CO 322 Data Structures and Algorithms

Lab 01

09th of July 2020

Name	: A.L.V.H. Dharmathilaka
Registration Number	: E/16/086

The run time of an algorithm is an important factor. In order to predict the runtime usually we want to know how many operations an algorithm will execute in proportion to the size of its input. In this problem we consider two different implementations of a function that calculates the given Fibonacci number and get most accurate data from code implementations. It is better to remind that Fibonacci numbers are sequence of numbers which starts with 0 and 1.Then the each successive number in the sequence is the summation of previous two Fibonacci numbers.

When the problem is small, the runtime of the function which uses recursive and the runtime which uses iterations have not a significant amount of difference in both languages. It is shown in graphs in Figure 1.0 and Figure 2.0. That small range can be identified from 0 to 25. After that range, runtime of the recursive method is increasing rapidly in both languages according to their time range but the runtime of both iteration methods increasing with a slight slop only.

We implemented two functions in java language and python language as mentioned earlier .According to the runtime, the python implementation consumes more time than the java implementation. When the problem size is increasing the runtime difference can be clearly identified. As an example the runtime ratio of finding 40th Fibonacci number using recursive method with python and java implementation is nearly 250:3 .Even the problem size is small this runtime difference is identified but not in similar ratio which we get in fourteenth Fibonacci number . In the finding of first Fibonacci number, the runtime ratio of python to java is close to 15717:7 in recursive method. After analyzing the data in figure 1.0 and figure 2.0 there is also a similarity in both python and java implementations. That is although the runtime amount is different, the increment of runtime has slight change after some problem size according to the figures. Then the runtime is dramatically increased in different rates for java and python implementations of recursive function. In iteration method also java implementation is faster than runtime of python implementation but both has similar behavior for runtime variation as shown in Figure3.0 and Figure4.0.

Generally the difference between these two languages is that Java is a statically typed and Python is a dynamically typed. So in python names in code are bound to strongly typed objects in runtime. It supports the operations required for the particular object instances in the program more likely to real world object usage. Though this makes hard to analyze the code, the writing is easy and readability also in good level. On the other hand in java names are bound to types at compile time via explicit type declaration which tend to have more errors in compilation and the code will not compile until errors have been fixed but the telling what type of object is that variable makes readers to understand the type of it easily. In a simple manner python perform type checking at runtime but the java perform type checking at the compile time. Java Virtual Machine (JVM) makes the execution faster through just in time (JIT) compilation. That JIT compiler compiles the bytecode to native machine code quickly.

Concepts of recursive function and iteration function has some differences in their functionality. The recursive functions is calling the same function again and again by itself. So the number of recursive calls happened affects in runtime. The runtime become large when the considerable amount of recursive calls are done because in the Fibonacci recursive function, for finding nth Fibonacci number two another function itself calls must be done according to the nature of the code. Because of that, time taken for a recursive function of finding Fibonacci number can be calculated as O(2^n) or exponential. Iteration method for finding Fibonacci number repeat a block of code. The number of repetitions will decide the runtime. The time complexity of it is linear since loop run from two to n. It can be calculated as O(n).

According to my point of view, it is fair to say that for small scale of problems both of these algorithms can be useful. Actually Fibonacci number pattern is naturally fit to the recursive method. It makes the code very short and increases the readability. On the other hand though the iteration code is comparatively lengthy, its runtime is really small in amount. For small problems that runtime is very similar to the runtime of recursive method. These reasons prove the correctness of above mentioned sentence about small scale problems.

For bigger problem size the n is increasing and recursive method gives noticeable runtime increase. The rapid increase of Figure 1.0 and Figure2.0 of java and python implementations clearly demonstrated this difference. So it is fair enough to say that generally recursive function for Fibonacci number is not useful for large size problems but it is advantageous in small size problems. It is important to say that java implementation for recursive method is more useful than the python recursive method comparatively because it gets only nearly 0.3 seconds to get the 40th Fibonacci number. In python it gets more than 17.5 seconds. So mainly the problem of efficiency is occurs when write recursive function in python language.

