По определению логарифм – число, на которое надо возвести 2, чтобы получить какое-то заданное число. 2log(n) = n. Если путём возведения в какую-то целую степень мы никак не можем получить это число, то будем брать наименьшую степень, при возведении числа 2 которой получится наименьшее число, которое больше, чем заданное.  
Например, log(1024) = 10, но log(1025) = 11, log(2047) = 11.

1. Чтобы подобрать верхнее значение на n, надо решить уравнение:  
   ⌈3\*n\*log(n) + 17\*n⌉ = 1 000 000 000  
   Существуют различные способы решения, в данном случае буду использовать метод перебора искомого значения (n). Перебор будет осуществляться, как бинарный поиск. Будем искать наибольшее значение, при котором компьютер выполнит программу меньше, чем за 1 секунду.  
   Левое значение выберем за ноль, правое – за 1 000 000 000.  
   Среднее значение получаем из левого и правого как среднее арифметическое из них (используется невещественное деление). Подставляя среднее значение в формулу (3\*middle\*log(middle) + 17\*middle), надо сравнить полученное значение и 109. Если оно меньше или равно 109, то меняем левое значение на среднее, иначе правое заменяем на среднее. Продолжаем это делать, пока разница между правым и левым не стала 1. В итоге получаем результат, равный 11235955.  
   Если подставить это число в формулу, то получим 999999995. Это не превышает 109, то есть, если сделать верхним ограничением 11235955, то это уложится в одну секунду.
2. Для того, чтобы подобрать новое ограничение на n, можно, опять-таки, использовать метод бинарного поиска. Для этого всего лишь надо подобрать левое и правое значения, которые бы дали точно меньше и точно больше результат соответственно, чем f при подстановке этих значений в формулу. (можно взять правое за f, левое за ноль), потом сравнивать f с 3\*middle\*log(middle) + 17\*middle. Сравнений придётся сделать немного, так как функция логарифма медленно растёт. Если в результате получится 0, то для такого f нельзя решить задачу, потратив менее 1 секунды.