Project Euler – Problem 74 Solution

Digit Factorial Chains

The first step to solve this problem is to compute the sum of the digits of the number N.
For example if N = 136, firstly we need to find the digits of the number N (that is 1, 3 and 6), compute the corresponding factorials 1! = 1, 3! = 6, 6! = 720 and then summing them up to the result 724. All this computation is done by the method ***sumFact(int)***.

By repeating this procedure on every new number, we end up creating a chain. The main goal, thus, of this problem is to compute the length of this chain. Hence two questions arise:

- How can we deduce that a chain is formed?
- How can we compute the length of the chain? (i.e. the cardinality of non-repeating terms)

Both questions can be answered and solved, thanks to the **ArrayList** data structure provided by java, since searching an element in the list and the size of the arraylist can be both computed in constant time (java uses hashing for this purpose).

Concretely for every new number N we create a new ArrayList and add the number N in the list. Then for every new element produced by the function sumFact(int), we search it in the list and if it is not found, then it is added. Once an element is found that is not contained in the list then we come to the conclusion that the chain is now formed and the algorithm for number N stops. That being said, we do the same procedure for all the numbers between one and one million.

Thus, we can easily find the number of elements that have sixty non-repeating terms by just checking for every number N if the corresponding arraylist contains exactly sixty elements.