



ALGORITHMES GLOUTONS

Ou pourquoi il ne faut pas laisser les pigeons conduire les bus



La stratégie gloutonne est une manière naïve de résoudre un problème d'optimisation. Dans cette stratégie, on ne regarde que les gains sur le court terme.



Si on a la choix entre une carotte appétissante et un cube de terre, on prend la carotte ! D'où le terme « **glouton** » !

C'EST QUOI UN PROBLÈME D'OPTIMISATION ?

Problème du
RENDU DE MONNAIE



Problème du
VOYAGEUR DE COMMERCE



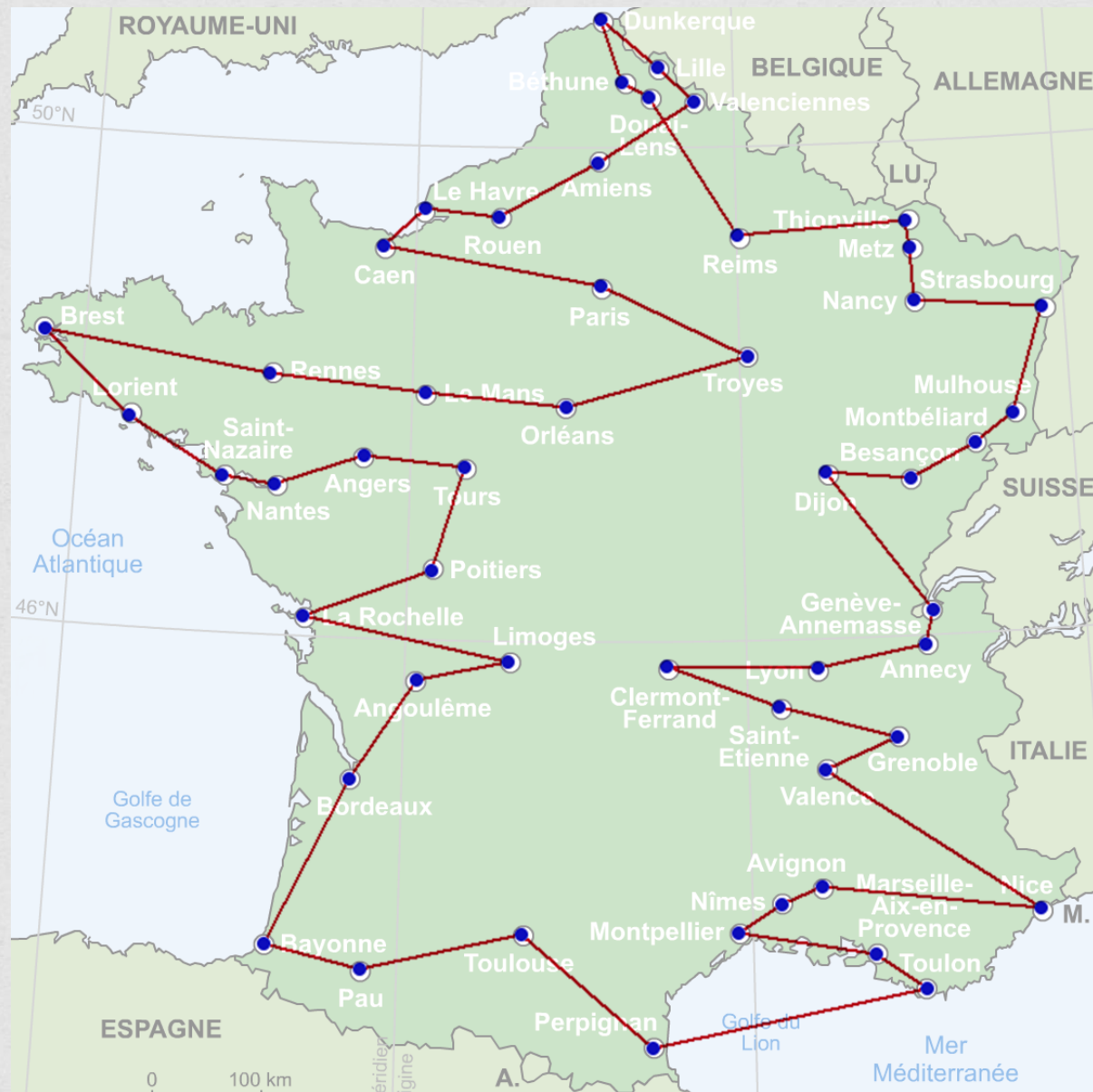
Problème du
SAC À DOS



LE TOUR DE FRANCE



LE TOUR DE FRANCE

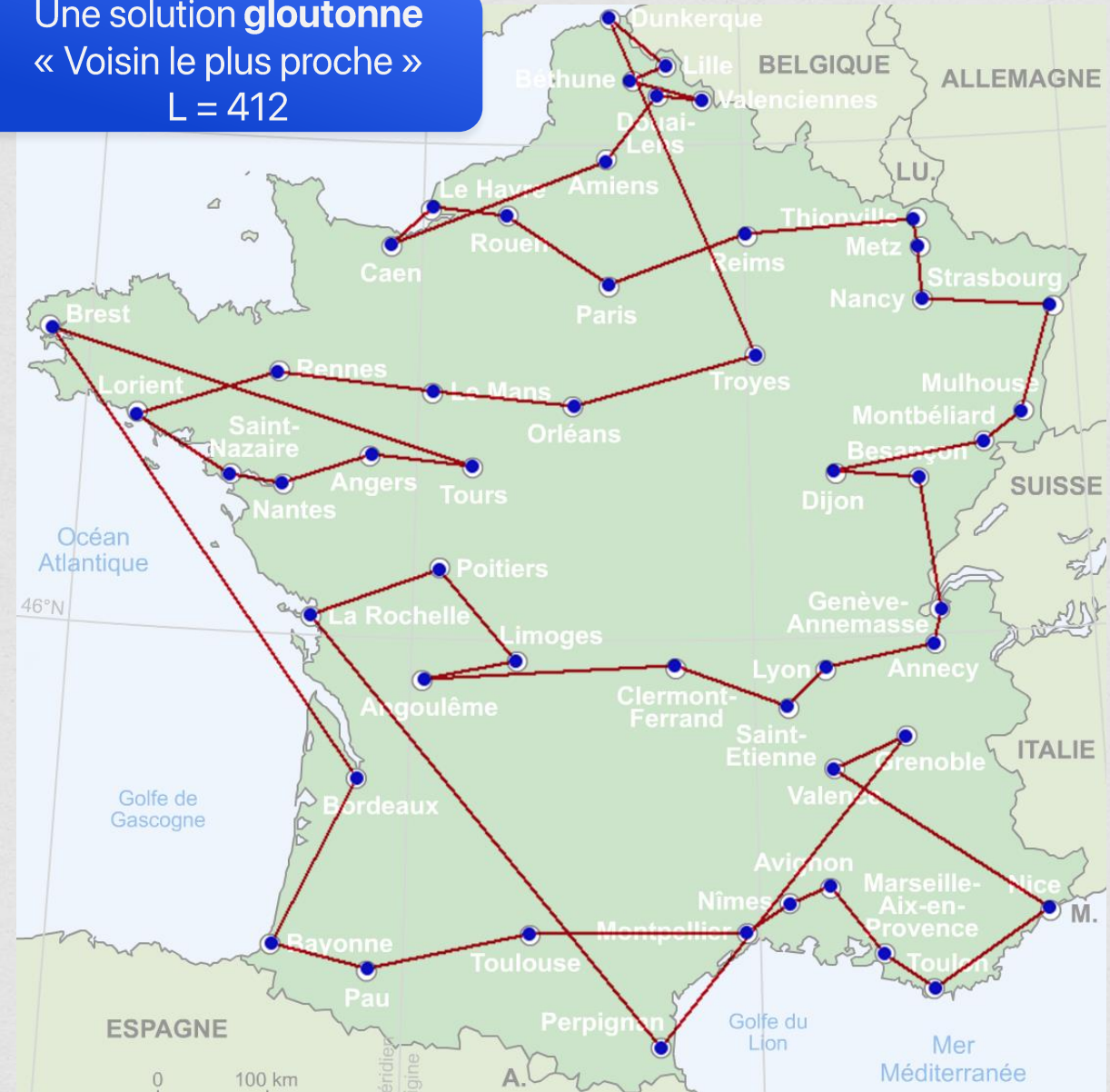


LE TOUR DE FRANCE

Le meilleur tour possible
La solution **optimale**
 $L = 307$



Une solution **gloutonne**
« Voisin le plus proche »
 $L = 412$



REND L'ARGENT !



