

Rendu de monnaie

On suppose disposer de différentes pièces dont les valeurs sont stockées, dans l'ordre décroissant, dans une liste `Pieces`. On souhaite trouver le moyen de rendre une certaine somme `s` en utilisant le moins de pièces possibles.

On supposera que le stock de pièces disponible est infini (et donc on n'aura toujours toutes les pièces disponibles pour ce rendu de monnaie).

On supposera également que la plus petite pièce possède une valeur 1 ce qui fait que le rendu de monnaie sera toujours possible.

1. Approche gloutonne

L'approche gloutonne est définie ainsi : pour rendre la monnaie, on commence par rendre la pièce dont la valeur est la plus grande possible (et on recommence sur la somme restante)

1. Écrire une fonction récursive `glouton(s:int,P:list)->list` qui prend en argument la somme `s` à rembourser (supposée ≥ 1) et la liste `P` des valeurs des pièces, et qui renvoie la liste des pièces à utiliser pour ce remboursement.
2. Que donne cette approche dans le cas où `s = 49`, lorsque les pièces sont `Pieces = [10,5,2,1]` puis lorsque les pièces sont `Pieces = [30,24,12,6,3,1]` ?

2. Programmation dynamique

3. Si on note N_s le nombre de pièces minimal nécessaire pour rembourser la somme `s` et $p_1, \dots, p_k$ la valeurs des pièces disponibles. Justifier que

$$N_s = 1 + \min\{N_{s-p_i}, i \in \llbracket 1, k \rrbracket, s - p_i \geq 0\}$$

4. Écrire une fonction récursive `rendu(s:int,P:list)->int` qui prend en argument la somme `s` à rembourser (supposée ≥ 1) et la liste `P` des valeurs des pièces, et qui renvoie le nombre de pièces minimal nécessaire pour rendre la somme `s`.
5. Rendre la fonction précédente plus efficace en utilisant la mémorisation avec un dictionnaire.
6. Écrire une fonction `BasEnHaut(n:int,P:list)->int` qui effectue le même calcul que la fonction précédente mais en allant de bas en haut (programmation itérative) : on remplira une liste `L` de sorte que `L[i]` soit le nombre de pièces à utiliser pour rendre la somme `i`.
7. Évaluer la complexité de cette dernière fonction.
8. On suppose que la liste remplie par la fonction `BasEnHaut`, avec le système de pièces `[4,3,1]`, est

$$[0, 1, 2, 1, 1, 2, 2, 2, 2, 3, 3].$$

Quelles sont les pièces à rendre pour rembourser 10 ?

9. Écrire une fonction `Rendu(s:int,P:list)->list` qui renvoie la liste des pièces à rendre pour rembourser la somme `s`.
10. Proposer une modification de la fonction `rendu`, de la question 5, lorsque le système de pièces ne comporte plus la pièce 1, ce qui amène à des sommes impossibles à rembourser.