SOLUTION Week 03 Prac – Recursion and Generator Functions

Last modified on Wednesday, 9 March 2022 by f.maire@qut.edu.au

Recursion is a key tool in AI. Many problems are approached by reducing a given problem to a set of smaller problems. The combination of recursion and caching is known as *Dynamic Programming* (DP). We will devote a whole week to DP as it is at the core of many algorithms in AI.

This prac focuses on recursion and how generator function help implement recursion effectively.

Exercise 1

- Implement a recursive function to compute the nth element of the **Fibonacci sequence**. Recall that The Fibonacci Sequence is the series of numbers: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, …
  The next number is found by adding up the two numbers before it.

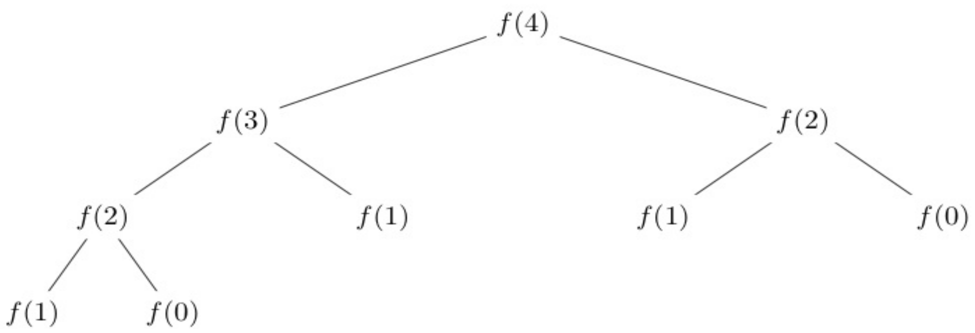
  The signature of the function is

  ```
  def fibo(n):
      '''
      Return the nth element of the Fibonacci sequence
      '''
  ```

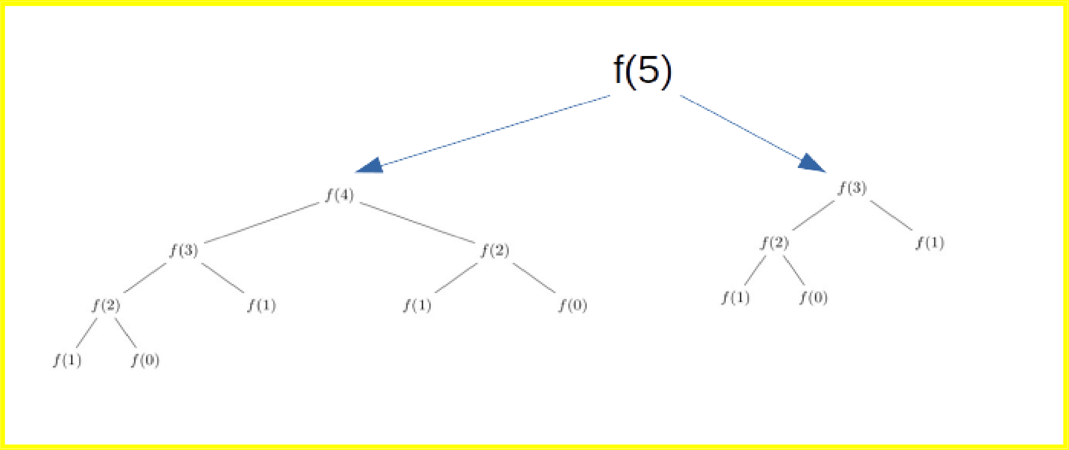
  See file SOLUTION_fibo_gen_fn.py

- Call *fibo(10), fibo(20), fibo(35)* and *fibo(40)*
  What do you observe?
  The running time grows exponentially. For n = 40, be very patient!

  Below is recursion tree for *fibo(4)*.



- Draw the tree of recursive function calls for *fibo(5)*.



- What do you observe? Are there repeated subtrees?

The subtree rooted with f(3) is repeated. Same observation for f(2) with 3 occurences

- Propose a method to avoid these repeated computations.

We could cache the computed values in a dictionary to avoid repeating the computation.

The Python **functools** library does exactly that with `@functools.lru_cache`

Exercise 2

Write a **generator function** to iterate over the Fibonacci sequence.

Test your generator function and compare its results with those of your function from Exercise 1.

See file SOLUTION_fibo_gen_fn.py

Exercise 3*

This exercise is harder. You will manipulate nested tuples.

- Download the file *tuple_max.py*
- Implement the function *get_max* to compute the maximum value *v* of a nested tuple and the index sequence *I* to reach this value. For example. if the tuple is

  T = ((-3, (-6,), 3), -9, ((-8, -6, 9, -5), (-3,), -2)) then v, I = 9, [2, 0, 2]

  See file SOLUTION_tuple_max.py

Exercise 4*

- This exercise is not trivial because the generator function *gen_satistactory_assignments* is itself recursive.
- Download the file *rec_gen_fn.py*

- Analyse how the toy CSP is declared and solved.
- Mimicking how the toy problem is solved, created the relevant data structures *my_zebra_variables*, *my_zebra_domains* and a constraint function *my_zebra_constraint_fn*
- Solve the zebra puzzle with the *md_constraint_search* function.
- Check that you obtain the same values as in the prac of Week 02!
- See file SOLUTION_rec_gen_fn.py