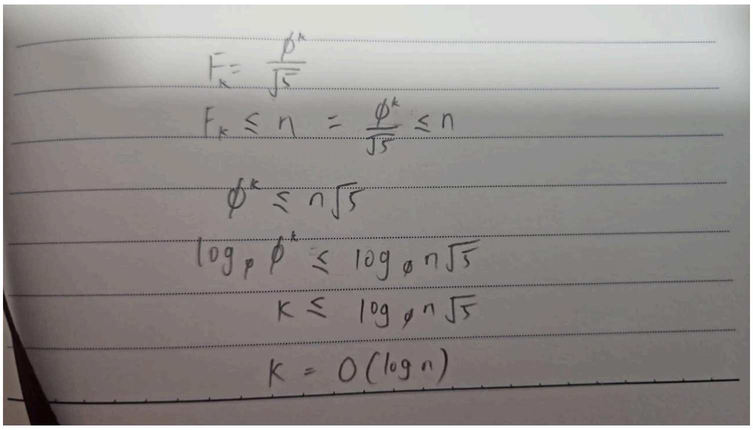
Question 2 complexity summary and justification

The goal of the question is to read one or more +ve integers from an input file and encode each using fibonacci encoding, then write the corresponding binary codeword to an output file. The encoding relies on generating fibonacci numbers up to the given integer and constructing the codeword using those fibonacci values. 1st we have to of course generate the fibonacci numbers up till n. In generate_fibonacci_up_to(n) we generate the fibonacci numbers up to n. Each iteration of the loop appends 1 fibonacci number until the sequence slightly exceeds n. Since Fibonacci numbers grow exponentially F = phi/sqrt(5) where phi is the golden ratio of 1.618, the number of fibonacci terms <= is bounded by O(log n).



$$F_k = \frac{\phi^k}{\sqrt{5}}$$

$$F_k \leq n = \frac{\phi^k}{\sqrt{5}} \leq n$$

$$\phi^k \leq n\sqrt{5}$$

$$\log_\phi \phi^k \leq \log_\phi n\sqrt{5}$$

$$k \leq \log_\phi n\sqrt{5}$$

$$k = O(\log n)$$

The loop executes once per fibonacci number so the time complexity is O(log n) since the operations are O(1). The space complexity is O(log n) since we have to store O(log n) number of fibonacci numbers. The next part of the code is the actual encoding where we use the generate_fibonacci_up_to(n). We then traverse the list of fibonacci numbers in reverse starting from the biggest fibonacci number to determine which fibonacci numbers sum to n so in the worst case we traverse the entire list O(log n). Lastly, we build the binary string of length O(log n) so the space and time complexity is O(log n). Finally in main, if we have m number of integers, each integer will take O(log n) time complexity as well as O(log n) space complexity to encode so we get total time complexity O(m log n_max) where n_max is the biggest integer since we are getting the worst case complexities. We also have O(log n_max) space complexity since each integer needs O(log n) space.