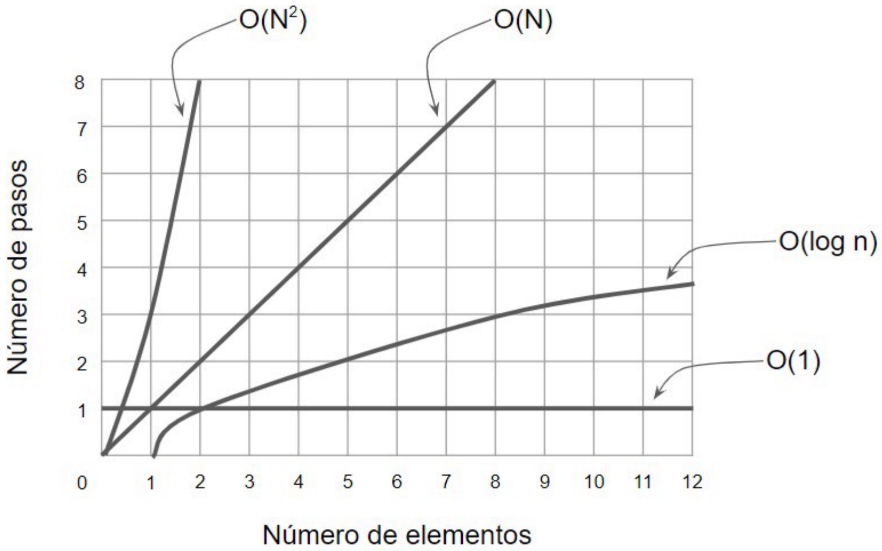
Ejercicio No. 1

La complejidad teórica de los cosos.



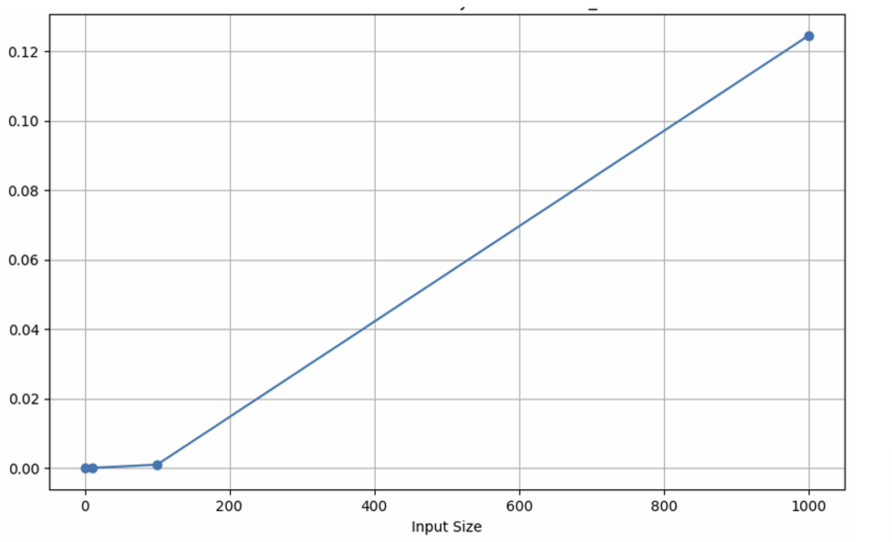
Ejercicio No. 1 (25%) – Para el siguiente programa:

