Course: ESO207A – Data Structures and Algorithms

Indian Institute of Technology Kanpur

Programming Assignment 1 : *Does the efficiency of algorithms really matter ?*

Most Important guidelines

- It is only through the assignments that one learns the most about the algorithms and data structures. You are advised to refrain from searching for a solution on the net or from a notebook. Remember - **Before cheating the instructor, you are cheating yourself**. The onus of learning from the course lies first on you.
- Keep in mind that there will be no one to assist you in the lab test held at the end of the course. So act wisely while working on the assignments of this course.
- Refrain from collaborating with the students of other groups or your friends. If any evidence is found that confirms copying, the penalty will be very harsh. Refer to the website at the link: https://cse.iitk.ac.in/pages/AntiCheatingPolicy.html regarding the departmental policy on cheating.
- This assignment **must** be done in groups of 2 only. It is your responsibility to find a partner.
- You need to upload 2 files. The first file will be a pdf file for the report that has the table, the graphs, and answers of all the questions of the assignment. The second file will have the C code that has the implementation of 3 algorithms as functions. Please follow the naming convention <RollNo1>_<RollNo2>_A1.pdf and <RollNo1>_<RollNo2>_A1.c while submitting your assignment on Moodle. For example - 210449_210733_A1.pdf and 210449_210733_A1.c. You may use any word processor (latex, word, ...) to prepare the report file. However, no scanned copy of a handwritten submission is allowed as report.
- Both the students in a group must submit the same files on Moodle.
- In case of any issue related to this assignment, send an email at ananyag@cse.iitk.ac.in (and not to the instructor)

The Objective of the Assignment

The followings are the objectives of this assignment.

1. To investigate whether the efficiency of an algorithm really matters in real world ?
2. To verify how well the RAM model of computation captures the time complexity of an algorithm.

Background of the Assignment

Fibonacci numbers have many applications in theoretical as well as applied computer science. They are used in pseudo-random number generators, Fibonacci heap data structure, analyzing efficiency of Euclids algorithm etc. Fibonacci numbers are also found in natural patterns like flower petals.

Let $F(n)$ denote nth Fibonacci number. Each of you would have written a code for Fibonacci numbers during the course ESC101. Consider a related computational problem whose input is an integer n, and output is $(F(n) \bmod 2021)$. In the lecture, three algorithms for solving this problem were discussed. The first algorithm was recursive and denoted by RFib. The second algorithm was iterative and denoted by IFib. And the third algorithm, CleverFib was quite complex – it involved repeated squaring to compute some power of a *cleverly* defined 2×2 matrix.

Tasks to be done

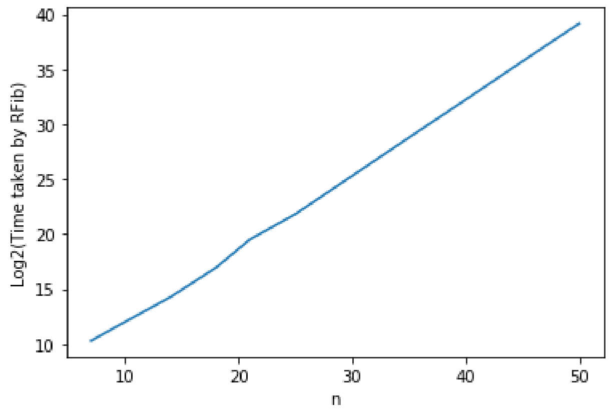
You have to implement the three algorithms mentioned above in C language, and find out, on your own, how efficiently RFib, IFib, and CleverFib solve the above problem in real time. The following tasks have to be completed in this assignment. ($marks = 4 + 4 + 8$)

1. The implementation of each algorithm is supposed to work for each possible value of n upto 10^{18}. However, it will turn out that some of these algorithms will start taking too much time as n increases. In order to observe this on your own, for each algorithm, you have to experimentally determine the largest possible value of n for which the corresponding implementation gives the output within the time limit (in seconds) from the set $\{0.001, 0.1, 1, 5, 60, 600\}$. You need to fill up the table given below with these values. If value of n exceeds 10^{18} for a specific time interval from the set, write $> 10^{18}$ in the corresponding entry in the table. ($marks = 10$)
Note: The hardware configuration used in the experiment may influence some of the values in the following table. However, it will have no impact on the nature of the graphs plotted below.

