PRÁCTICA I

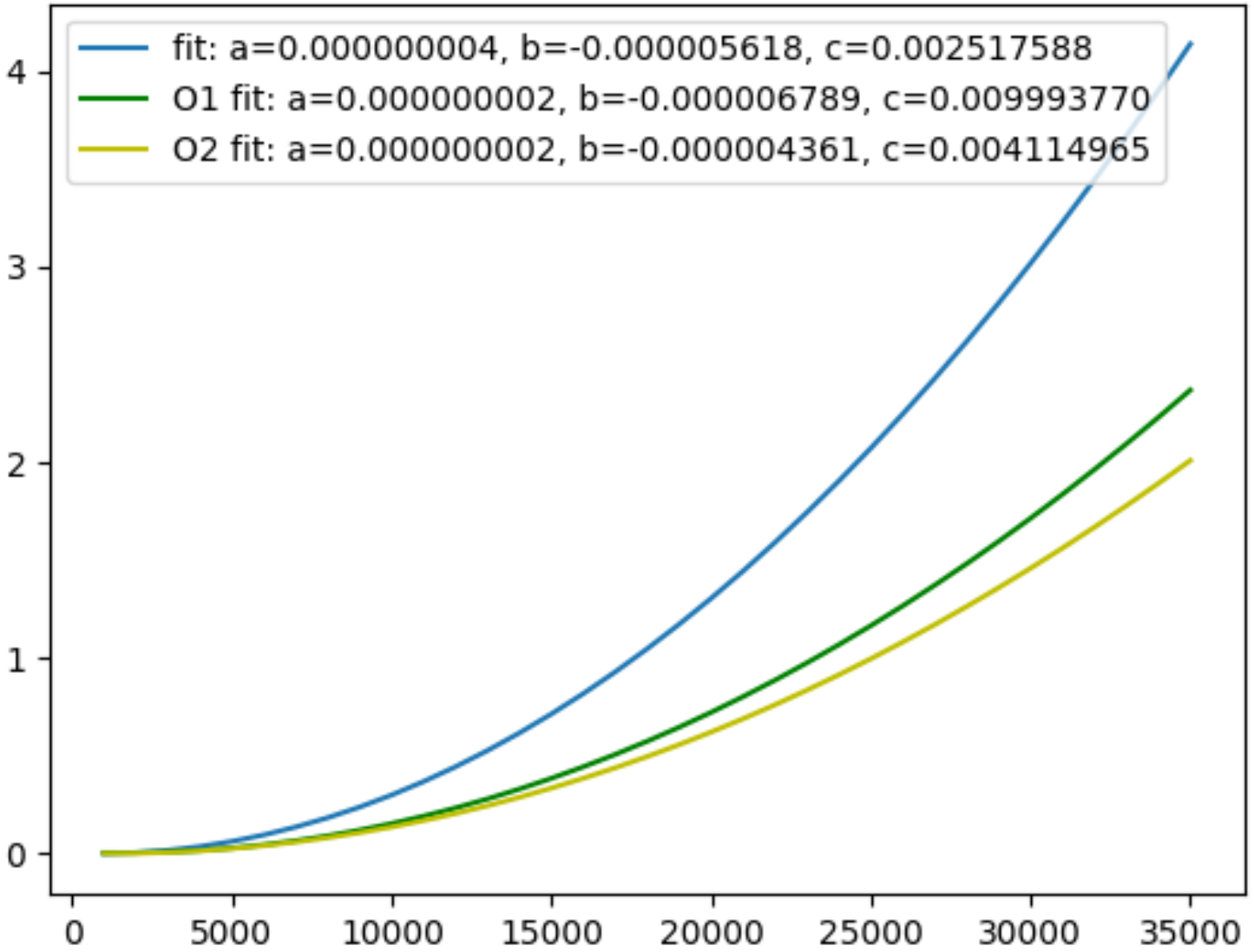
EFICIENCIA DE ALGORITMOS

By: Elena Merelo y Antonio Gámiz

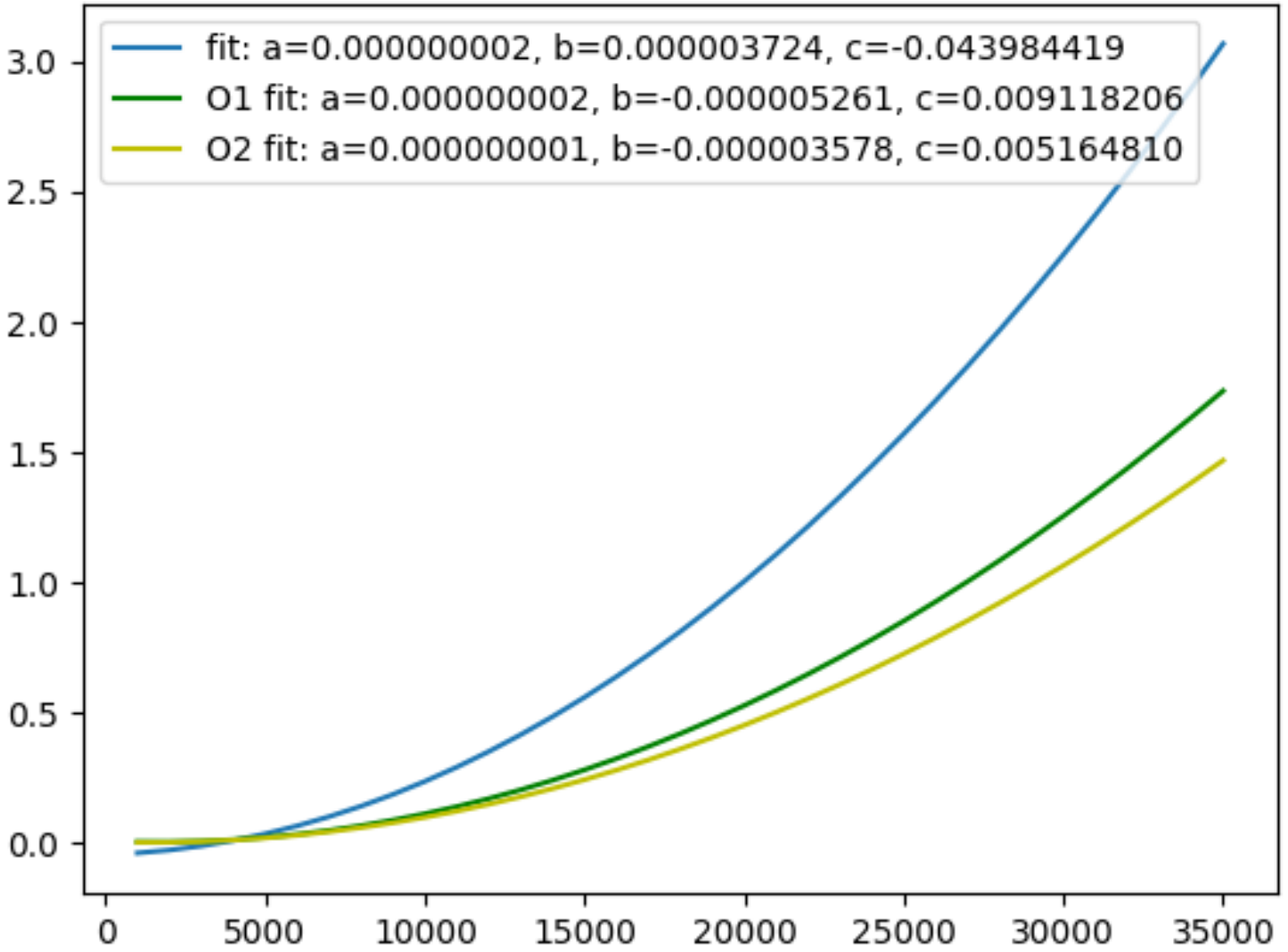
Algoritmos analizados

Algoritmo	Orden de Eficiencia
Burbuja	$O(n^2)$
Inserción	$O(n^2)$
Selección	$O(n^2)$
Mergesort	$O(n \log(n))$
Quicksort	$O(n \log(n))$
Heapsort	$O(n \log(n))$
Floyd	$O(n^3)$
Hanoi	$O(2^n)$

