Aula 1 - Prática

```
def is_prime_recursive(n, div = 2, div_count = 0): # melhor caso (n < 2) => O(1)
  if n < 2 or div_count > 0: return False              # caso médio (n == 2) => O(2)
  if div == n: return n                                # pior caso (n > 2)   => O(n - 2) ≃ O(n)
  if n % div == 0: div_count+=1
  return is_prime_recursive(n, div+1, div_count)

def is_prime_iterative(n): # melhor caso (n < 2) => O(1)
  if n < 2: return False      # caso médio (n == 2) => O(3)
  div = 2                     # pior caso (n > 2)   => O(n - 2) ≃ O(n)
  while div < n:
    if n % div == 0: return False
    div+=1
  return n
```

Analisando o pior caso, ambas as funções tem complexidade de tempo O(n), embora a função recursiva tenha um tempo de execução aprox. 3x maior que a versão iterativa. Isso ocorre pois é necessário fazer o endereçamento das sub-rotinas de cada chamada recursiva.