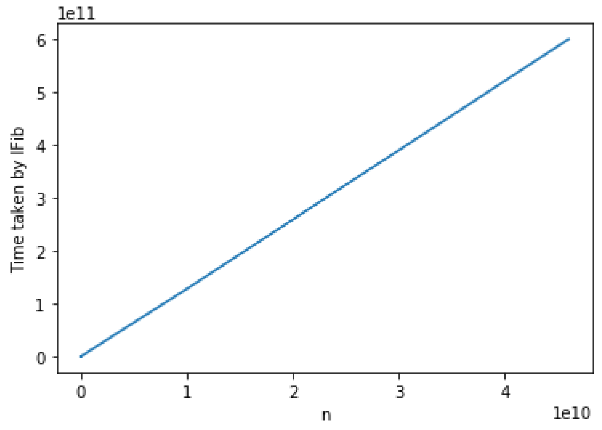
Time $\rightarrow$	0.001 sec	0.1 sec	1 sec	5 sec	60 sec	600 sec
RFib	22	31	36	40	45	49
IFib	$7 * 10^4$	$7.9 * 10^6$	$8 * 10^7$	$4 * 10^8$	$4.9 * 10^9$	$4.6 * 10^{10}$
CleverFib	$> 10^{18}$	$> 10^{18}$	$> 10^{18}$	$> 10^{18}$	$> 10^{18}$	$> 10^{18}$

2. Plot the following graphs. ($marks = 5 + 5 + 5$)

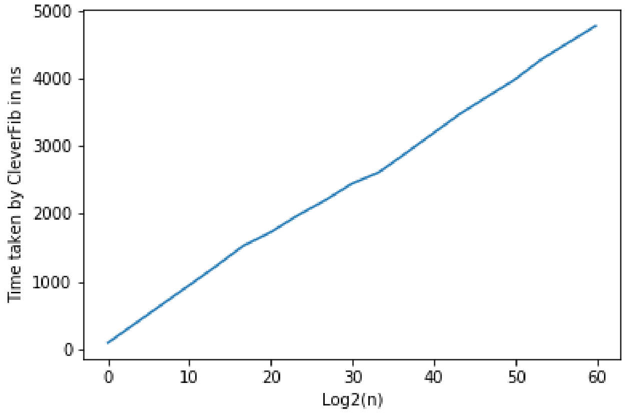
   (i) Log_2(Time taken by RFib in nanoseconds) as a function of n.



   (ii) Time taken by IFib in nanoseconds as a function of n.



(*iii*) Time taken by CleverFib in nanoseconds as a function of $\log_2(n)$.



(a) (Answer this question in at most 3 sentences)
Provide precise and concise justification for the shapes of the graphs you obtain. (10)
Each graph is nearly a straight line. Thus, we can conclude that for Rfib $\log_2$(Time taken) is proportional to n, for IFib time taken is proportional to n and for cleverfib, time taken is proportional to $\log_2(n)$. This matches perfectly with the time complexity of these algorithms in the RAM model of computation.

(b) (Answer this question in at most 3 sentences)
If a graph is a line, what is the value of its slope ? If each graph is a line, can you provide an explanation for the difference in their slopes ? (15)
The slope value for RFib is nearly 0.7, for IFib, the slope value is close to 12 and for CleverFib, the value is approximately 70. For IFib and CleverFib, the difference in slopes is due to the difference in number and kind of operations in the functions of both these algorithms.
Note: For RFib, the slope value is independent of the hardware configuration as discussed in a lecture. The slope values for IFib and CleverFib depends upon the hardware configuration. However, the slope value of CleverFib is bound to be greater than that of IFib.

