

# Informatique - Chapitre 3

## Programmation dynamique

### I. Exemple et résolutions

1. Fonction récursive
2. Programmation dynamique
3. Mémoïsation

### II. Généralisation

### III. Dictionnaires

1. Rappels : définition et utilisation
2. Accès et complexité

### IV. Exemple : le problème du sac à dos

1. Description du problème
2. Algorithme glouton
3. Résolution dynamique

## Le problème du sac à dos : code et résultats

```
1 import numpy as np
2
3 # Problème du sac à dos : algorithme glouton par valeur simple
4 def sacados_glouton_simple(valeurs: [int], masses: [int], Masse_max: int) -> int:
5     # Récupération des objets
6     objets = []
7     for i in range(len(valeurs)):
8         objets.append( [valeurs[i], masses[i]] )
9     # Tri des objets par valeur décroissante (tri selon le premier élément)
10    objets = sorted(objets, reverse = True)
11    print(" Liste des objets classés :", objets)
12    # Remplissage du sac
13    valeur = 0
14    objets_sac = []
15    for objet in objets:
16        if objet[1] <= Masse_max:
17            valeur = valeur + objet[0]
18            Masse_max = Masse_max - objet[1]
19            objets_sac.append(objet)
20    return valeur, objets_sac
21
22 # Problème du sac à dos : algorithme glouton par valeur massique
23 def sacados_glouton_massic(valeurs: [int], masses: [int], Masse_max: int) -> int:
24     # Récupération des objets
25     objets = []
26     for i in range(len(valeurs)):
27         objets.append( [valeurs[i], masses[i]] )
28     # Tri des objets par valeur massique décroissante
29     def valeurmassique(objet):
30         return objet[0] / objet[1]
31     objets = sorted(objets, key = valeurmassique, reverse = True)
32     print(" Liste des objets classés :", objets)
33     # Remplissage du sac
34     valeur = 0
35     objets_sac = []
36     for objet in objets:
37         if objet[1] <= Masse_max:
38             valeur = valeur + objet[0]
39             Masse_max = Masse_max - objet[1]
40             objets_sac.append(objet)
41     return valeur, objets_sac
```

```

1  # Problème du sac à dos : algorithme dynamique avec tableau
2  def sacados_dynamique_tab(valeurs: [int], masses: [int], Masse_max: int) -> int:
3      n = len(valeurs)
4      # Remplissage du tableau de valeurs optimales
5      s = np.zeros( (n+1, Masse_max+1), dtype = int)
6      for k in range(1, n+1):
7          for M in range(1, Masse_max+1):
8              if masses[k-1] <= M:
9                  s[k, M] = max( s[k-1,M], s[k-1,M-masses[k-1]] + valeurs[k-1] )
10             else:
11                 s[k, M] = s[k-1, M]
12             # Affichage du tableau des valeurs calculées
13             print(" Tableau des valeurs calculées, de taille",(n+1), "*", (Masse_max+1))
14             print(s)
15             # Récupération des différents objets
16             M = Masse_max
17             objets = []
18             for k in range(n, 0, -1):
19                 if s[k-1, M] < s[k, M]:
20                     objets.append( [ valeurs[k-1], masses[k-1] ] )
21                     M = M - masses[k-1]
22             return s[n, Masse_max], objets
23
24 # Problème du sac à dos : algorithme dynamique avec dictionnaire
25 def sacados_dynamique_dico(valeurs: [int], masses: [int], Masse_max: int) -> int:
26     s = {}
27     n = len(valeurs)
28     # Fonction récursive appliquant l'équation de Bellman
29     def calcul(k, M):
30         if (k,M) not in s:
31             if k == 0 or M == 0:
32                 x = 0
33             elif masses[k-1] <= M:
34                 x = max(calcul(k-1,M), calcul(k-1,M-masses[k-1]) + valeurs[k-1])
35             else:
36                 x = calcul(k-1,M)
37             s[ (k,M) ] = x
38             #print(" Nouveau calcul avec k =",k,"et M =",M,": s =", s[(k,M)])
39         #else:
40             # print(" Pas de modification pour k =",k,"et M =",M,": s =",s[(k,M)])
41         return s[ (k,M) ]
42     # Calcul
43     val = calcul(n, Masse_max)
44     # Affichage du dictionnaire des valeurs calculées
45     print(" Dictionnaire des valeurs calculées, de taille",len(s))
46     print(s)
47     # Récupération des différents objets
48     M = Masse_max
49     objets = []
50     for k in range(n, 0, -1):
51         if s[ (k-1,M) ] < s[ (k,M) ]:
52             objets.append([ valeurs[k-1], masses[k-1] ])
53             M = M - masses[k-1]
54     return val, objets