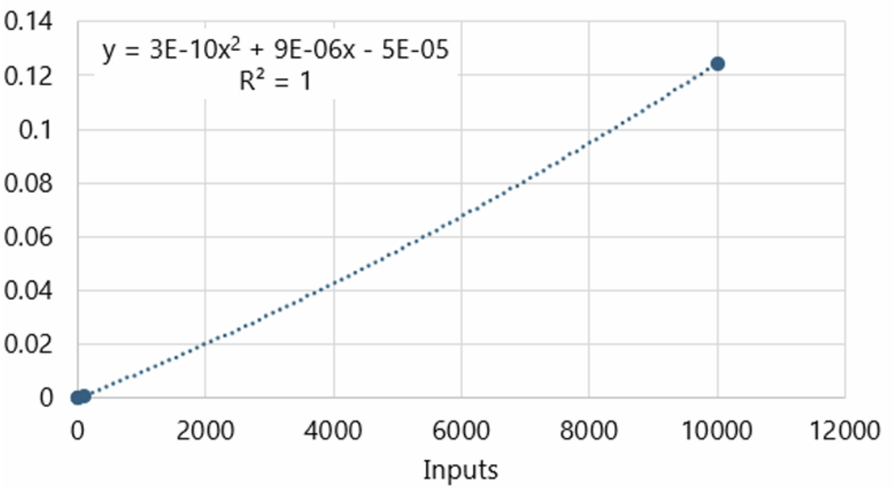
```
void function (int n) {
      int i, j, k, counter = 0;
      for (i = n/2; i <= n; i++) {
            for (j = 1; j+n/2 <= n; j++) {
                  for (k = 1; k <= n; k = k*2) {
                        counter++;
                  }
            }
      }
}
```

for (i = n/2; i <= n; i++)
n/2 veces → **O(n)**
for (j = 1; j + n/2 <= n; j++)
Se ejecuta **O(n/2) = O(n)** veces.
for (k = 1; k < n; k = k * 2)
Número de iteraciones ≈ $\log_2(n)$ → **O(log n)**

O(n)∗O(n)∗O(logn)=O(n2log⁡n)O(n) * O(n) * O(log n) = O(n^2 \log n)O(n)∗O(n)∗O(logn)=O(n2logn)

Complejidad temporal: $O(n^2 \log n)$





Ejercicio No. 2

Ejercicio No. 2 (25%) – Para el siguiente programa:

```
void function (int n) {
        if (n <= 1) return;
        int i, j;
        for (i = 1; i <= n; i++) {
                for (j = 1; j <= n; j++) {
                        printf (“Sequence\n”);
                        break;
                }
        }
}
```

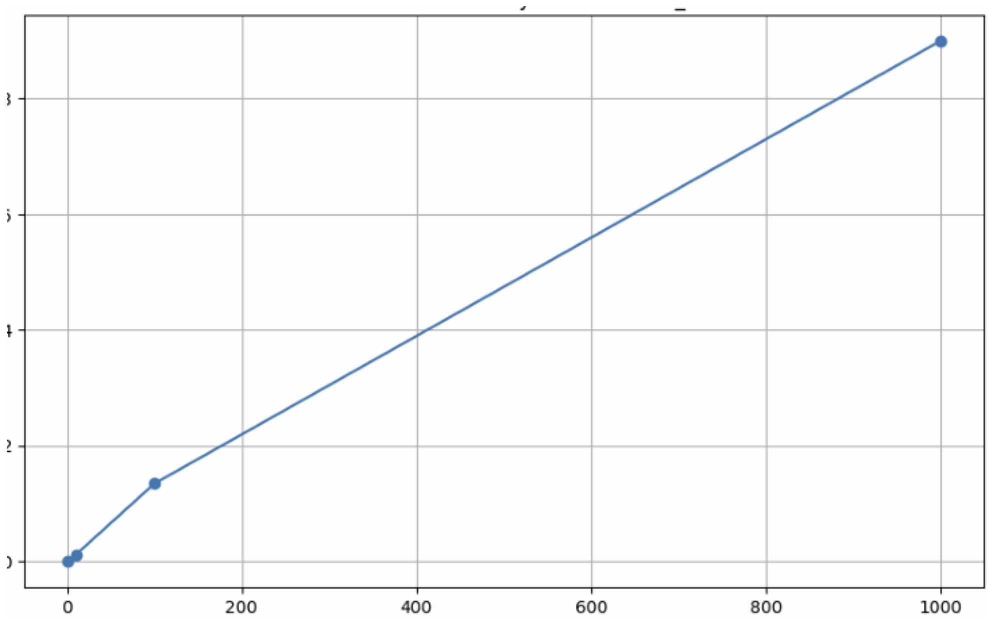
if (n <= 1) return;
O(1).