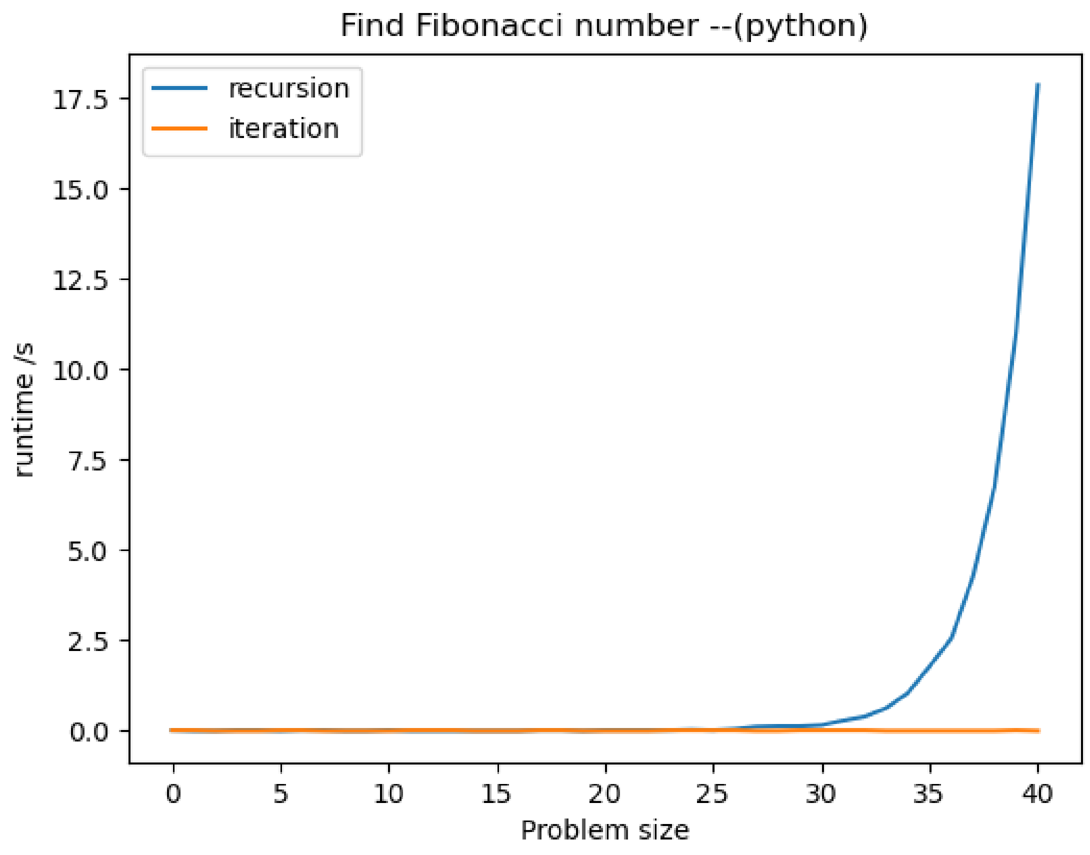

Figure 1.0 : Find Fibonacci number using python implementation

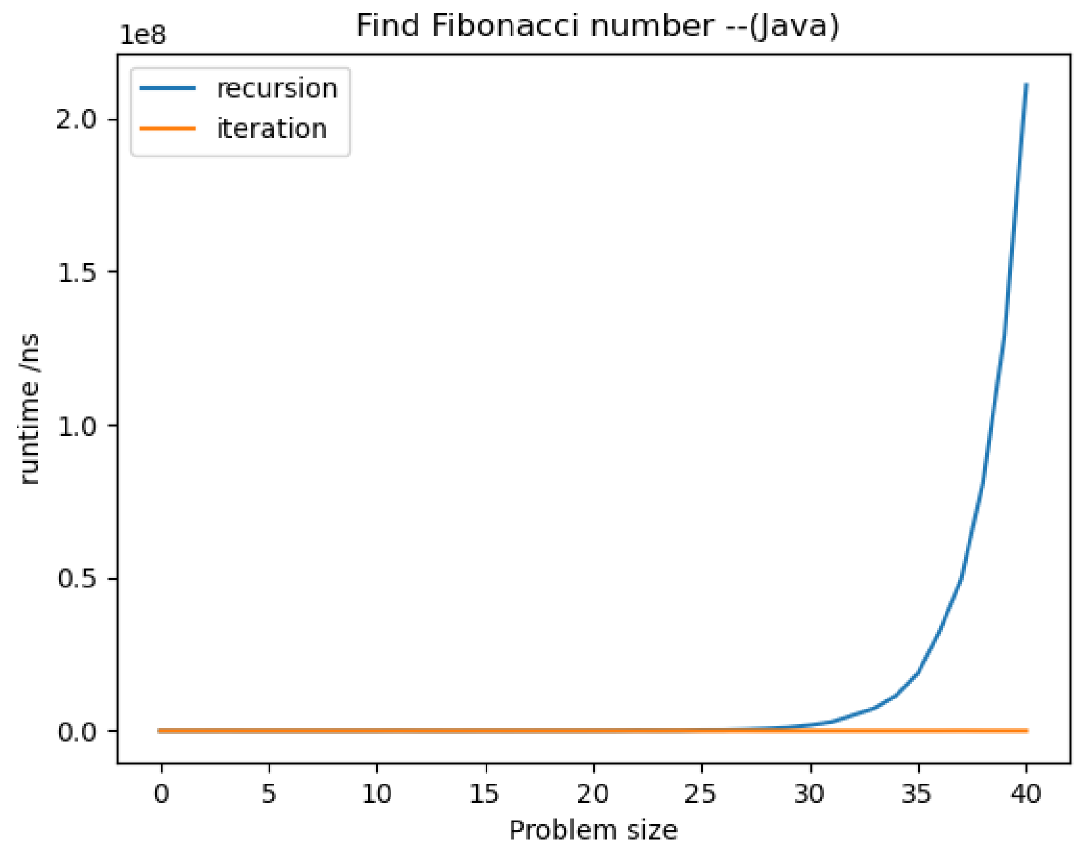

Figure 2.0 : Find Fibonacci number using java implementation

To clearly show the runtime in iteration method, they are plotted again in separate graphs.

