TD 6. Recherche exhaustive

Exercice 1. *Parcours d'objets*

Les algorithmes de recherche exhaustive nécessitent de parcourir différents types d'objets. Le but de cet exercice est d'écrire quelques uns de ces parcours.
Pour chacun des ensembles suivants, effectuer les tâches suivantes :

1. déterminer un ordre de parcours de l'ensemble, avec un début et une fin ;
2. écrire un algorithme SUIVANT qui, étant donné un élément de l'ensemble, renvoie le suivant pour l'ordre déterminé ;
3. analyser la complexité de l'algorithme SUIVANT ;
4. compter le nombre total d'éléments dans l'ensemble.

   i. Ensemble des n-uplets d'entiers entre 0 et $k-1$.
   ii. Ensemble des n-uplets d'entiers *décroissants* entre 0 et $k-1$.
   iii. Ensembles des sous-ensembles de $\{0,\dots,n-1\}$.
   iv. Ensemble des mots de longueur n sur un alphabet Σ à s lettres.

Exercice 2. *Somme et sac-à-dos*

Étant donné un tableau d'entiers T et une cible c, on cherche à déterminer s'il existe une sous-liste de T dont la somme des éléments vaut s exactement. *Par exemple, si* $T = [-5,2,7,-3,2,-6]$ *et* $c = 5$*, alors il suffit de prendre* $[2,7,2,-6]$ *dont la somme bien* 5*. Par contre si* $c = 10$ *ou* $c = -13$*, il n'y a aucune solution.*

1. Décrire une façon de représenter et parcourir tous les sous-tableaux de T.
2. Écrire un algorithme pour résoudre le problème et analyser sa complexité.

Le *problème du sac-à-dos* est le suivant : étant donné n objets $(t_0, v_0), \dots, (t_{n-1}, v_{n-1})$ et une taille S, on cherche un sous-ensemble $I \subset \{0,\dots,n-1\}$ qui maximise la *valeur* totale $\sum_{i\in I} v_i$ tout en respectant la contrainte $\sum_{i\in I} t_i \leq S$.

3. Adapter l'algorithme précédent à ce problème et analyser sa complexité.

Exercice 3. *Huit reines*

Le *problème des huit reines* demande s'il est possible de placer 8 reines sur un échiquier 8×8, sans qu'elles se menacent l'une l'autre : deux reines se menacent si elle sont sur la même ligne, la même colonne, ou la même diagonale (ou anti-diagonale). C'est en effet possible, et il y a même plusieurs solutions. Ce qui nous intéresse ici est de calculer toutes les solutions possibles. De plus, on généralise le problème au cas des *n reines* : on cherche à placer n reines sur un échiquier $n \times n$, avec toujours les mêmes conditions.

1. **i.** Montrer que le problème n'admet aucune solution pour $n = 2$ et $n = 3$.
   ii. Pourquoi ne peut-on pas mettre strictement plus de n reines sur un échiquier $n \times n$?
   iii. Démontrer que dans toute solution, chaque ligne de l'échiquier contient exactement une reine.

On représente une solution par un tableau Q de taille n, tel que $Q_{[i]} = j$ s'il y a une reine en case (i, j) de l'échiquier.

2. **i.** Quelle propriété de Q est impliquée par le fait deux reines ne peuvent pas appartenir à la même colonne ?
   ii. Écrire une fonction qui teste si Q est une solution valide et analyser sa complexité.
   iii. En déduire un algorithme pour le problème et analyser sa complexité.

Exercice 4. *Codes de Gray*

Un code de Gray est une énumération de tous les mots binaires de longueur n dans un ordre particulier, afin que pour passer d'un mot au suivant, seul un bit ait besoin d'être modifié. Par exemple, l'énumération suivante est un code de Gray sur 2 bits : 00, 01, 11, 10. Les codes de Gray permettent une énumération très efficace des mots binaires, et donc des sous-ensembles d'un ensemble donné.

1. Écrire un code de Gray sur 3 bits.
2. Quelle est la longueur d'un code de Gray ?
3. Soit T un tableau contenant un code de Gray sur n bits. On définit les tableaux T^0 et T^1 par $T^0_{[i]} = 0 \cdot T_{[i]}$ et $T^1_{[i]} = 1 \cdot T_{[i]}$: on concatène 0 ou 1 en tête de $T_{[i]}$.
   i. Montrer que tous les mots binaires sur $n + 1$ bits sont dans $T^0 \cup T^1$.
   ii. Montrer comment construire *facilement* un tableau T^+ contenant un code de Gray sur $n + 1$ bits à partir des tableaux T^0 et T^1.
   iii. En déduire un algorithme de construction d'un code de Gray sur n bits et analyser sa complexité.