Un caissier souhaite rendre une certaine **somme** à un client. Il dispose d'une quantité illimitée de **pièces de différentes valeurs**.

Son objectif est de rembourser cette somme en rendant le **minimum** de pièce.



Par exemple, le caissier doit rendre une somme de **29€** avec des pièces de **1€, 2€, 5€ et 10€**.



Facile ! Il rend **5** pièces. Plus précisément, il rend les pièces : **10€, 10€, 5€, 2€, 2€**



Le caissier doit rendre une somme de **25€** avec des pièces de **1€, 5€, 6€ et 7€**



Peut-être **7** pièces ? **7€, 7€, 7€, 1€, 1€, 1€, 1€**



Une solution avec **4** pièces ! **7€, 7€, 6€, 5€**



Rendre **1093€** avec des pièces de **1€, 21€, 23€, 29€** ?



Il existe une solution avec **38** pièces !

REND L'ARGENT !



Un caissier souhaite rendre une certaine **somme** à un client. Il dispose d'une quantité illimitée de **pièces de différentes valeurs**.

Son objectif est de rembourser cette somme en rendant le **minimum** de pièce.

Un problème d'optimisation



Un **problème d'optimisation** est un problème où l'on souhaite **minimiser** ou **maximiser** une certaine quantité. Cette quantité est appelée **objectif**.



Pour le problème du rendu de monnaie, on souhaite **minimiser la quantité de pièces rendues**.



Par exemple, le caissier doit rendre une somme de **29€** avec des pièces de **1€, 2€, 5€ et 10€**.



Facile ! Il rend **5** pièces. Plus précisément, il rend les pièces : **10€, 10€, 5€, 2€, 2€**



Le caissier doit rendre une somme de **25€** avec des pièces de **1€, 5€, 6€ et 7€**



Peut-être **7** pièces ? **7€, 7€, 7€, 1€, 1€, 1€, 1€**



Une solution avec **4** pièces ! **7€, 7€, 6€, 5€**



Rendre **1093€** avec des pièces de **1€, 21€, 23€, 29€** ?



Il existe une solution avec **38** pièces !

REND L'ARGENT !



Un caissier souhaite rendre une certaine **somme** à un client. Il dispose d'une quantité illimitée de **pièces de différentes valeurs**.

Son objectif est de rembourser cette somme en rendant le **minimum** de pièce.



Une **instance** d'un problème d'optimisation définit concrètement les **données** de celui-ci.



Quelles sont les données qui définissent une instance du problème du rendu de monnaie ?



Une instance se définit par :

- Une somme à rendre
- Les valeurs des différentes pièces à disposition



Par exemple, le caissier doit rendre une somme de **29€** avec des pièces de **1€, 2€, 5€ et 10€**.



Facile ! Il rend **Une instance** précisément, il rend les pièces : **10€, 10€, 5€, 2€, 2€**



Le caissier doit rendre une somme de **25€** avec des pièces de **1€, 5€, 6€ et 7€**



Peut-être 7 pièces ! **Une instance** 1€, 1€, 1€, 1€



Une solution avec **4** pièces ! **7€, 7€, 6€, 5€**



Rendre **1093€** avec des pièces de **1€, 21€, 23€, 29€** ?



Il existe une solution ! **Une instance** pièces !

REND L'ARGENT !



Un caissier souhaite rendre une certaine **somme** à un client. Il dispose d'une quantité illimitée de **pièces de différentes valeurs**.

Son objectif est de rembourser cette somme en rendant le **minimum** de pièce.



Par exemple, le caissier doit rendre une somme de **29€** avec des pièces de **10€, 5€, 2€ et 1€**.

Coût de la solution
Valeur de l'objectif réalisée



Facile ! Il rend **5** pièces. Plus précisément, il rend les pièces : **10€, 10€, 5€, 2€, 2€**

Une solution



Le caissier doit rendre une somme de **25€** avec des pièces de **1€, 5€, 6€ et 7€**



Peut-être **7** pièces ? **7€, 7€, 7€, 1€, 1€, 1€, 1€**



Une solution avec **4** pièces ! **7€, 7€, 6€, 5€**



Rendre **1093€** avec des pièces de **1€, 21€, 23€, 29€** ?