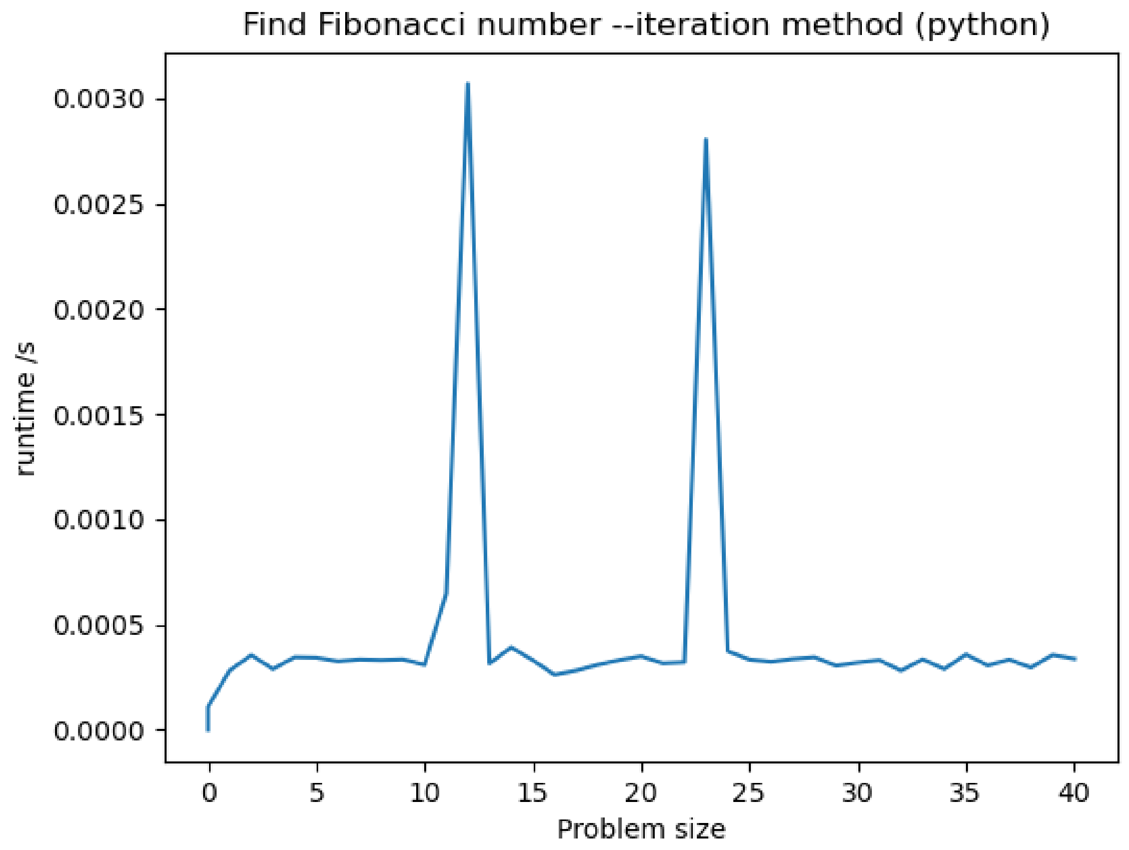

Figure 3.0 : Find Fibonacci number using iteration function (python implementation)

