Part 1: Two Fibonaccis

$fib(n)$:
  if $n = 0$ or $n = 1$ **then**
    return n
  else
    return $fib(n-1) + fib(n-2)$
  end if

We can state a recurrence for this algorithm:

$$\begin{aligned} T(n) &= T(n-1) + T(n-2) + O(1) \\ &\geq fib(n-1) + fib(n-2) \\ &= fib(n) \end{aligned}$$

$fib2(n)$:
  if $n = 0$ **then**
    return 0
  end if
  create array $f[0..n]$
  $f[0] \leftarrow 0, f[1] \leftarrow 1$
  for $i \leftarrow 2$ to n **do**
    $f[i] \leftarrow f[i-1] + f[i-2]$
  end for
  return $f[n]$

Addition of two numbers in the preceding algorithm takes constant time until the values exceed the maximum value that can be stored in a word. After which, we need to consider how values of arbitrary length are added.

Part 2 - Integer Multiplication

Let us consider an integer X which is composed of X_L which are the leftmost bits of X, and X_R which are the rightmost bits of X.

$$X = X_L | X_R$$

We can multiply integers X, Y as follows:

$$\begin{aligned} XY &= (2^{n/2} X_L + X_R)(2^{n/2} Y_L + Y_R) \\ &= 2^n X_L Y_L + 2^{n/2} X_L Y_R + 2^{n/2} X_R Y_L + X_R Y_R \end{aligned}$$

Which gives the recurrence

$$\begin{aligned} T(n) &= 4T(n/2) + O(n) \\ &\leq 4T(n/2) + cn \\ &\leq 4(4T(n/4) + cn/2) + cn \\ &\leq 4(4(4T(n/8) + cn/4) + cn/2) + cn \\ &\leq 64T(n/8) + cn(1 + 2 + 4) \end{aligned}$$

Which we can see by unfolding is:

$$\begin{aligned} T(n) &\leq cn(1 + 2 + 4 + ...) \\ &= cn \sum_{n=0}^{\infty} 2^n \end{aligned}$$