Travaux dirigés n^o 9
Algorithmes gloutons (ou pas)

Les deux premiers exercices portent sur des *algorithmes gloutons*, le dernier exercice illustre la technique de *backtracking* en revenant sur l'exemple du cours.

Exercice 9.1 — Sac-à-dos fractionnel

On considère un ensemble d'items i avec un poids w_i et une valeur v_i associés. On peut considérer les poids et valeurs entiers au départ. On précise que les items sont **fractionnables**.

Données :

— une collection d'items sous la forme de triplets (i, w_i, v_i)

— un poids W

Problème :

On veut maximiser le bénéfice en valeur sous la contrainte du poids total maximal W.
Il faut retourner le bénéfice réalisé et la liste d'items avec le poids prélevé de chacun.

1. Comprenez bien le problème sur l'exemple suivant afin de savoir quelle structure de données utiliser avant de le résoudre :

   $$[(\text{'ananas'}, 4, 12), (\text{'banane'}, 8, 32), (\text{'clémentine'}, 2, 40), (\text{'kiwi'}, 6, 30), (\text{'pomme'}, 1, 50)]$$

   Le poids maximal W est fixé à 10.
2. Trouvez un algorithme *glouton* pour résoudre ce problème.

Exercice 9.2 — Ordonnancement de tâches

Voici un certain nombre de tâches à exécuter le matin avant de sortir de chez soi :

1. se lever
2. verser son lait au chat
3. partir
4. s'habiller
5. se réveiller
6. prendre son petit-déjeûner
7. se laver

Ces tâches matinales ne sont pas totalement ordonnées or on aurait besoin d'un *ordonnancement* de ces tâches, autrement dit, d'un ordre total. Il faut *linéariser* cet ordre, c'est à dire l'étendre en un ordre total qui lui soit compatible. L'algorithme qui résout ce problème est couramment appelé *tri topologique*.

Entrée : un ensemble fini de n tâches muni d'un ordre partiel
Problème : un ordre total sur ces n tâches compatible avec l'ordre partiel donné

1. Définissez un ordre partiel sur ces 7 éléments et tracez le graphe orienté *acyclique* correspondant.
2. Trouvez un algorithme glouton fournissant un ordre total à partir du graphe d'un ordre partiel *(Indication : on pourra conserver dans un tableau le nombre de prédécesseurs d'un sommet).*
3. Appliquez votre algorithme à l'exemple ci-dessus.
4. Existe-t-il d'autres solutions que celle renvoyée par l'algorithme ? Pourquoi ?

Exercice 9.3 — Les 8 reines

Contrairement au précédent, cet exercice ne porte pas sur les algorithmes gloutons mais sur la technique de *backtracking* (ou rétro-parcours en français). Afin de bien le comprendre, on se propose de revenir sur le problème des 8 reines traité en cours :

Entrée : un échiquier de $n \times n$ cases
Problème : placer n reines sans qu'elles soient mutuellement en prise

1. Assurez-vous de bien avoir compris l'algorithme du cours, notamment les fonctions choixCorrect (Y,k) et prolonge(Y,k).
2. Quelle est la complexité de l'algorithme vu en cours ? Qu'en penser ?
3. Effectuez la trace des appels à la fonction prolonge(Y,k) de cet algorithme, pour le cas où $n = 4$ afin de trouver toutes les solutions.