Euler Project n°167

MERCADAL François

Problem link: https://projecteuler.net/problem=167
Github link: https://github.com/FrancoisMERCADAL/Euler_Problem_167

Problem presentation

This problem aims at calculating the sum different Ulam sequences.

$$\sum U(2,2n+1)_k \text{ for } 2 \leq n \leq 10, \text{ where } k = 10^{11}.$$

An Ulam sequence is a number sequence written as $U(a,b)_k$. Depending on k value, the sequence will take different values by following 3 rules:

- if $k = 1$, $U(a,b)_1 = a$
- if $k = 2$, $U(a,b)_2 = b$
- if $k > 2$, $U(a,b)_k$ is the smallest integer greater than $U(a,b)_{(k-1)}$ and with one way to write it as a sum of previous integer in the sequence

The technical challenge

The challenge here is to be able to calculate the sum presented earlier.

The main difficulty can be summed up by $k = 10^{11}$ which means that the computer has to perform more than 10^{11} operations in an O(n) case.

However this scenario is very expensive in terms of time and resources.

Thus, the challenge is to find a way to make all this process faster to make it able to calculate all the sequences' terms.

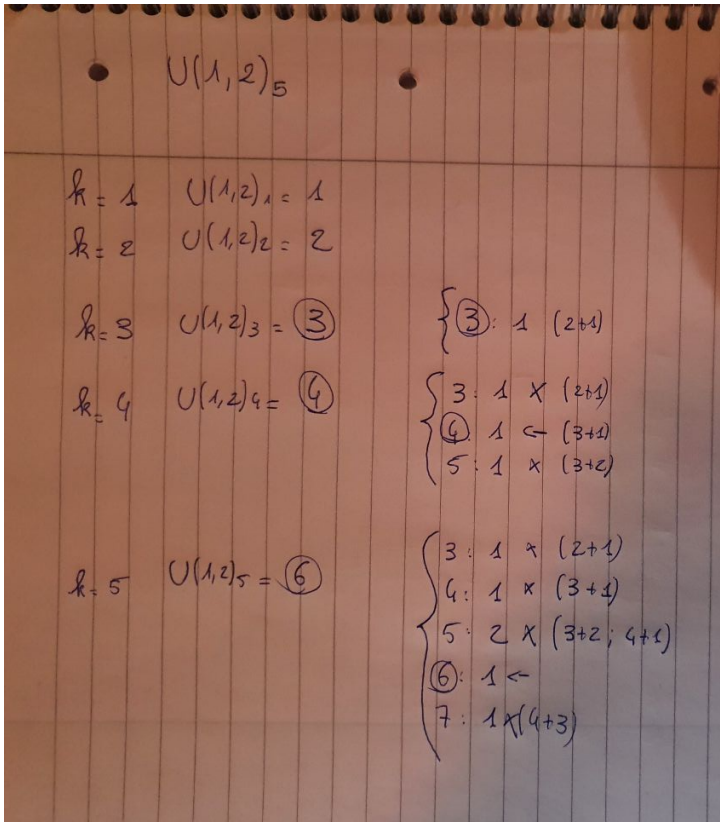
My approach

For k > 2, my approach consists at editing a dictionary (one dictionary per Ulam sequence) which contains all the possible numbers as keys and a count for each one which represents the number of possible ways to calculate it.

Then a function inspects that dictionary and returns the next value to append in the Ulam Sequence.

At the end when all the sequences are completed (one by one) a last function sums all of them into a final list.

My approach (with the given example in the problem)



My approach (pseudo code for one Ulam sequence $U(a,b)_k$)

```
list = []
dict = {}
for 0 < i < k:
    if i == 1:
        list.append(a)
    elif i == 2:
        list.append(b)
    elif k > 2:
        update the dictionary (as shown in the previous slide)
        select the value to apend in the list and return it
        append the new value to the list
return list
```

Problems encountered

The main problem I encountered is the processing time: mine is way too long!

Here I computed 10^9 “continue” commands in a for loop: it takes around 45 seconds for my computer to compute it.

```
t0= time.clock()
for i in range(1000000000):
    continue
t= time.clock() - t0
print("Runtime: " + str(t))
```

```
Runtime: 44.1856231
```

Problems encountered

So it is easy to understand that computing 10^{11} numbers with a for loop will take a way bigger time.

I estimated that my computer can solve the problem in 10 hours with the code I wrote.

So I think that my strategy isn’t the right one yet.