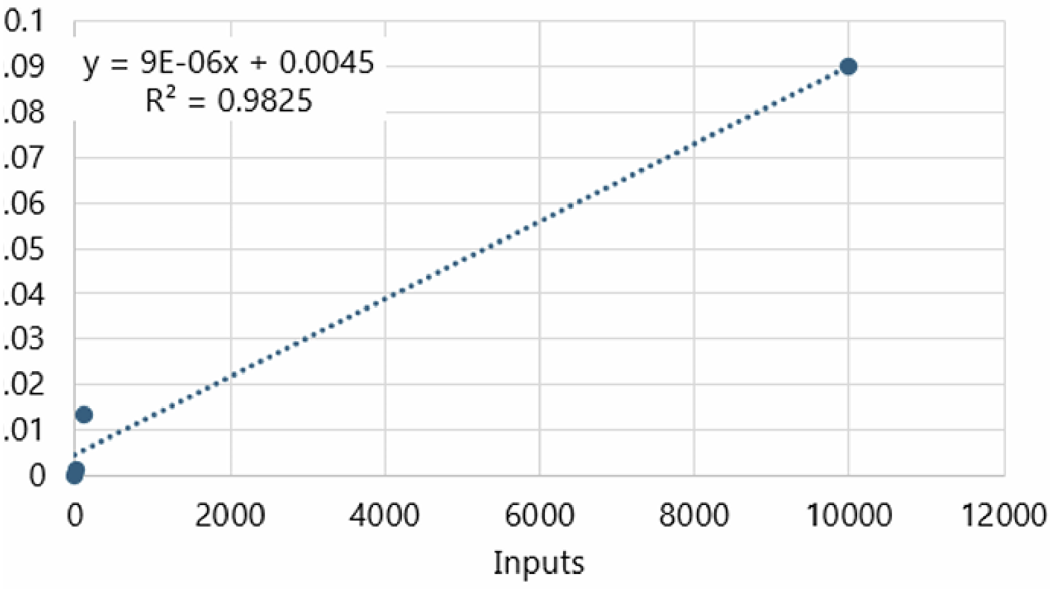
for (i = 1; i <= n; i++)
n veces → O(n).

for (j = 1; j <= n; j++) { ... break; }
i → O(1).

*O(1)+O(n)∗O(1)=O(n)O(1) + O(n) * O(1) = O(n)*O(1)+O(n)∗O(1)=O(n)

Complejidad temporal: O(n)





