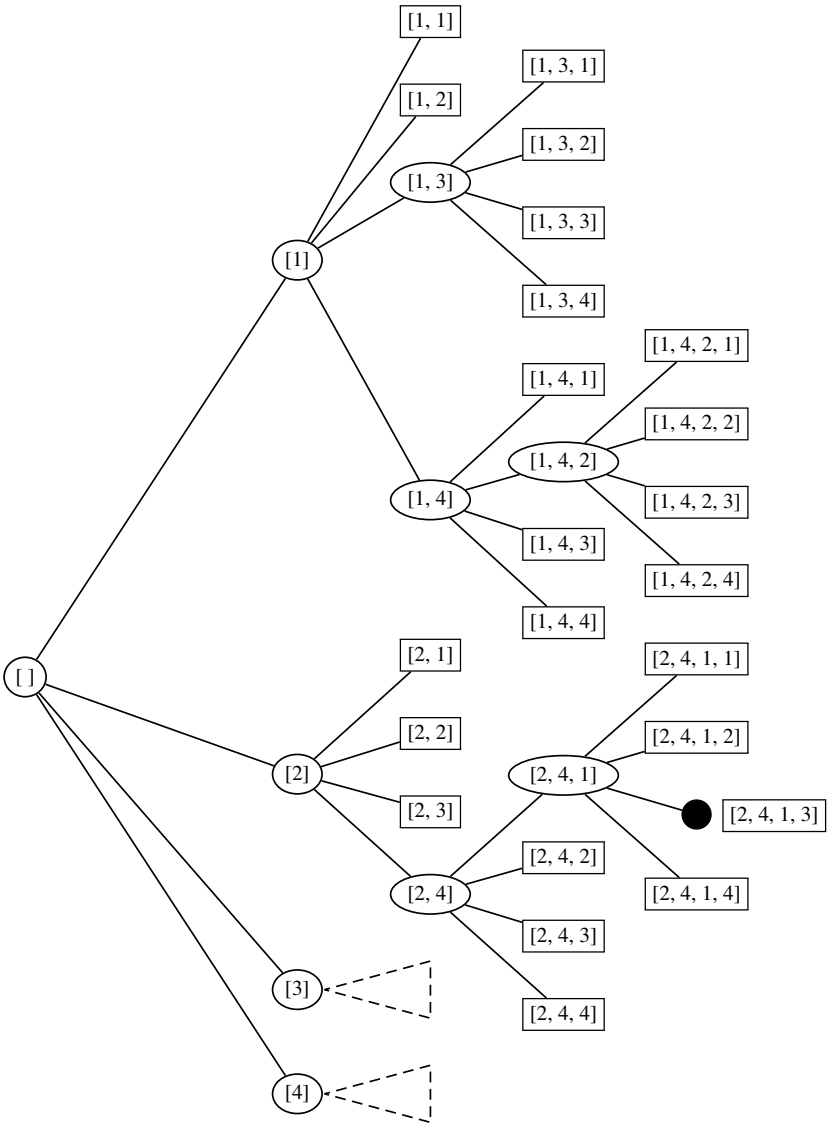
1. *Combien y a-t-il de manières de placer* 8 *dames sur un échiquier de* 64 *cases (sans contrainte particulière) ? Est-il raisonnable de tester toutes ces possibilités ?*

2. *Il est très facile de ne générer que les dispositions dans lesquelles il n'y a qu'une dame par ligne et par colonne. Combien y a-t-il de telles dispositions sur un échiquier $n \times n$? Est-il raisonnable de toutes les tester pour $n = 10$? $n = 20$?*

Le principe du *backtracking* (ou *retour sur trace*) s'applique à des problèmes de recherche exhaustive pour lesquels on peut facilement déterminer si une solution partielle a une chance de pouvoir donner une solution complète du problème. Ici, si l'on a déjà placé k dames (sur les k premières colonnes, par exemple), les seules positions à considérer pour la $k+1$-ème dame sont celles n'étant pas déjà menacées par l'une des k premières dames.

La bonne manière de raisonner est en termes d'arbres : un nœud interne de profondeur k correspond au placement des k premières dames (de manière correcte ou non), ses enfants aux différentes manières de rajouter la $k+1$-ème dame. Les feuilles sont les solutions complètes (n dames placées), dont seulement certaines sont correctes. L'idée du backtracking est de réaliser un parcours en profondeur de cet arbre en arrêtant l'exploration d'une branche dès que l'on sait qu'elle ne peut contenir de solution correcte.

On a représenté à la page suivante l'arbre correspondant au problème des 4 dames. Les nœuds (internes ou non) rectangulaires correspondent à des solutions incorrectes, ceux de forme ovale à des solutions « pour l'instant correctes », le disque noir à la solution complète et correcte. Les sous-arbres correspondant à un placement de la première dame sur la troisième ou quatrième ligne n'ont pas été représentés, ils s'obtiennent par symétrie (et contiennent donc une deuxième solution $[3, 1, 4, 2]$).



Il est possible (et même tentant) d'écrire une solution *ad hoc* à chaque fois que l'on est confronté à un problème se résolvant par *backtracking* : une approche adaptée au problème permet souvent de gagner un peu en efficacité et en simplicité. Cependant, il est important de comprendre la structure générale sous-jacente. Dans le fichier que vous avez récupéré, vous trouverez des définitions de type et une fonction `resout : 'a problem -> 'a option`. Cette fonction prend en argument la spécification d'un problème et renvoie `Some x` (où `x` est une solution) s'il le problème a une solution, `None` sinon. Le problème est spécifié par la donnée de deux fonctions et d'une configuration initiale :

- `accept : 'a -> 'a reponse` qui prend une solution (partielle) `x` et renvoie :
  — `Accepte y` si la solution est correcte et complète ;
  — `Partiel y` si la solution est (ou semble pour l'instant) cohérente, mais n'est pas complète ;
  — `Rejette` si l'on est sûr que la solution ne peut être complétée en une solution correcte (il y a déjà une incohérence).
  Le `y` renvoyé sera le plus souvent égal au `x` fourni en argument, mais l'on peut imaginer renvoyer un nouveau candidat dans lequel certaines déductions ont été faites.
- `enfants : 'a -> 'a list` qui prend une solution partielle et renvoie la liste de ses «enfants» (dans l'arbre décrit plus haut).

Je vous invite à lire attentivement le code fourni jusqu'à être sûr que vous avez bien compris le principe.

Exercice 2

1. *Écrire une fonction renvoyant une configuration correcte pour le problème des n dames (sous forme de liste ou de tableau). Vous devriez obtenir une réponse instantanément pour $n = 8$, et en quelques secondes pour $n = 20$.*
2. *Écrire une fonction dénombrant les solutions. Pour $n = 8$, on doit obtenir 92 (instantanément ou presque), pour $n = 12$ le calcul peut prendre quelques secondes.*

2 Tableaux auto-référents

Un tableau t d'entiers de taille n (indices allant de 0 à $n-1$) est dit *auto-référent* si, pour chaque i entre 0 et $n-1$, le nombre d'occurrences de i dans t est égal à t_i. Par exemple :

indices	0	1	2	3
t	2	0	2	0
occurrences	2	0	2	0

Exercice 3

1. *Écrire une fonction renvoyant un tableau auto-référent de taille n (s'il en existe). Pour l'instant, on ira au plus simple.*
2. *Écrire une fonction énumérant toutes les solutions de taille n.*
3. *Votre fonction marche-t-elle pour $n = 8$? 10 ? 20 ? 50 ? 80 ? Faire en sorte que ce soit le cas...*