E/16/388

CO 322

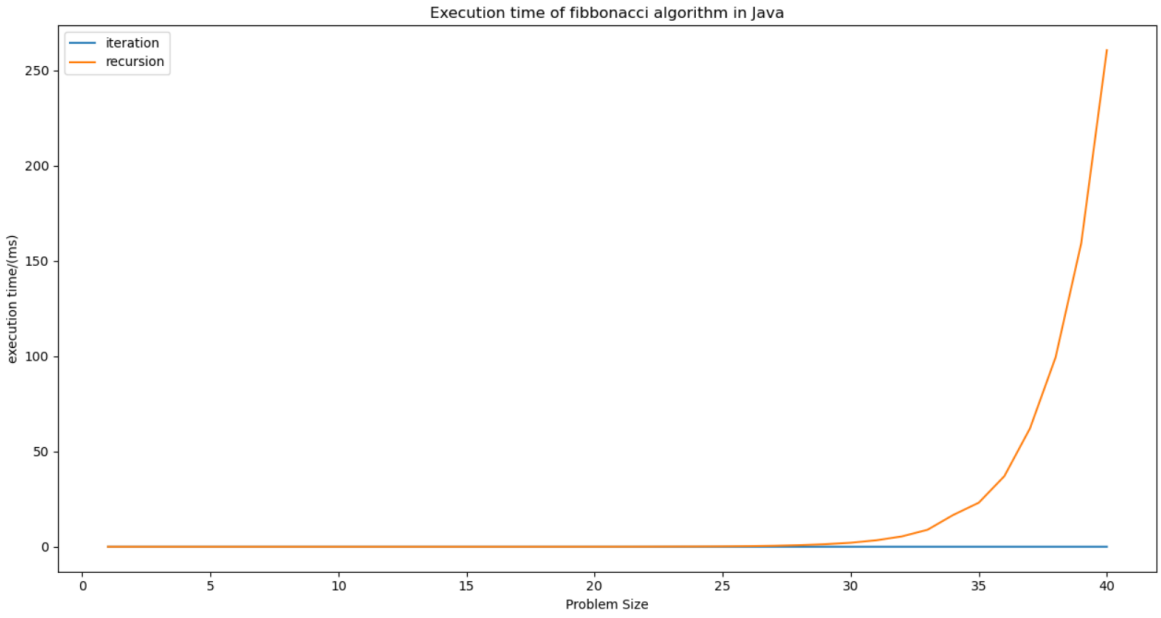
Lab 1 eassay questions

(a) Is there a difference in the runtime between the two implementations when the problem is small?

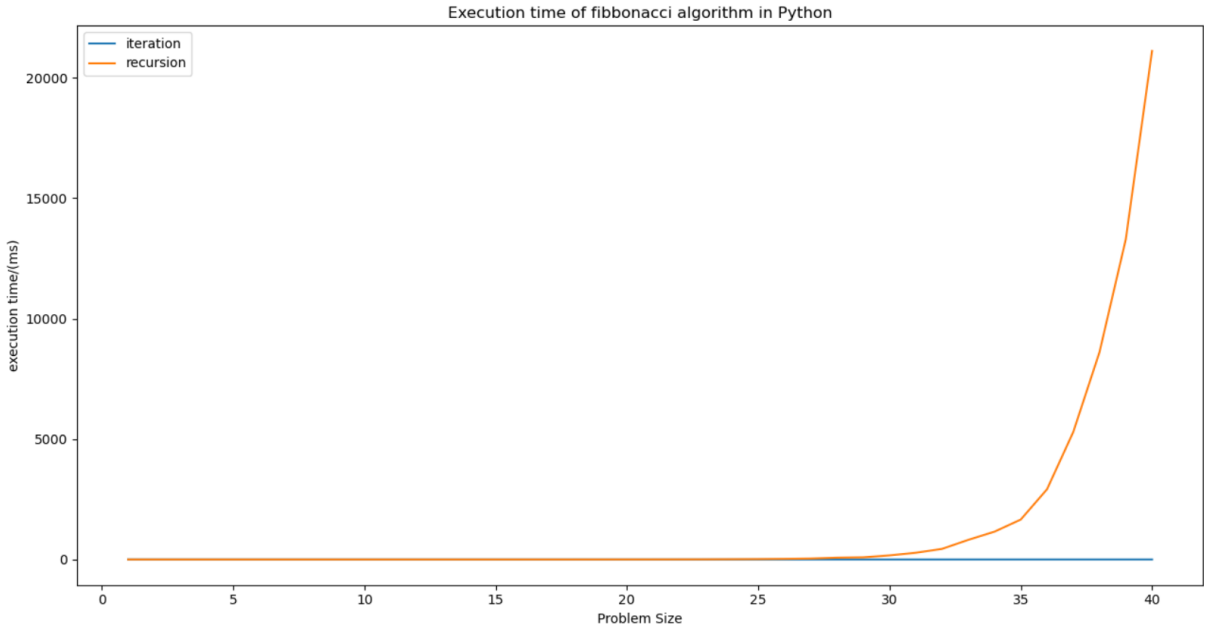
In both implementations, runtime for the recursive algorithm increases significantly with the problem size when comparing to the iterative algorithm. When considering for small problem sizes, in iteration algorithm for both python and java implementations there are very mall differences in the runtime therefore it is negligeble. The difference is increasing with the problem size but that too is a small difference. Considering recursive algorithm for both languages, when the problem size is small, time difference is small as it were in iteration algorithm. With the problem size increasing that difference is rapidly increased. Still the java implements take slightly lesser time than the python implementations.