Ejercicio No. 3

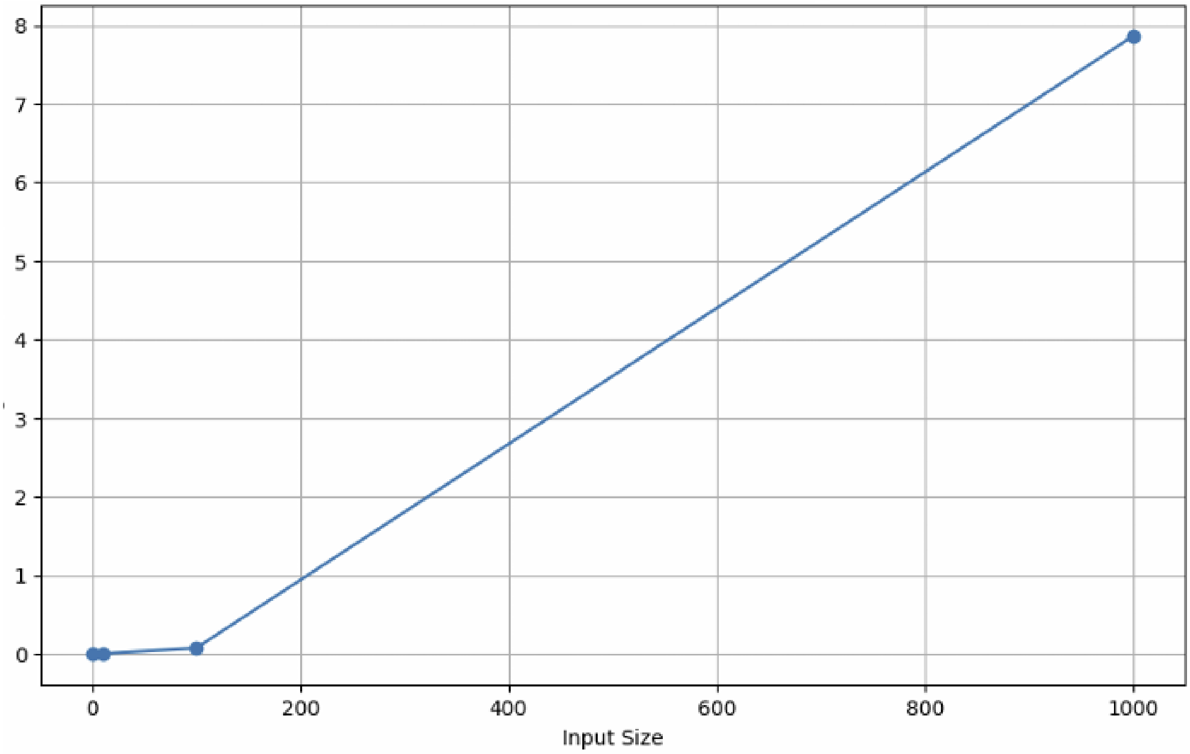
Ejercicio No. 3 (25%) – Para el siguiente programa:

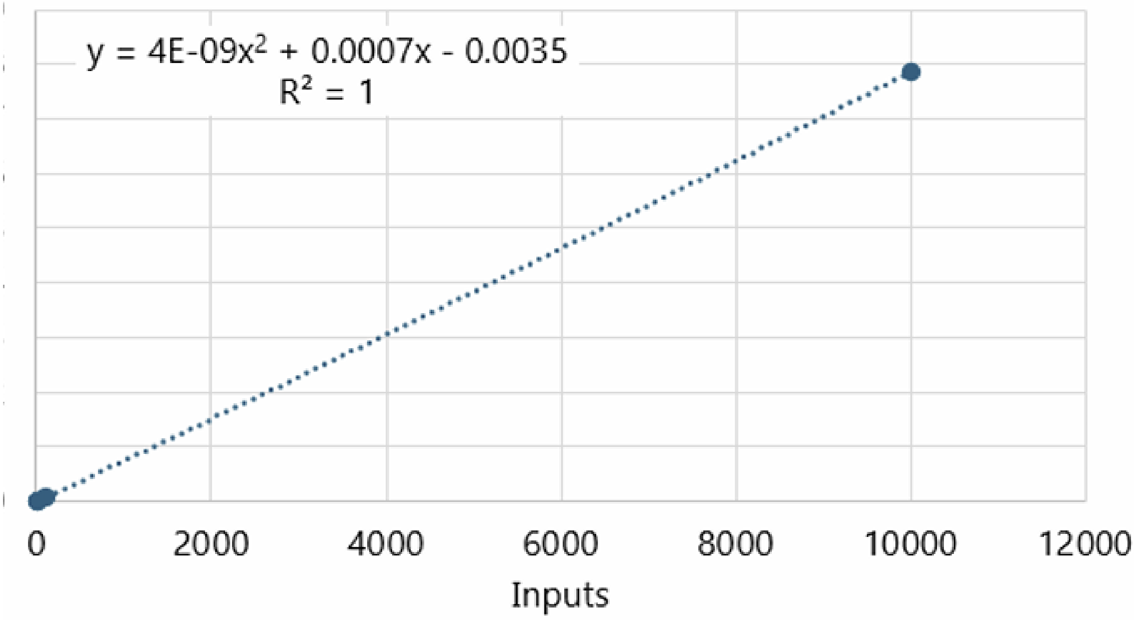
```
void function (int n) {
        int i, j;
        for (i=1; i<=n/3; i++) {
                for (j=1; j<=n; j+=4) {
                        printf("Sequence\n");
                }
        }
}
```

