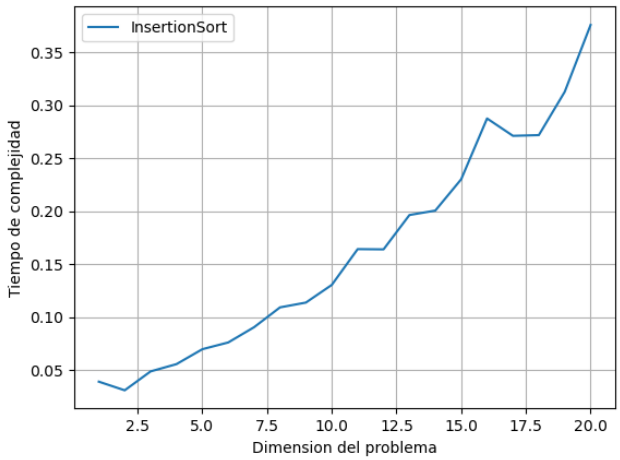
INSERTION SORT

1. El algoritmo insertion sort no sería apropiado para ordenar grandes volúmenes de datos debido a que al tener una complejidad O(n)=n^2, esto indica que la cantidad de operaciones que se realizan durante la ejecución crecerán de forma cuadrática, lo que hace que no sea un algoritmo tan apropiado en tales casos.

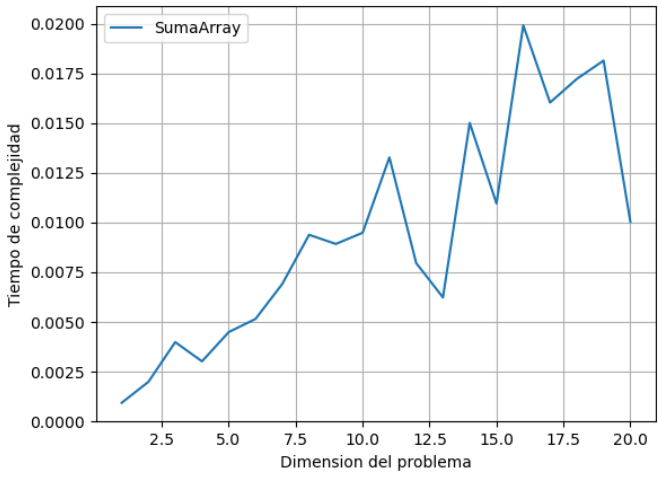


La complejidad que obtuvimos fue de:

```
T(n)=C1*n+n*C2+C3*n(n+1)/2+C4*n(n+1)/2+C5*n(n+1)/2+C6*n(n+1)/2+C7*n(n+1)/2
T(n)=(n(n+1)/2*)(C3+C4+C5+C6)+C'-->Regla de las sumas y producto
T(n)=(n^2+n)*1/2*C'---->Se quita la suma interna
T(n)=n^2+n---->Se quitan los productos y la suma interna
T(n)=O(n^2)
"""
```

ARRAYSUM

1. La diferencia en el tiempo de ejecución entre la implementación con ciclos y la implementación con recursión, no es muy significativa. Esto se pudo notar ya que, al calcular las complejidades de ambas, se determinó que las dos eran de O(n) = n, lo que quiere decir que aproximadamente realizar la misma cantidad de operaciones.



La complejidad que obtuvimos para:

- Suma de los elementos de un arreglo con ciclos.

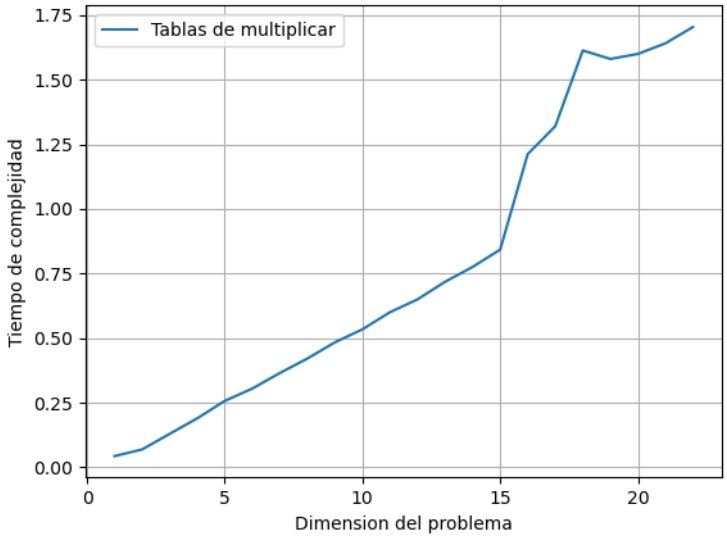
```
#T(n) = c1 + c2*n + c3*n + c4
#T(n) = n*(c2+c3)
#T(n) = n*c
#T(n) = O(n)
```

- Suma de elementos de un arreglo con recursión.

```
#T(n) = c3 + T(n-1)
#T(n) = T(n-1)
# O(n) = n
```

TABLAS DE MULTIPLICAR

1. De forma experimental obtuvimos que la complejidad para el peor caso en el algoritmo que calcula las tablas de multiplicar de 1-n, es de O(n)= n^2, por lo que podemos decir que la teoría de notación asintótica si corresponde con lo encontrado de forma experimental.



La complejidad que obtuvimos fue:

```
#T(n)=C1*n+C2*n^2+C3*n^2+C4*n^2
#T(n)=n^2*(C1+C3+C4)+C'-->Regla de las sumas y producto
#T(n)=n^2*C'---->Se quitan los productos y se simplifica la suma
#T(n)=O(n^2)
```