
ITC 2° ANNÉE - TP 2 : MÉMOISATION & PROGRAMMATION DYNAMIQUE DESCENDANTE

Fichier fourni

Vous trouverez sur <https://sbautista.fr/itc2/> un fichier `itc2_tp_2_squelette.py` qui contient :

- Une fonction `fibonacci_rec(n)` qui calcule le $n^{\text{ième}}$ terme de la suite de Fibonacci de façon récursive, sans mémoïsation
- Une fonction `fibonacci_memo` qui est une version mémoïsée de `fibonacci_rec`
- Des fonctions `rendu_rec`, `grecoille_rec` et `pyramide_rec` qui serviront pour les exercices 1, 2 et 3 respectivement

Exercice 1 (Rendu de monnaie)

1. Proposer une fonction `rendu_memo` qui résout le même problème que `rendu_rec`, mais en utilisant de la programmation dynamique descendante.
2. Vérifiez que l'appel `rendu_memo(79, [1, 3, 6, 12, 24, 30])` finit en un temps raisonnable et renvoie bien 4.

On rappelle le problème algorithmique résolu par `rendu_rec` :

Entrées :

- Un entier naturel `a_rendre` de centimes à rendre
- Une liste `valeurs_pieces` d'entiers naturels représentant les différentes valeurs possibles de chaque pièce qu'on peut utiliser pour rendre la monnaie.

Sortie : Le nombre minimal de pièces à utiliser pour rendre `a_rendre` centimes.

Exercice 2 (Le problème de la grenouille)

1. Proposer une fonction `grenouille_memo` qui résout le même problème que `grenouille_rec`, mais en utilisant de la programmation dynamique descendante.
2. Vérifiez que l'appel `grenouille_memo(exemple_hauteurs)` renvoie bien 6
3. Vérifiez que l'appel `grenouille_memo(liste_alea(900))` finit en un temps raisonnable.

On rappelle le problème algorithmique résolu par `grenouille_rec` : On s'imagine n pierres alignées, ayant différentes hauteurs $h_0, \dots, h_{n-1}$. Une grenouille commence à la pierre d'indice 0 et veut se rendre jusqu'à la pierre d'indice $n - 1$. À chaque étape, si la grenouille est sur la pierre d'indice i , elle peut

- soit se rendre sur la pierre d'indice $i + 1$ avec une coût $|h_{i+1} - h_i|$
- soit se rendre sur la pierre d'indice $i + 2$ avec un coût $|h_{i+2} - h_i|$

Entrée : une liste `hauteurs` contenant des entiers qui représentent les hauteurs des différentes pierres

Sortie : Le coût minimal en lequel la grenouille peut aller de la pierre 0 à la pierre $n - 1$

Exercice 3 (Somme maximale descendant une pyramide d'entiers)

1. Proposer une fonction `pyramide_memo` qui résout le même problème que `pyramide_rec`, mais en utilisant de la programmation dynamique descendante.
2. Vérifiez que l'appel `pyramide_rec(ex_pyramide_1)` renvoie bien 30
3. Vérifiez que l'appel `pyramide_rec(pyramide_alea(100))` finit en un temps raisonnable.

Voici le problème algorithmique résolu par `pyramide_rec`.

On considère une pyramide d'entiers, comme celle illustrée en fin de sujet.

On appelle *chemin descendant une pyramide d'entiers* un chemin qui commence au sommet de la pyramide, et qui descend un par un les étages de la pyramide, en allant un cran à gauche ou un cran à droite pour chaque étage descendu.

On appelle *somme* le long d'un chemin descendant la pyramide, la somme de tous les entiers par lequel passe le chemin.

Entrée : La représentation d'une pyramide d'entiers

Sortie : La plus grande somme d'un chemin descendant la pyramide