(c) In each call of CleverFib, the number of instructions executed (excluding the recursive call invoked) are significantly more than RFib. Moreover, CleverFib involved multiplication operation whereas the other two algorithms involved only addition operation. We know that multiplying a pair of numbers takes **more time** than adding a pair of numbers on a computer.

i. (Answer this question in at most 3 sentences)
Did these facts influence the running time of CleverFib ? (You might like to refer to the graph of CleverFib and the graphs of RFib and IFib to answer this question). (10)
Yes, the running time of CleverFib is influenced by the higher number of operations performed and the use of multiplication operations. CleverFib shows a higher slope due to this reason. For IFib, the number of operations per iteration is far less.

ii. (Answer this question in at most 3 sentences)
Did these facts affect the relative speed of CleverFib compared to the other 2 algorithms ? If not, then state the reason. (10)
For small values of n (say, around 10-15), IFib is the fastest and RFib is the slowest. For these values, CleverFib is not faster than IFib due to large (but constant) number of operations and slower multiplication operations that CleverFib needs to execute during a recursive call. However, CleverFib needs to invoke fewer recursive calls ($O(\log_2 n)$) compared to the $O(n)$ iterations of IFib, so CleverFib becomes the fastest as n grows.

3. (Answer the following questions in yes or no only)
Based on the observations and inferences from 1 and 2 above, how accurate did you find the RAM

model of computation in measuring the running time of an algorithm ? How accurate did you find the RAM models of computation in comparing the efficiency of a pair of algorithms ? (5 + 5)
Yes.
The RAM model is accurate as it can predict when an algorithm would break due to its running time - An $O(c^n)$ time algorithm will break as n approaches log of the frequency of the CPU. An $O(n^2)$ time algorithm will break as n approaches square root of the frequency of the CPU. An $O(n)$ time algorithm will break as n approaches the frequency of the CPU. An $O(\log n)$ time algorithm will never break.
Yes.
The model can successfully and accurately compare the efficiency of multiple algorithms for a problem as was evident in this assignment.

4. (This question is optional. Feel free not to attempt it.)
Did this assignment achieve its objective stated on the 2nd page ? To answer this question, please choose one of the following options honestly, sincerely, and purely based on your own experience. Also provide a brief justification if possible.

   (a) This assignment was like any other assignment we do routinely. I did not find any thing special in it.
   (b) This assignment does not seem to achieve the objective stated on the 2nd page.
   (c) This assignment was an eye-opener for me. I found it immensely useful.

   You are welcome to share any experience related to this assignment if you wish.

   Important note: Answer to this question will have absolutely no impact on your marks of this or future assignments. However, your response will be seriously considered the next time this course will be offered by the instructor.

Useful pointers and suggestions

1. You may use any kind of machine (desktop/laptop,...) to run the C code of each of the 3 algorithms. Just make sure that the same machine is used for each of them.

2. For calculating the actual running time in executing a function, you may use clock() function in C. For this you need to include library time.h library. The following code shows how to use it to find the time taken in executing a part of the code:

```
#include <stdio.h>
#include <time.h>    // for clock_t, clock()


void func(){
    printf("Hello World");
}
int main()
{
    clock_t start_t, end_t;
    double total_t;
    start_t = clock();

    func();

    end_t = clock();

    total_t = (double)(end_t - start_t) / CLOCKS_PER_SEC;
    // CLOCKS_PER_SEC is a constant defined in time.h and its value is 10^6.
    printf("Total time taken: %f\n", total_t  );

    return 0;
}
```

   clock() returns the actual time in microseconds. It might be the case that some algorithms are so fast that for small values of n, you won't be able to find the precise time taken during its execution using the above method. However, you are strongly advised to use your creative skills to deduce the precise time using the same method. The following idea might give you the right direction: *how to measure thickness of a page of a book consisting of 1000 pages using an inch tape ?*

3. You can plot using gnuplot in Linux. To plot using gnuplot, you need create a data file first. The data file "plot.dat" should look like this:

```
#n      time
1       132
10      360
50      1000
```

   Use the command "plot plot.dat" to make the final plot.
   You can also use Matplotlib in python or you can plot using Microsoft Excel in Windows. Feel free to use any other tool as well.

4. For the 2nd part, you should pick *sufficiently* many values of n and from *suitably wide* range of n for each algorithm. Just apply your common sense here.

5. The most important part of the assignment is the 2nd part. Aanlyse the graphs patiently and answer each question very carefully. Why were you not asked to plot the runtime versus n ? Ponder over it.