for (i = 1; i <= n/3; i++)
n/3 veces
for (j = 1; j <= n; j += 4)
n/4 veces → **O(n)**.

*O(n)∗O(n)=O(n2)O(n) * O(n) = O(n^2)*O(n)∗O(n)=O(n2)

Complejidad temporal: $O(n^2)$





Ejercicio No. 4

Agregue un código en go para los casos.

Mejor caso: O(1)

- El elemento x se encuentra **en la primera posición**.
- Solo se realiza **una comparación**.
- **$T(n) = 1 \rightarrow O(1)$**

Caso promedio: O(n)

- El elemento x puede estar en **cualquier posición** del arreglo con igual probabilidad.
- En promedio, se harán n/2 comparaciones.

*T(n)=(1+2+3+⋯+n)/n=(n+1)/2T(n) = (1 + 2 + 3 + \dots + n) / n = (n+1)/2*T(n)=(1+2+3+⋯+n)/n=(n+1)/2

Peor caso: O(n)

- El elemento x está **al final del arreglo** o **no está presente**.
- Se recorren **todos los elementos**: n comparaciones.

Ejercicio No. 5

a) **Verdadero**

- Sea **$f(n) = \Theta(g(n))$**.
  Existen **n_1** y constantes positivas **c_1**, **c_2** tales que, para todo **$n \geq n_1$**,
  $c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$
- Sea **$g(n) = \Theta(h(n))$**.
  Existen **n_2** y constantes positivas **d_1**, **d_2** tales que, para todo **$n \geq n_2$**,
  $d_1 \cdot h(n) \leq g(n) \leq d_2 \cdot h(n)$
- Tomando **$n_0 = \text{máx}(n_1, n_2)$**, para todo **$n \geq n_0$** se cumple:
  $c_1 \cdot d_1 \cdot h(n) \leq f(n) \leq c_2 \cdot d_2 \cdot h(n)$
- Por lo tanto, **$f(n) = \Theta(h(n))$**, y también **$h(n) = \Theta(f(n))$** por simetría.

b) Verdadero

b) Verdadero.

$f(n) = O(g(n))$ significa que existen n_1 y $a > 0$ tales que, para todo $n \geq n_1$,
$f(n) \leq a \cdot g(n)$.

$g(n) = O(h(n))$ significa que existen n_2 y $b > 0$ tales que, para todo $n \geq n_2$,
$g(n) \leq b \cdot h(n)$.

Tomando $n_0 = \text{máx}(n_1, n_2)$, para todo $n \geq n_0$ se cumple:
$f(n) \leq a \cdot g(n) \leq a \cdot (b \cdot h(n)) = (a \cdot b) \cdot h(n)$.

Por tanto, $f(n) = O(h(n))$.
Y por definición, $f(n) = O(h(n)) \Leftrightarrow h(n) = \Omega(f(n))$.
Por lo tanto, la afirmación es verdadera.

c) Falso.
El programa tiene dos bucles anidados con $\Theta(n^2)$ iteraciones, pero dentro de cada una se ejecuta `S.add(atupla[i:j])`, que crea una subtupla de tamaño $(j - i)$, lo cual cuesta $\Theta(j - i)$.
Al sumar todos los costos, el tiempo total es:

$T(n) = \Sigma_{i=0}^{n-1} \Sigma_{j=i+1}^{n-1} \Theta(j - i) = \Theta(n^3)$

Por tanto, $f(n) = \Theta(n^3)$, no $\Theta(n^2)$.