By: Mohit Sharma (11434)

Ayush Mittal (11183)

CS-210: ASSIGNMENT #1

Problem 1: Nth Fibonacci number (mod m)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Observation*  *table* | **RECURSIVE**  **ALGORITHM** |  | **ITERATIVE**  **ALGORITHM** |  | **MATRIX**  **ALGORITHM** |
| TIME (sec) | N | N/(10^7) | N/(10^17) |
| 1 | 40 | 4 | 1 |
| 10 | 44 | 35 | - |
| 60 | 48 | 220 | - |
| 600 | 53 | 3000 | - |

Note: m taken to be 25, 1000,10^10 according to Fib(N).

**THEORITICAL TIME COMPLEXITY:**

* Recursive algorithm: O(2n)
* Iterative algorithm: O(n)
* Matrix algorithm: O(log2n)

**INFERENCES:**

* Graphical analysis shows:

1. Exponential order algorithms die out abruptly.
2. Graphs are in good agreement with theoretical time complexities.

* Time completely depended on N; m did not play any significant role.
* Efficiency of an algorithm comes into play at large values of input.
* Every single statement starts to matter when input size become large. So unnecessary additions should be avoided.
* Calculated values should be made full use of. Smart algorithms can avoid unnecessary recalculations.

Problem 2: Factorial of a large number n.

|  |  |
| --- | --- |
| **TIME (in sec)** | **N** |
| .001 | 1000 |
| 1 | 18000 |
| 60 | 1,20,000 |
| 600 | 5,30,000 |

**TIME COMPLEXITIES:**

Theoritical time complexity: O(n2.log2n)

Time complexity from graph:

**INFERENCES:**

* Performing digit by digit multiplication in usual way had the same time complexity but was much slower.
* Increasing number of digit in which we multiply in one go increases the efficiency, however there is an upper cap on this due to capacity of long long int. Hence best combination of resources are to be identified in designing an algorithm.