1. Insertion Sort:

for i = 1 to n
  key = A[i]
  // Insert A[i] into the sorted subarray A[1:i - 1].
  j = i - 1
  while j >= 0 and A[j] > key
  A[j + 1] = A[j]
  j = j - 1
  A[j + 1] = key
	N + 1
	N
	
	N
	N(N)
	N(N - 1)
	N(N - 1)
	N
	
	= 2N(N-1) + 2N^2 + 4N + 1
	**j = N - 1
	The time complexity is quadratic.

2. Matrix Multiplication

MATRIX_MULTIPLY(A, B):
  if columns(A) ≠ rows(B):
  raise ValueError("DNE")
 	
  rows_A ← number of rows in A
  cols_A ← number of columns in A
  cols_B ← number of columns in B
  result ← matrix of size rows_A x cols_B filled with zeros
 	
  for i from 1 to rows_A do:
  for j from 1 to cols_B do:
  sum ← 0
  for k from 1 to cols_A do:
  sum ← sum + A[i][k] * B[k][j]
  result[i][j] ← sum return result
	
	1
	1
	
	1
	1
	1
	1
	
	i
	i (j)
	i (j - 1)
	i (k)
	i (k - 1)
	i
	
	= i(k) + i(j) + i(k -1) + i(j-1) + 2i + 7
	= i(k + j) + i(k -1) + i(j-1) + 2i + 7
	
	i = # of rows in A
	j = # of columns in B
	k = # of columns in A
	
	The time complexity is quadratic.