**Problem 3.1** a) Code in ADS\_3a.c

b) Table: Time Computations for increasing n

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **n** | **naïve** | **n** | **bottom** | **n** | **Closed form** | **n** | **Matrix** |
| 0 | 0.000001 | 0 | 0.000001 | 0 | 0.000001 | 0 | 0.000001 |
| 1 | 0.000001 | 1 | 0.000001 | 1 | 0.000001 | 1 | 0.000002 |
| 2 | 0.000002 | 2 | 0.000001 | 2 | 0.000002 | 2 | 0.000002 |
| 4 | 0.000002 | 4 | 0.000001 | 4 | 0.000003 | 4 | 0.000001 |
| 6 | 0.000003 | 6 | 0.000001 | 6 | 0.000005 | 6 | 0.000002 |
| 8 | 0.000003 | 8 | 0.000001 | 8 | 0.000004 | 8 | 0.000003 |
| 12 | 0.000004 | 12 | 0.000001 | 12 | 0.000004 | 12 | 0.000002 |
| 18 | 0.000045 | 18 | 0.000001 | 18 | 0.000003 | 18 | 0.000003 |
|  |  | 27 | 0.000001 | 27 | 0.000004 | 27 | 0.000004 |
|  |  | 35 | 0.000001 | 35 | 0.000005 | 35 | 0.000005 |
|  |  | 45 | 0.000001 | 45 | 0.000004 | 45 | 0.000004 |
|  |  | 60 | 0.000001 | 60 | 0.000005 | 60 | 0.000005 |
|  |  | 80 | 0.000001 | 80 | 0.000004 | 80 | 0.000004 |
|  |  | 110 | 0.000001 | 110 | 0.000006 | 110 | 0.000007 |
|  |  | 150 | 0.000002 | 150 | 0.000008 | 150 | 0.000006 |
|  |  | 250 | 0.000002 | 250 | 0.000007 | 250 | 0.000005 |
|  |  | 400 | 0.000003 | 400 | 0.000005 | 400 | 0.000005 |
|  |  | 600 | 0.000004 | 600 | 0.000008 | 600 | 0.000004 |
|  |  | 900 | 0.000006 | 900 | 0.000007 | 900 | 0.000006 |
|  |  | 1500 | 0.000016 | 1500 | 0.000006 | 1500 | 0.000006 |
|  |  | 2500 | 0.000023 | 2500 | 0.000007 | 2500 | 0.000008 |
|  |  | 4000 | 0.000047 | 4000 | 0.000007 | 4000 | 0.000009 |
|  |  | 7000 | 0.00004 | 7000 | 0.000007 | 7000 | 0.000007 |

b)

The time for each step has been taken 100 times and average is calculated. This is also the reason why randomness is reduce in the graph and we don’t have random peaks(also, because I used a curve instead of joining with lines).

c)The closed formula method doesn’t return the same Fibonnacci value for larger n because there is rounding involved, and the presence of floating point precision gives us different values(not as precise as expected). However, the other 3 methods return same Fibonnacci number for same n.

d) The graph clearly demonstrates the time complexity of naïve recursive being O(n^2), bottom up being O(n), and closed formula and matrix being O(log n). Time grows very significantly for recursion, and very slow for matrix and closed formula (there’s not much difference between them for the same time complexity), and time growth is linear for bottom up approach. 