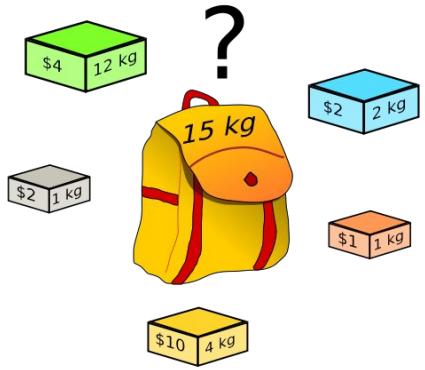

Comparons

les algorithmes gloutons

à d’autres algorithmes

Un algorithme stupide

```python
from random import randint

def valeur(L):
    return L[1]

def poids(L):
    return L[2]

def algoAleatoire(L):
    randL = [x for x in L if randint(0,1)==1]
    pSac = sum([poids(x) for x in randL])
    if pSac <= pMax:
        return randL

pMax = 30
Objets = [("A",20,29),("B",12,10),("C",12,10),("D",12,10),("E",4,1)]

print(algoAleatoire(Objets))
```

```
[('B', 12, 10), ('E', 4, 1)]
```

objets	valeur	poids
A	20	29
B	12	10
C	12	10
D	12	10
E	4	1

Un algorithme (moins) stupide

```python
from random import randint

def valeur(L):
    return L[1]

def poids(L):
    return L[2]

def algoAleatoires(L,n):
    vSacMax = 0
    for i in range(n):
        randL = [x for x in L if randint(0,1)==1]
        vSac = sum([valeur(x) for x in randL])
        pSac = sum([poids(x) for x in randL])
        if (pSac <= pMax and vSac > vSacMax):
            vSacMax = vSac
            optL = randL
    return optL

essais = 100
pMax = 30
Objets = [("A",20,29),("B",12,10),("C",12,10),("D",12,10),("E",4,1)]

print(algoAleatoires(Objets,essais))
```

```
[('B', 12, 10), ('C', 12, 10), ('D', 12, 10)]
```

objets	valeur	poids
A	20	29
B	12	10
C	12	10
D	12	10
E	4	1

Un algorithme (brutalement) efficace

```python
from itertools import combinations

def valeur(L):
    return L[1]

def poids(L):
    return L[2]

def algoForceBrute(L):
    n = len(L)
    vSacMax = 0
    for i in range(n+1):
        for c in combinations(L,i):
            combL = list(c)
            vSac = sum([valeur(x) for x in combL])
            pSac = sum([poids(x) for x in combL])
            if (pSac <= pMax and vSac > vSacMax):
                vSacMax = vSac
                optL = combL
    return optL

pMax = 30
Objets = [("A",20,29),("B",12,10),("C",12,10),("D",12,10),("E",4,1)]

print(algoForceBrute(Objets))
```

```
[('B', 12, 10), ('C', 12, 10), ('D', 12, 10)]
```

objets	valeur	poids
A	20	29
B	12	10
C	12	10
D	12	10
E	4	1

Un algorithme glouton

```python
def valeur(L):
    return L[1]

def poids(L):
    return L[2]

def algoGlouton(L):
    sortL = sorted(L, key=valeur, reverse=True)
    pSac = 0
    optL = []
    for x in sortL:
        if pSac + poids(x) <= pMax:
            optL.append(x)
            pSac = pSac + poids(x)
    return optL

pMax = 30
Objets = [("A",20,29),("B",12,10),("C",12,10),("D",12,10),("E",4,1)]

print(algoGlouton(Objets))
```

```
[('A', 20, 29), ('E', 4, 1)]
```

objets	valeur	poids	valeur/kg
A	20	29	0,69
B	12	10	1,2
C	12	10	1,2
D	12	10	1,2
E	4	1	4

algorithme force brute vs algorithme glouton

```python
def algoForceBrute(L):
    compteur = 0
    n = len(L)
    vSacMax = 0
    for i in range(n+1):
        for c in combinations(L,i):
            combL = list(c)
            vSac = sum([valeur(x) for x in combL])
            pSac = sum([poids(x) for x in combL])
            if (pSac <= pMax and vSac > vSacMax):
                vSacMax = vSac
                optL = combL
            compteur += 1
    return compteur

pMax = 30
Objets = [("A",20,29),("B",12,10),("C",12,10),("D",12,10),("E",4,1)]

print(algoForceBrute(Objets), "boucles")
```

```
32 boucles
```

```python
def algoGlouton(L):
    compteur = 0
    sortL = sorted(L, key=valeur, reverse=True)
    pSac = 0
    optL = []
    for x in sortL:
        if pSac + poids(x) <= pMax:
            optL.append(x)
            pSac = pSac + poids(x)
        compteur += 1
    return compteur

pMax = 30
Objets = [("A",20,29),("B",12,10),("C",12,10),("D",12,10),("E",4,1)]

print(algoGlouton(Objets), "boucles")
```

```
5 boucles
```

algorithme force brute vs algorithme glouton

nombre d'objets (taille de la liste)	nombre de boucles algorithme brute force	nombre de boucles algorithme glouton
5	32	5
10	1024	10
20	1048576	20
40	1099511627776	40

algorithme **force brute**

- ➔ retourne toujours la meilleure solution
- ➔ mais ça peut prendre beaucoup de temps

algorithme **glouton**

- ➔ ne retourne pas forcément la meilleure solution
- ➔ mais ça peut prendre beaucoup moins de temps