```

```

1 # Problème du sac à dos : résolution d'un exemple et comparaison des méthodes
2 valeurs = [ 12, 4, 10, 6, 8, 3 ]
3 masses = [ 6, 2, 4, 2, 3, 2 ]
4 masse_sac = 8
5 print("*** Algorithme glouton par valeur simple")
6 s, objets = sacados_glouton_simple(valeurs, masses, masse_sac)
7 print(" Liste des objets sélectionnés : ", objets)
8 print(" Valeur totale dans le sac : ", s, "\n")
9 print("*** Algorithme glouton par valeur massique")
10 s, objets = sacados_glouton_massic(valeurs, masses, masse_sac)
11 print(" Liste des objets sélectionnés : ", objets)
12 print(" Valeur totale dans le sac : ", s, "\n")
13 print("*** Algorithme dynamique avec tableau")
14 s, objets = sacados_dynamique_tab(valeurs, masses, masse_sac)
15 print(" Liste des objets sélectionnés : ", objets)
16 print(" Valeur totale dans le sac : ", s, "\n")
17 print("*** Algorithme dynamique avec dictionnaire")
18 s, objets = sacados_dynamique_dico(valeurs, masses, masse_sac)
19 print(" Liste des objets sélectionnés : ", objets)
20 print(" Valeur totale dans le sac : ", s)

```

## Résultats

\*\*\* Algorithme glouton par valeur simple

Liste des objets classés : [[12, 6], [10, 4], [8, 3], [6, 2], [4, 2], [3, 2]]

Liste des objets sélectionnés : [[12, 6], [6, 2]]

Valeur totale dans le sac : 18

\*\*\* Algorithme glouton par valeur massique

Liste des objets classés : [[6, 2], [8, 3], [10, 4], [12, 6], [4, 2], [3, 2]]

Liste des objets sélectionnés : [[6, 2], [8, 3], [4, 2]]

Valeur totale dans le sac : 18

\*\*\* Algorithme dynamique avec tableau

Tableau des valeurs calculées, de taille 7 \* 9

[[ 0 0 0 0 0 0 0 0 0 ]

[ 0 0 0 0 0 0 12 12 12 ]

[ 0 0 4 4 4 4 12 12 16 ]

[ 0 0 4 4 10 10 14 14 16 ]

[ 0 0 6 6 10 10 16 16 20 ]

[ 0 0 6 8 10 14 16 18 20 ]

[ 0 0 6 8 10 14 16 18 20 ]]

Liste des objets sélectionnés : [[6, 2], [10, 4], [4, 2]]

Valeur totale dans le sac : 20

\*\*\* Algorithme dynamique avec dictionnaire

Dictionnaire des valeurs calculées, de taille 37

{(0, 8): 0, (0, 2): 0, (1, 8): 12, (0, 6): 0, (0, 0): 0, (1, 6): 12, (2, 8): 16,  
 (0, 4): 0, (1, 4): 0, (1, 2): 0, (2, 4): 4, (3, 8): 16, (2, 6): 12, (1, 0): 0,  
 (2, 2): 4, (3, 6): 14, (4, 8): 20, (0, 5): 0, (1, 5): 0, (0, 3): 0, (1, 3): 0,  
 (2, 5): 4, (0, 1): 0, (1, 1): 0, (2, 1): 0, (3, 5): 10, (2, 3): 4, (3, 3): 4,  
 (4, 5): 10, (5, 8): 20, (2, 0): 0, (3, 4): 10, (4, 6): 16, (3, 1): 0, (4, 3): 6,  
 (5, 6): 16, (6, 8): 20}

Liste des objets sélectionnés : [[6, 2], [10, 4], [4, 2]]

Valeur totale dans le sac : 20