TP 5 option informatique - MP* - Algorithmes de retour sur trace - 09/11/2023

L'objectif du TP est d'étudier différents algorithmes de retour sur trace (backtracking).

Vocabulaire 1. Un ***problème de satisfaction de contraintes*** est un problème dans lequel on cherche à construire un objet qui satisfait un ensemble de contraintes.

Exemple 2. Exemples de problèmes de satisfactions de contraintes :

- ⋆ Jeu du sudoku : étant donnée une grille de sudoku partiellement remplie, peut-on la compléter ?
- ⋆ Problème SAT : étant donnée une formule booléenne φ, existe-t-il une valuation des variables propositionnelles qui satisfait φ ?

Vocabulaire 3. Un ***algorithme de retour sur trace*** permet de résoudre un problème de satisfaction de contraintes. Si on note S l'ensemble des objets solutions au problème, un algorithme de retour sur trace va essayer de construire petit à petit un objet de S. À chaque étape, l'algorithme fait un choix (par exemple pour le sudoku : placer un chiffre dans une case), puis :

(a) Si ce choix lui permet de construire une solution $s \in S$, il renvoie s. En général, la construction de s se fait par un appel récursif.

(b) Sinon, il annule son choix, en fait un autre et retourne à l'étape (a). Le nom "retour sur trace" évoque cette étape de la procédure.

Dans l'étape (b), lorsque tous les choix possibles ont été explorés sans trouver de solution, c'est qu'il n'y en a pas (l'algorithme peut donc s'arrêter).

Remarque 4. Les algorithmes de retour sur trace sont en général très lents, mais certains problèmes ne peuvent pas être résolus autrement.

Exemple 5. Les algorithmes de retour sur trace sont utilisés pour résoudre le jeu « le compte est bon » (voir DM de l'année dernière), le jeu du sudoku (voir TP 16 de l'année dernière) et le problème SAT (on parle de « SAT-solver »).

Exercice 1. Le problème de la somme des sous-ensembles

Soit `li` une liste d'entiers et n un entier. Le problème de la somme des sous-ensembles (« subset-sum ») consiste à construire une sous-liste de `li` dont la somme des éléments vaut n.

1. Dans ce cadre, que vaut l'ensemble S évoqué dans l'introduction ? Quels sont les "choix" possibles pour l'algorithme à chaque étape ?
2. Écrire une fonction qui résout le problème de la somme des sous-ensembles. On utilisera le type `'a option` de OCaml.
3. Quelles sont les complexités spatiales et temporelles de vos fonctions ?

```
(* type déjà défini en OCaml *)
type 'a option =
  | Some of 'a
  | None;;
```

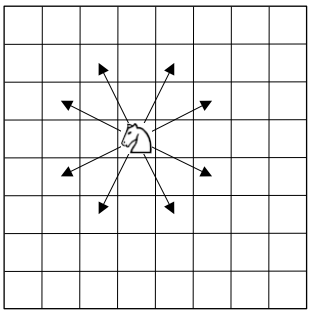
Exercice 2. Le problème des n dames

Soit $n \in \mathbb{N}$ un entier. On souhaite placer n dames sur une grille de taille $n \times n$ de sorte qu'aucune dame ne soit attaquée par une autre. Une dame attaque toutes les cases se trouvant sur la même ligne, la même colonne ou bien en diagonale par rapport à elle.

1. Écrire une fonction qui renvoie une solution au problème des n dames.
2. Écrire une fonction qui calcule le nombre de solutions au problème des n dames.
3. Montrer que la complexité spatiale est $\mathcal{O}(n^2)$ et que la complexité temporelle est en $\mathcal{O}(n2^{n^2})$.

Exercice 3. Cavalier d'Euler

On considère une grille carrée de taille $n \times n$ sur laquelle se trouve un cavalier (la figure ci-contre représente les 8 déplacements possibles). Un « tour d'Euler » est une suite de $n^2 - 1$ déplacements permettant de visiter une et une seule fois chaque case de la grille.



1. Écrire une fonction qui renvoie un tour d'Euler.
2. Écrire une fonction qui renvoie le nombre de tours d'Euler.
3. Quelles sont les complexités spatiales et temporelles de vos fonctions ?

Exercice 4. Le problème des n dames (avec des exceptions)

On souhaite résoudre le problème des n dames (voir exercice 2) en utilisant des exceptions.

```
(* Exemple d'utilisation d'une exception *)
exception Indice of int;;

let rec f li i = match li with
  | [] -> print_string "La liste ne contient pas 0"
  | 0 :: _ -> raise (Indice i)
  | _ :: q -> f q (i+1);;

let main li = try f li 0 with
  | Indice i -> print_string "La liste contient 0 à l'indice ";
                print_int i;;

main [1;2;3];;   (* Affiche "La liste ne contient pas 0" *)
main [1;2;0;3];; (* Affiche "La liste contient 0 à l'indice 2" *)
```

Pour résoudre le problème des n-dames, on définit l'exception suivante :

```
exception Fini of (int*int) list;;
```

1. Écrire une fonction qui renvoie une solution au problème des n dames. Pour cela, on définira une fonction intermédiaire `backtrack` dont le type de retour est `unit`. Lorsque `backtrack` a trouvé une solution au problème, elle déclenche l'exception « `Fini li` » où `li` la liste des positions des dames. La fonction principale doit renvoyer la liste des positions (et non déclencher l'exception `Fini`).

Exercice 5. Carrés magiques

Un ***carré magique normal*** est une grille de taille $n \times n$, où chaque entier de $[\![1; n^2]\!]$ apparaît une et seule fois, et telle que les sommes des entiers sur chaque ligne, chaque colonne et chacune des deux diagonales soient égales à $\frac{n(n^2+1)}{2}$. La figure de droite est un carré magique normal pour $n = 5$.

7	18	23	5	12	→65
15	20	1	21	8	→65
3	19	14	4	25	→65
24	2	17	13	9	→65
16	6	10	22	11	→65
65	65	65	65	65	65	65

1. Écrire une fonction qui prend en entrée une grille partiellement remplie et la complète en un carré magique normal.