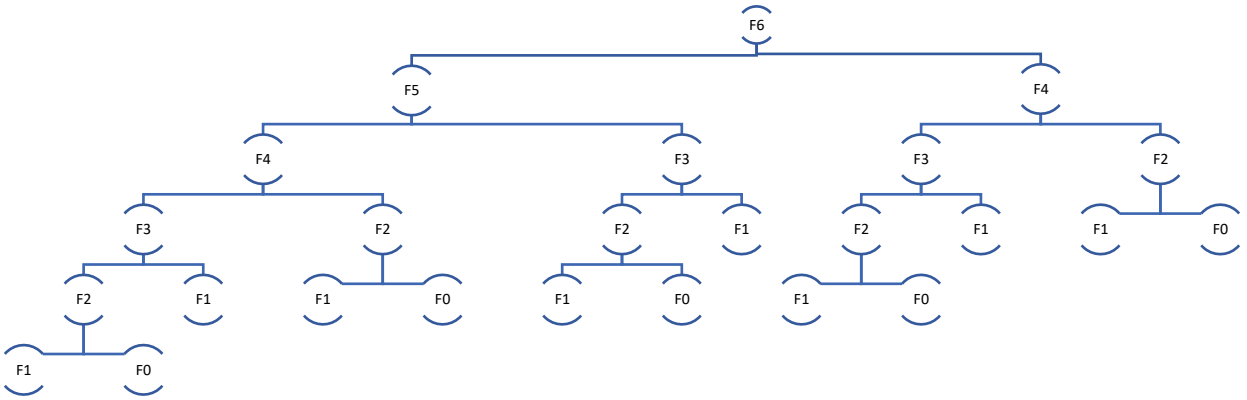
1.

a. Yes it performs redundant calculations. For example, for calculating F6 it has to recalculate F5,..F2 all again as it doesn't have it saved. The redundancy can be seen in part (b).

b.



c. F2: 5 times
F3: 3 times
F4: 2 times
F5: 1 time

2. By performing Algorithm Rate Analysis of algorithm 2:
Algorithm "algorithm2", with f(n)= 6n+6 is O(n). this is because the algorithm was based on one loop that will has linear growth. Tables showing the growth analysis of algorithm2 are attached in Appendix A pdf file.

3. By performing algorithm analysis of algorithm 3: Algorithm "algorithm3", with $f(n)=4\log_2(n)+18$ is $O(\log_2 n)$. the algorithm was based on a call for a recursive function that has a logarithm growth, so its normal for f(n) to be O(logn). Tables showing the growth rate analysis of algorithm3 are attached in Appendix A pdf file

4. according to experimental studies results, as shown in the graphs plotted below, and by observing graph (algorithm2 vs algorithm3), we can see that algorithm2 outputs better results than algorithm3 for the range of $n<27$ where the two graphs intersect. Algorithm1 gives acceptable results for $n<20$, and it starts lagging for $n>20$ but because of its exponential growth it will be absolute to get a

Fibonacci number for n>45. Overall analysis, algorithm2 is better to be used for n<27, while algorithm3 is to be used for all n>27. (n represents the n^{th} Fibonacci number)

Graphs: