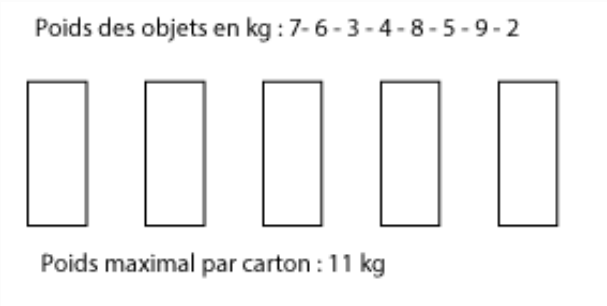
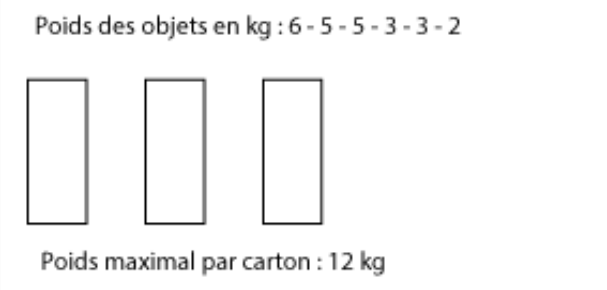
# LEÇON N°9 : ALGORITHMES GLOUTONS


## 1. ALGORITHME DE RANGEMENT

On considère différents objets dont on connait le poids et que l'on veut ranger dans un certain nombre de cartons ayant un même poids limite maximal. On désire de plus limiter au maximum le nombre de cartons.



Pour résoudre ce type de problème, on peut chercher à mettre en place différents algorithmes. Un premier algorithme serait de chercher toutes les solutions possibles puis de choisir la meilleure. Il s'agit là d'une **recherche exhaustive** très couteuse en calculs et donc en temps. Une autre approche consiste à chercher un algorithme qui construit une solution pas à pas :

- sans revenir sur ces décisions,
- en choisissant à chaque étape la solution qui semble la meilleure,
- en espérant obtenir ainsi un résultat globalement optimum.

Un tel algorithme de recherche est appelé **algorithme glouton**.

Ce type d'algorithme, suivant les problèmes, ne garantit pas une optimalité de la réponse mais propose une très bonne approche d'une solution qui si elle n'est pas optimale a au moins été produite de manière peu coûteuse. Un algorithme glouton repose sur la recherche **d'un optimum local** à chaque étape, en espérant s'approcher ainsi pas à pas d'un **optimum global**.

Pour le problème de rangement, un bon choix d'algorithme glouton est de placer en premier les objets de plus grand poids dans le premier carton disponible. Cet algorithme ne fournir pas toujours la meilleure solution mais en propose une de manière rapide et peu coûteuse.

# 2. RENDU DE PIÈCES


Le **problème du rendu de monnaie** est un problème d'algorithmique. Il s'énonce de la façon suivante : étant donné un système de monnaie (pièces et billets), comment rendre une somme donnée de façon optimale, c'est-à-dire avec le nombre minimal de pièces et billets ?
Dans la zone euro, le système de pièces et de billets en vigueur est, en mettant de côté les centimes d'euros : **{1, 2, 5, 10, 20, 50, 100, 200 €}**

**Hypothèses :** on dispose d'autant d'exemplaires qu'on souhaite de chaque pièce et billet. Par la suite nous désignerons par monnaie, indifféremment une pièce ou un billet.

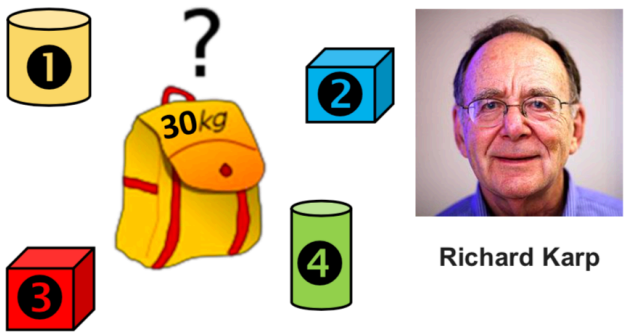
Algorithme glouton proposé :

**Exercice 1**

1. Tester l'algorithme glouton pour un rendu de 47 Euros.
2. Cet algorthme est-il optimal ?

   On dit d'un système monétaire qu'il est canonique lorsque l'algorithme glouton de rendu de monnaie donne la réponse optimale. Ce n'est pas le cas de tous les systèmes monétaires.
3. On considère le système monétaire {1;3;6;12;24;30}. Appliquer l'algorithme glouton de rendu de monnaie pour un rendu de 49 €. Est-il optimal ?

# 3. ENCORE UN PROBLÈME DE RANGEMENT : LE KP




Le problème du sac à dos : quelles boîtes choisir afin de maximiser la somme emportée tout en ne dépassant pas les 30 kg autorisés ?



Richard Karp

En algorithmique, le **problème du sac à dos**, noté également **KP** (en anglais, *Knapsack problem*) est un problème d'optimisation combinatoire. Il modélise une situation analogue au remplissage d'un sac à dos, ne pouvant supporter plus d'une certaine masse, avec tout ou partie d'un ensemble donné d'objets ayant chacun une valeur spécifique en euros. Les objets mis dans le sac à dos doivent maximiser la valeur totale en euros, sans dépasser la masse maximale. Le problème du sac à dos est l'un des 21 problèmes NP-complets du mathématicien américain **Richard Karp**, exposés dans son article de 1972. Ce dernier reçu le prix Turing en 1985 pour ses travaux.

| Objets | ❶ | ❷ | ❸ | ❹ |
|---|---|---|---|---|
| Valeurs $v_i$ (€) | 70 | 40 | 30 | 30 |
| Masse $m_i$ (kg) | 13 | 12 | 8 | 10 |

1. Recherche exhaustive des solutions et détermination de la solution optimale.

2. Un algorithme glouton proposé

Une méthode approchée a pour but de trouver une solution avec un bon compromis entre la qualité de la solution et le temps de calcul. Pour le problème du sac à dos, voici un exemple d'algorithme de ce type :

- calculer le rapport ($v_i$ / $m_i$) pour chaque objet i ;
- trier tous les objets par ordre décroissant de cette valeur ;
- sélectionner les objets un à un dans l'ordre du tri et ajouter l'objet sélectionné dans le sac si le poids maximal reste respecté.

Effectuons le déroulé de cet algorithme sur notre exemple :

■ **Première étape :**

| Objets | ❶ | ❷ | ❸ | ❹ |
|---|---|---|---|---|
| $v_i$ (€) | 70 | 40 | 30 | 30 |
| $m_i$ (kg) | 13 | 12 | 8 | 10 |
| $v_i$ / $m_i$ | 5,4 | 3,3 | 3,7 | 3,0 |

■ **Deuxième étape :**

| Objets | ❶ | ❸ | ❷ | ❹ |
|---|---|---|---|---|
| $v_i$ (€) | 70 | 30 | 40 | 30 |
| $m_i$ (kg) | 13 | 8 | 12 | 10 |
| $v_i$ / $m_i$ | 5,4 | 3,7 | 3,3 | 3,0 |

■ **Troisième étape :**

**Objet ❶ :** on le met dans le sac vide, la masse du sac est alors de 13 kg ≤ 30 kg.

**Objet ❸ :** on le met dans le sac car 13 + 8 = 21 kg ≤ 30 kg.

**Objet ❷ :** on ne le met pas dans le sac car 21 + 12 = 33 kg ≥ 30 kg.

**Objet ❹ :** on ne le met pas dans le sac car 21 + 10 = 31 kg ≥ 30 kg.