Il existe une solution avec **38** pièces !

RENDS L'ARGENT !



Un caissier souhaite rendre une certaine **somme** à un client. Il dispose d'une quantité illimitée de **pièces de différentes valeurs**.

Son objectif est de rembourser cette somme en rendant le **minimum** de pièce.



Par exemple, le caissier doit rendre une somme de **29€** avec des pièces de **1€, 2€, 5€ et 10€**.



Facile ! Il rend **5** pièces. Plus précisément, il rend les pièces : **10€, 10€, 5€, 2€, 2€**



Le caissier doit rendre une somme de **25€** avec des pièces de **1€, 5€, 6€ et 7€**



Peut-être **7** pièces ? **7€, 7€, 7€, 1€, 1€, 1€, 1€**



Une solution avec **4** pièces ! **7€, 7€, 6€, 5€**

Coût **optimal**

Solution **optimale**



Rendre **1093€** avec des pièces de **1€, 21€, 23€, 25€** ?



Il existe une solution avec **38** pièces !

ATTENZIONE PICKPOCKET !



Robin des Bois s'introduit dans le château de Richard Cœur de Lion. Ce château renferme tout un tas d'**objets** de **poids** et de **valeurs** différents.



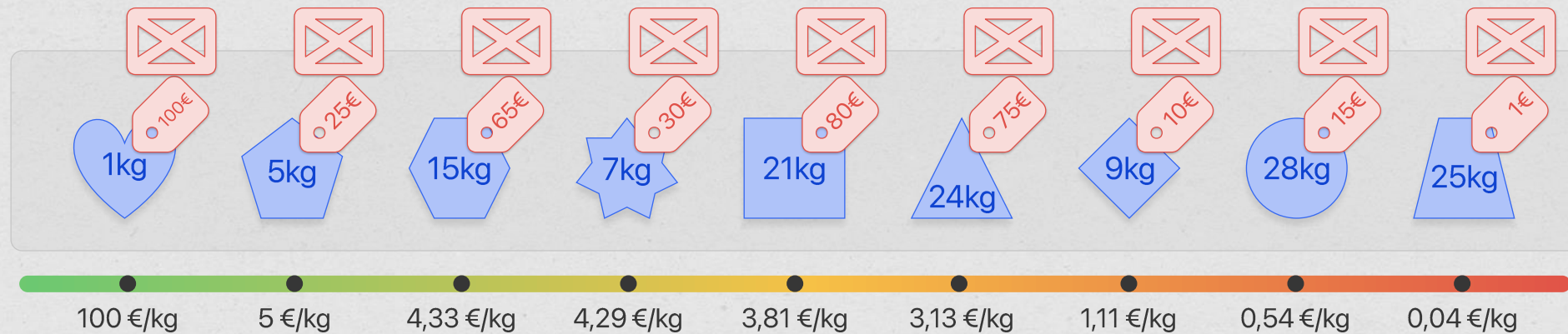
Sachant que Robin des Bois ne peut transporter que **40 kg**, quels objets doit-il dérober pour **maximiser** la valeur de son recel ? Voici les objets :

Objet n°	1	2	3	4	5	6	7	8	9
Poids kg	28	5	15	9	24	7	21	25	1
Prix €	15	25	65	10	75	30	80	1	100

ATTENZIONE PICKPOCKET !

