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.
- Please follow the naming convention <RollNo1>_<RollNo2>_A1.pdf and <RollNo1>_<RollNo2>_A1.c while submitting your assignmment on Moodle.
- Both the students in a group must submit the same files on Moodle.
- In case of any doubts, feel free to drop an email at ananyag@cse.iitk.ac.in

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 $(F(n) \bmod 2022)$. 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 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.

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.

Time →	0.001 sec	0.1 sec	1 sec	5 sec	60 sec	600 sec
RFib						
IFib						
CleverFib						

2. Plot the following graphs.
   - (i) Log_2(Time taken by RFib) as a function of n.
   - (ii) Time taken by IFib as a function of n.
   - (iii) Time taken by CleverFib 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.

   (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 ?

   (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 operations whereas the other two algorithms involved addition operation only. 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 fact 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).

      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.

3. (Answer the following questions in yes or no only)
   Based on the facts you observed in 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 ?

4. (This question is optional. Feel free not to attempt it.)
   Did this assignment achieve its objective stated on the previous page ? You are welcome to share any experience related to this assignment if you wish. Answer to this question will have absolutely no impact on your marks of this or future assignments.

Useful suggestions and pointers

1. 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 ?*

2. 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.

3. 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.

4. 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.