Graph representations,

For Java, **Fig 1**



For Python, **Fig 2**



(b) Is there a difference in the runtime between the two languages?

When comparing runtimes for python and java using same algorithms and same problem sizes we can detect a runtime difference. Algorithm using java takes less runtime than python. The main reason could be, python is an interpreted language, therefore it determines the type of data at runtime. Java takes less runtime because it is a compiled language and it compiles to native, therefore it fully compile before run. Therefore takes less time to execute a program. We could see the differences in the runtime graphically when we plot all 4 graphs in a single graph.

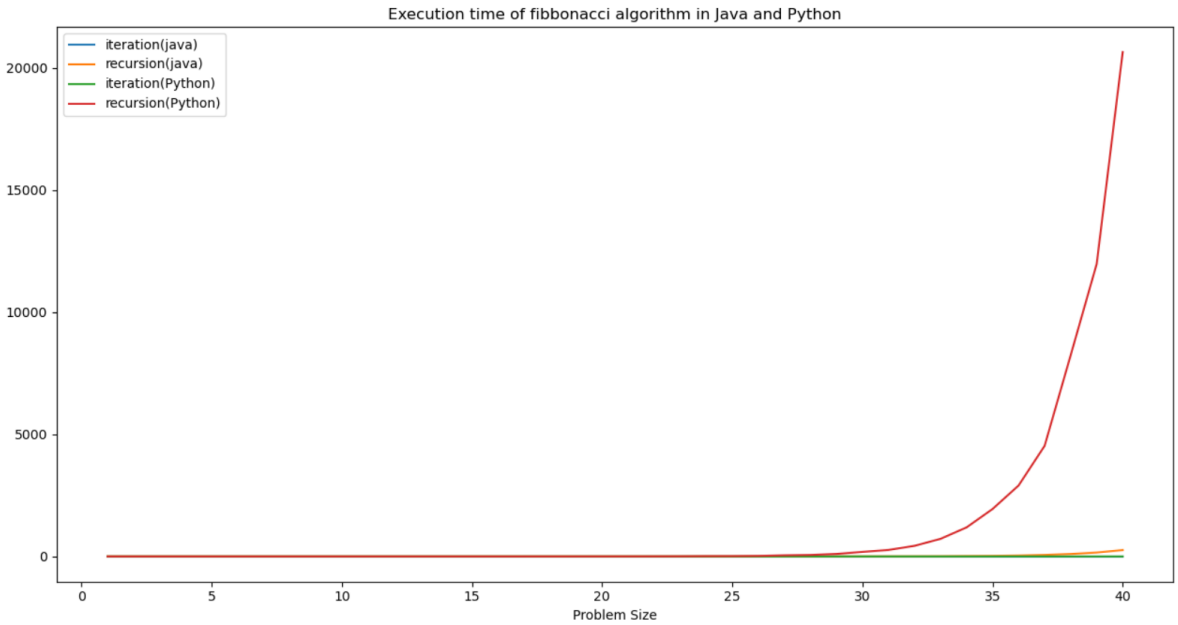


Fig 3

(c) Is there a difference between the way the runtime changes in the two languages?

When comparing the plots of runtime for both languages, we can see both algorithms behave in a simillar way. In iterative algorithm, runtime diffeence is increasing slowly (linear runtime distribution) with the problem size but recursive algorithm runtime increases in a expotential way. Therefore we can determine that the behaviour of the runtime is independent of the language but depends on the algorithm. Language only effects the value of the runtime. We can observe this in **Fig 3** graph.

(d) “If the problem is small both algorithms are useful”. Do you agree with this statement? Justify your answer.

When comparing the behaviour of the graphs, we can see for a certain language when problem size is approximately under 25 runtime is in the same range for recursion and iteration. Those values can be small differences in the value but when representing them in a graph **(Fig 3**), they fall into same range and has small differences. Therefore we can say when the problem size is small, both algorithms are useful.

(e) “If the problem is large fib r is not useful”. Do you agree with this statement? Justify your answer.

When increasing the problem size, we can clearly see in the graphical representation of the runtimes, that fibonacci algorithm takes considerable runtime than the iterative algorithm. Iterative algorithm takes neglegebaly small time than fibbonacci algorithm for the same problem sizes when the problem size is larger. We can cleary decide that iteration is much more efficient and it doesn't exaust computers resources much as the recursion fibbonacci algorithm. Therefore if we consider runtime, when we have iteration as an option it is unwise to use time consuming recursion algorithm.

when listing performance according to runtime according to **Fig 3,**

1. Java iteration
2. Python Iteration
3. Java Recursion
4. Python Recursion

But if we need to optimize the size of our code, recursion is better therefore it is not completely useless, but most of the time we consider efficiency therefore for both languages, when the problem size is larger it is better to use iteration to get better runtime.