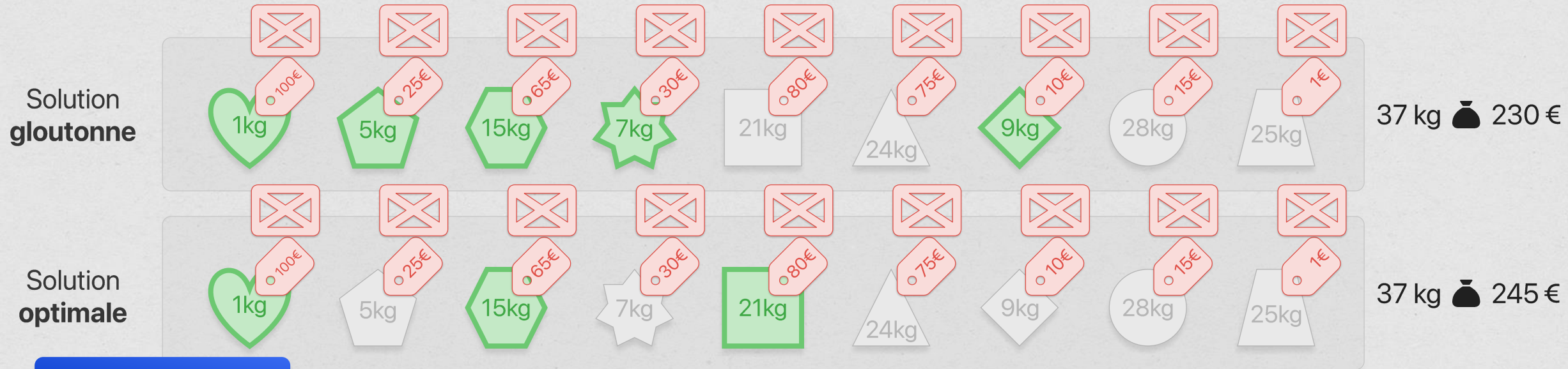


ATTENZIONE PICKPOCKET !



Une stratégie est de considérer les objets selon leur rapport **prix / poids** (efficacité) par ordre décroissant. Suivant cet ordre, on ajoute un objet si son ajout respecte la contrainte de capacité.

ATTENZIONE PICKPOCKET !



Stratégie gloutonne



Une stratégie est de considérer les objets selon leur rapport **prix / poids** (efficience) par ordre décroissant. Suivant cet ordre, on ajoute un objet si son ajout respecte la contrainte de capacité.



Est-ce que cette stratégie aboutit nécessairement au meilleur recel, ou plus formellement à la **solution optimale** ?

MODÉLISER UN PROBLÈME D'OPTIMISATION

PROBLÈME D'OPTIMISATION



DONNÉES



VARIABLES



OBJECTIF



CONTRAINTES

DESCRIPTION



Les données définissent une **instance** du problème.



Les variables définissent une **solution** du problème.



La fonction objectif donne le **coût** réalisé suivant une solution.



Les **contraintes** imposées par le problème.

EXEMPLE — RENDU DE MONNAIE



La somme à rendre et les valeurs des différentes pièces.



Les pièces rendues.



Le nombre de pièces rendues. On souhaite **minimiser** cette quantité.



La somme des pièces rendues doit être égale à la somme à rendre.

MODÉLISER UN PROBLÈME D'OPTIMISATION

PROBLÈME D'OPTIMISATION



DONNÉES



VARIABLES



OBJECTIF



CONTRAINTES

EXEMPLE — RENDU DE MONNAIE



La somme à rendre et les valeurs des différentes pièces.



Les pièces rendues.



Le nombre de pièces rendues. On souhaite **minimiser** cette quantité.



La somme des pièces rendues doit être égale à la somme à rendre.

TRADUCTION MATHÉMATIQUES



Somme S à rendre
 n pièces de valeurs $P = (p_1, p_2, p_3, \dots, p_n)$



Soit $x_i \in \mathbb{N}$ le nombre de pièces p_i rendues pour $i \in \llbracket 1, n \rrbracket$.



