TP 5
Mémoïsation

Master SID 1 SD
Benoist GASTON
benoist.gaston@univ-rouen.fr

TP06 Ex01 Fibonacci Recursive

Le notebook `M1SD-HPC-TP05-Memoisation-Ex01-Fib.ipynb` définit les fonctions `fibo_iter` et `fibo_rec` qui calculent la $n^{ième}$ valeur de Fibonacci en appliquant respectivement l'algorithme itératif et l'algorithme récursif.

1. Mesurer les temps de restitution de `fibo_iter` et `fibo_rec` pour des valeurs de n (10, 20, ...).

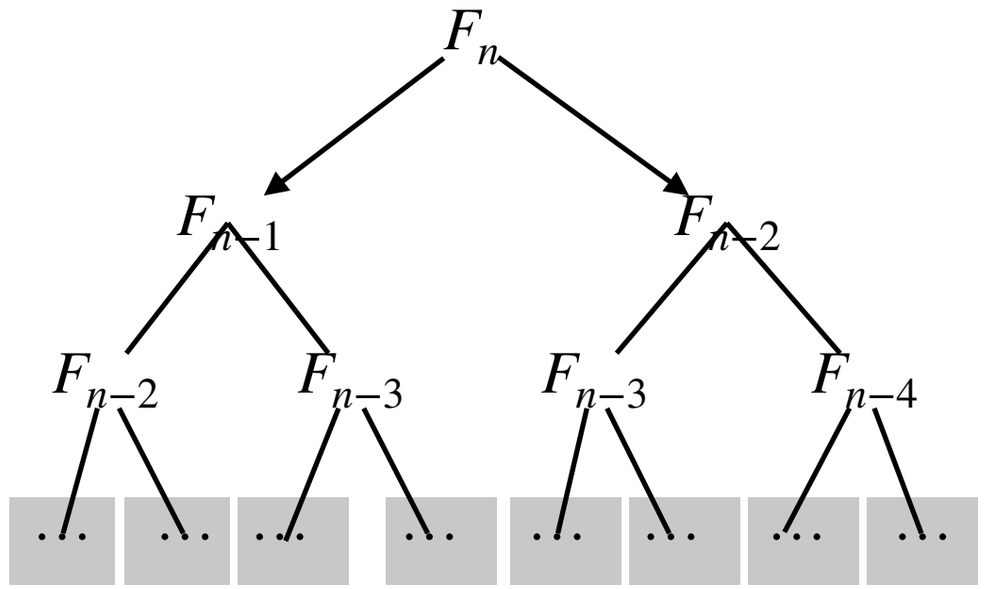
Mémoïsation

La *mémoïsation* est une technique qui consiste à stocker les résultats d'appels de fonctions déjà calculés afin d'éviter de les recalculer le cas échéant.

2. Sur la base de `fibo_rec`

    1. Ecrire une fonction `fibo_memo` qui utilise un dictionnaire pour stocker les différentes valeurs de fibonacci calculées lors des appels récursifs. Mesurer et comparer les temps de restitution.

    2. Le package `functools` de python propose une solution de memoïsation à l'aide du décorateur `lru_cache`. Ecrire une fonction `fibo_lru` qui utilise ce décorateur. Mesurer et comparer les temps de restitution.



TP06 Ex01 Fibonacci Recursive

Numba

1. Compilation à la volée.

    1. Utiliser le décorateur `njit` pour compiler `fibo_iter` et `fibo_rec` à la volée. Comparer les temps de restitution.

    2. tenter une version *jitté* de `fibo_memo`. Quel est le problème. Contournez le en utilisant un ndarray ou bien un objet de type `Dict` de `numba.typed`. Comparer les temps de restitution.

Multithreading

2. Sur la base de la fonction **non** *jitté* `fibo_rec`.

    1. Ecrire une fonction récursive `fibo_thread` où chaque appel récursif est une tâche soumise à un thread d'un `ThreadPoolExecutor` via la méthode `submit`.

    2. Exécuter cette fonction pour des valeur de `n = 2, 3, 4, 5, 6, ....` Quelles observations/ problèmes ?