SÉANCE 5

# QUELQUES EXEMPLES DE PROBLÈMES D'OPTIMISATION

```
## Exemple 1

def gain1(a):
    g = 0
    for i in range(len(a)):
        for j in range(i+1, len(a)):
            if a[j]-a[i] > g:  # meilleure affaire ?
                g = a[j]-a[i]
    return g


def gain1_bis(a):
    g = 0
    i0, j0 = 0, 0 # dates de la meilleure vente
    for i in range(len(a)):
        for j in range(i+1, len(a)):
            if a[j]-a[i] > g: # meilleure affaire ?
                g = a[j]-a[i]
                i0, j0 = i, j
            elif a[j]-a[i] == g and j-i < j0-i0: # durée plus courte ?
                i0, j0 = i, j
    return(g, i0, j0)


''' Si la chute du taux i par rapport au taux i-1 est supérieure
        au gain courant en i-1 alors le gain courant en i est nul:
            gainCourant_i = max (0 , gainCourant_{i-1} + a[i] - a[i-1] ).
    Dans un tour de boucle, on tient constamment à jour le gain courant
        et on garde trace dans une variable du meilleur gain courant. '''


def gain2(a):
    gc = 0 # gain courant
    g = 0  # gain i.e. le max des gains courants
    for i in range(1,len(a)):
        if gc+a[i]-a[i-1] > 0:
            gc = gc + a[i]-a[i-1]
        else:
            gc = 0
        if gc > g:
            g = gc
    return g
```

```python
def gain2_bis(a):
    gc = 0
    g = 0
    i = 0 # indice du prix minimum
    i0, j0 = 0, 0
    for j in range(1,len(a)):
        if gc + a[j] - a[j-1] > 0:
            gc += a[j] - a[j-1]
        else:
            gc = 0
            i = j
        if g < gc :
            g = gc
            i0, j0 = i, j
        if g == gc and j-i < j0-i0:
            i0, j0 = i, j
    return (g, i0, j0)



## Exemple 2


def monnaie_a_rendre (somme_a_rendre, systeme):
    """ entrée : int somme_a_rendre, list d'int systeme
        détermine le nombre de pièce pour chaque type de pièce de systeme
            afin d'obtenir somme_a_rendre
        utilise pour cela un algorithme glouton :
            on commence par les pièces de plus grande valeur
        sortie : list d'int correspondant au nombre de chaque pièce
    """

    liste_monnaie = [0 for _ in systeme]  # liste de 0 de même longueur que systeme
    k = len(systeme) - 1

    while somme_a_rendre > 0 and k >= 0:
        nombre_pieces = somme_a_rendre // systeme[k]
        somme_a_rendre = somme_a_rendre % systeme[k]
        liste_monnaie[k] = nombre_pieces
        k = k - 1

    return liste_monnaie


assert monnaie_a_rendre(8, [1, 4, 6]) == [2, 0, 1]
assert monnaie_a_rendre(49, [1, 3, 6, 12, 24, 30]) == [1, 0, 1, 1, 0, 1]
assert monnaie_a_rendre(11, [2, 4, 6]) == [0, 1, 1]
```