$\min x_1 + x_2 + x_3 + \dots + x_n$



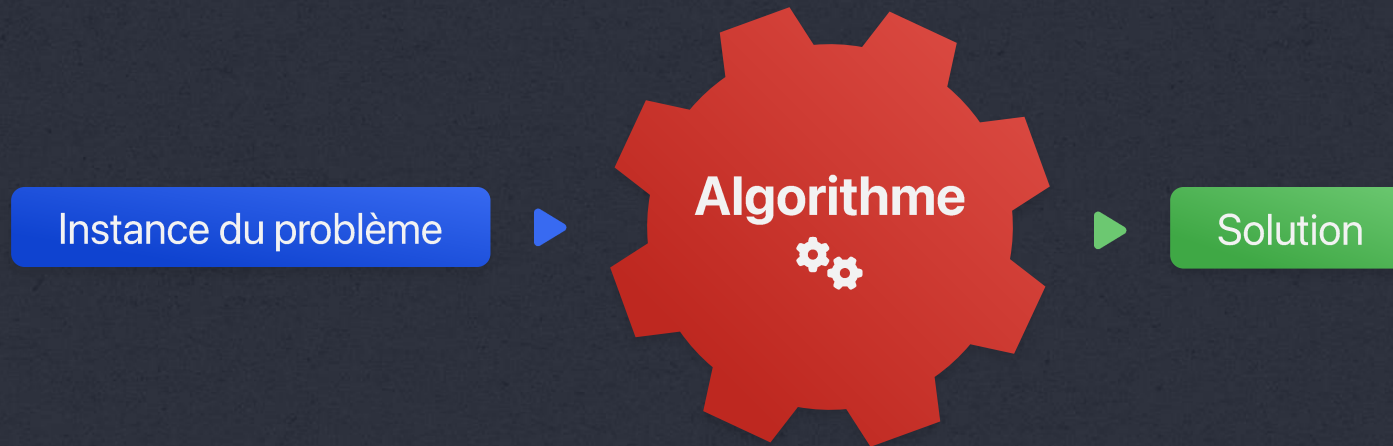
$p_1x_1 + p_2x_2 + p_3x_3 + \dots + p_nx_n = S$

$$c^*(S, P) = \min \left\{ \sum_{i=1}^n x_i \mid \sum_{i=1}^n p_i x_i = S, x \in \mathbb{N}^n \right\}$$

NOTRE OBJECTIF : CRÉER UN ALGORITHME DE RÉOLUTION



Un algorithme de résolution est **spécifique** à un unique problème d'optimisation particulier !



```
def résoudre(instance):  
    # ...  
    return coût, solution
```




EXEMPLE – PROBLÈME DU SAC À DOS



Capacité du  : 40 kg

**Algorithme
glouton**



Valeur totale du  : 230€
Poids total du  : 37 kg

RÉSoudre, C'EST CHOISIR



Pour résoudre un problème, on considère souvent une séquence de **choix** successifs.



Pour le problème du rendu de monnaie, à chaque étape, on choisit une pièce à rendre :

Pièce → Pièce → Pièce → Pièce → etc.



Pour le problème du sac à dos, à chaque étape, on choisit un objet à affecter au sac :

Objet → Objet → Objet → Objet → etc.



Pour le problème du voyageur de commerce, à chaque étape, on choisit une ville à visiter :

Ville → Ville → Ville → Ville → etc.

LA STRATÉGIE GLOUTONNE



Pour résoudre un problème, on considère souvent une séquence de **choix** successifs.



Une **stratégie gloutonne** donne une manière d'effectuer ces choix. Elle effectue à chaque étape, le choix qui semble être « le meilleur possible » sur le moment.



Pour le problème du rendu de monnaie, à chaque étape, on choisit une pièce à rendre :

Pièce → Pièce → Pièce → Pièce → etc.



À chaque étape, on donne la plus grande pièce possible.



Pour le problème du sac à dos, à chaque étape, on choisit un objet à affecter au sac :

Objet → Objet → Objet → Objet → etc.



À chaque étape, on affecte au sac l'objet non-affecté qui puisse rentrer dans le sac avec le plus grand prix / poids.



Pour le problème du voyageur de commerce, à chaque étape, on choisit une ville à visiter :

Ville → Ville → Ville → Ville → etc.



À chaque étape, depuis la dernière ville visitée, on visite la ville non-visitée la plus proche.

LA STRATÉGIE GLOUTONNE



Plus formellement, une stratégie gloutonne effectue des **choix localement optimaux définitifs** dans l'espoir qu'ils conduisent à une **solution globalement optimale**. Il ne revient pas sur les choix antérieurs ni n'anticipe les étapes suivantes au moment de prendre une nouvelle décision.



En étant aveugle au futur et têtue sur les choix passés, une approche gloutonne privilégie donc les gains sur le **court terme** plutôt que sur le long terme, c'est une stratégie souvent **intuitive** et simple à mettre en place.



En pratique, l'optimum n'est pas nécessairement atteint, mais on utilise souvent cette approche gloutonne pour confectionner des **heuristiques** (un algorithme qui fournit rapidement une solution réalisable, pas nécessairement optimale) à des problèmes d'optimisation difficile.

$f(x)$

Exemple d'un optimum (ici minimum) **local** et **global** d'une fonction quelconque :