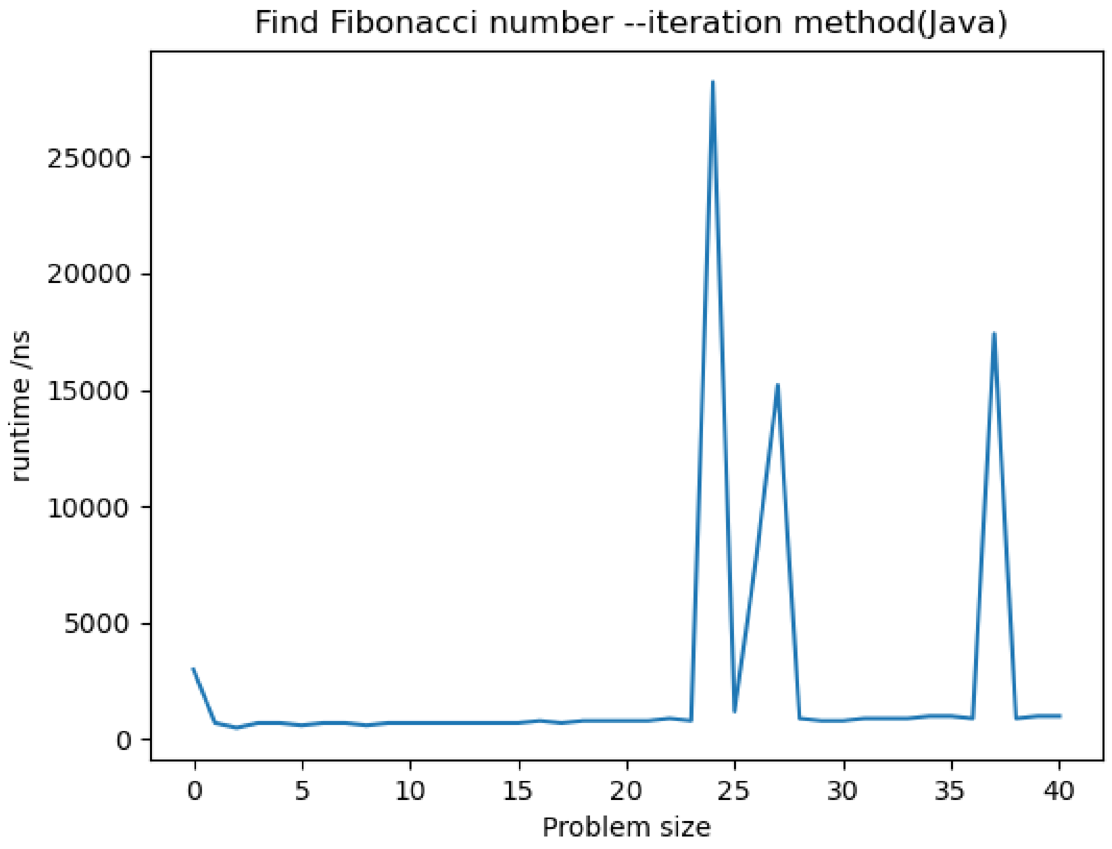

Figure 4.0 : Find Fibonacci number using iteration function (java implementation)