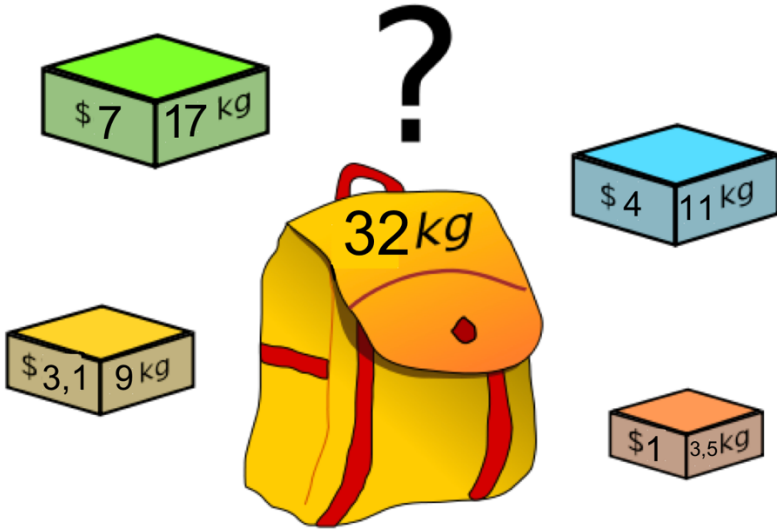
Le problème du sac à dos

Supposons que vous découvriez la caverne d'Ali Baba (et des 40 voleurs), il y a une infinité d'objets de valeur `v1, v2, · · · , vn` mais chaque objet de valeur `vi` a un poids de `pi`. Comment **remplir votre sac-à-dos** avec le contenu qui **rapportera le plus**, sachant que votre sac ne supporte qu'un poids `W`?



1. Adapter la stratégie du rendu de monnaie

Structure de données des objets proposés:

rendu de monnaie : $	sac à dos (vi,pi)
[1, 2, 5, 10, ...]	[(1, 3.5), (3.1, 9), (4,11), ...]

Algorithme:

rendu de monnaie	sac à dos
Pour tout montant de piece de la caisse parcourue en sens inverse	Pour tout objet (vi, pi) du tresor parcouru en sens (vi/pi décroissant)
Test : le montant de la piece est-il < i?	**Test** : Le poids de l'objet est-il < i?
Si oui: rendre la piece	Si oui: prendre l'objet
Sinon, passer à la piece suivante	

Question 1:

Déterminer le choix idéal pour un sac à dos de capacité **32 kg**, avec les objets suivants, proposés en nombre infini: `**[{"valeur": 7, "poids":17},{"valeur": 4, "poids":11},{"valeur": 3.1, "poids":9},{"valeur": 1, "poids":3.5}]**

Objets	1	2	3	4
valeur	7	4	3.1	1
poids (kg)	17	11	9	3.5
ratio val/poids	0.41	0.36	0.34	0.29

La *stratégie*, sur le modèle du rendu de monnaie sera la suivante:

- classer les objets dans l'ordre décroissant de leur taux de valeur (taux de valeur = valeur / masse). Ainsi le premier élément de la liste sera celui ayant le meilleur rapport valeur/masse.
- on prend le premier élément de la liste, puis le deuxième, etc., tant que le sac peut encore les contenir.

2. Algorithme de type glouton

Nous allons chercher à adapter ici aussi l'algorithme glouton du rendu de monnaie:

Cela reviendra à remplir le tableau suivant, de manière ascendante:

- Commencer par le sac de plus petite capacité.
- Pour chaque objet de poids < capacité du sac:
  - prendre l'objet.
  - Compléter le sac avec la solution optimale déjà trouvée dans le tableau.
- Choisir parmi les solutions trouvées, celle de gain optimal.
- Continuer avec un sac de capacité supérieure.

capacité du sac (kg)	Liste des objets (valeur,poids)	coût cumulé ($)
4, 5, 6	`[(1,3.5)]`	1
7,8	`[(1,3.5), (1,3.5)]`	2
9	..	..
10	..	..
11	..	..
12	..	..
13	..	..
14	..	..
15	..	..
16	..	..
17	..	..
18	..	..

***Question* 2**

Déterminer le choix idéal pour un sac à dos de capacité **18 kg**, avec les objets suivants, proposés en nombre infini: `` `[{"valeur": 7, "poids":17},{"valeur": 4, "poids":11},{"valeur": 3.1, "poids":9},{"valeur": 1, "poids":3.5}]``

S'il reste du temps : **EPREUVE PRATIQUE : S11.2** (programmation dynamique, liste, déterminer la plus grande somme possible de ses éléments), **S31.1** (sequence recherche motif dans une chaine), **S32.2** (rendu glouton, listes), **S33.2** (empaqueter, algo glouton)