Algoritmo de burbuja - Elena



Algoritmo de burbuja - Antonio

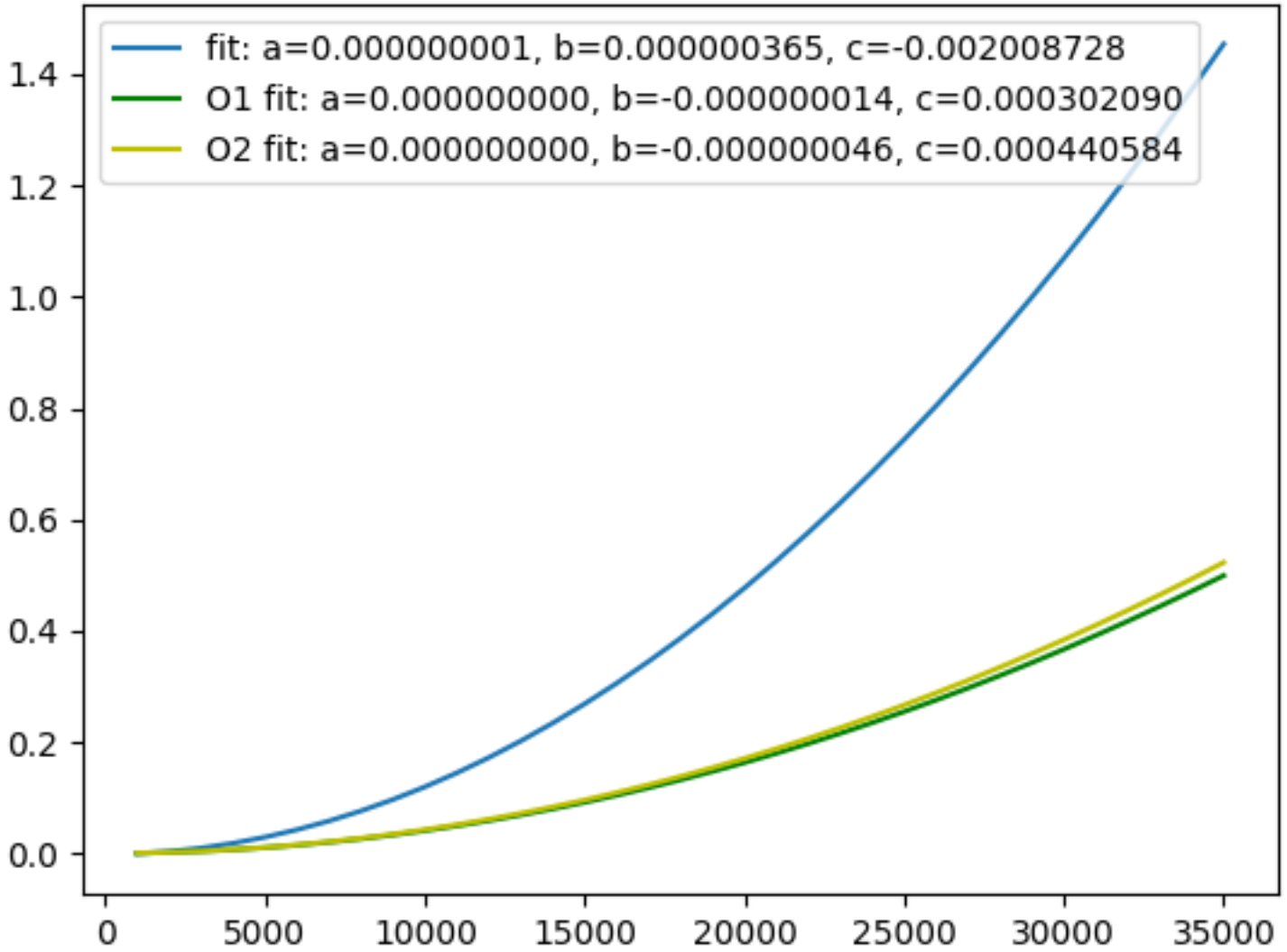


Algoritmo de inserción – Análisis teórico

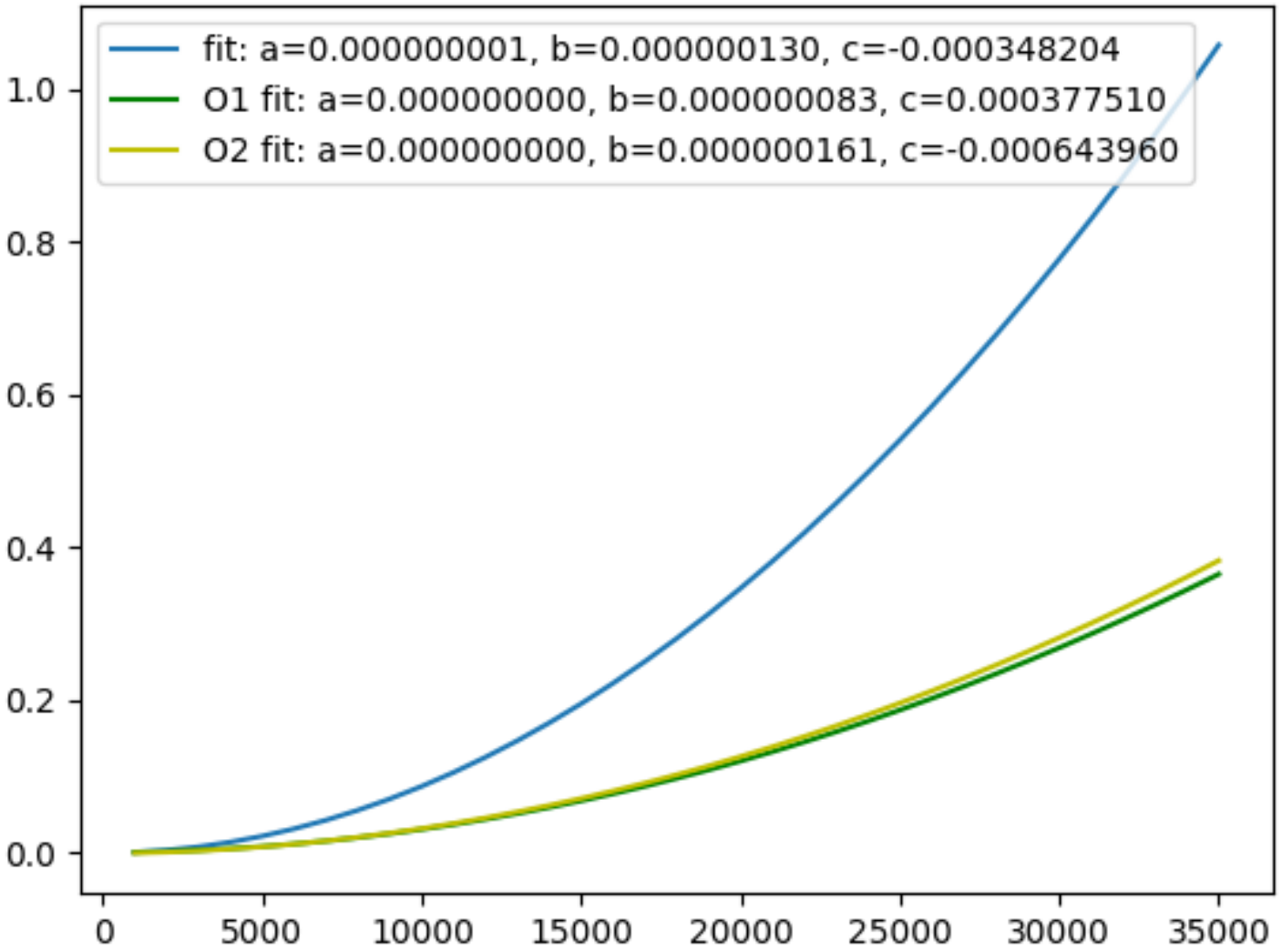
```
1  inline static void insercion(int T[], int num_elem)
2  {
3    insercion_lims(T, 0, num_elem);
4  }
5
6  static void insercion_lims(int T[], int inicial, int final)
7  {
8    int i, j;
9    int aux;
10   for (i = inicial + 1; i < final; i++) {
11     j = i;
12     while ((T[j] < T[j-1]) && (j > 0)) {
13       aux = T[j];
14       T[j] = T[j-1];
15       T[j-1] = aux;
16       j--;
17     };
18   };
19 }
```

$$T(n) = \sum_{i=1}^{n-1} \sum_{j=1}^{i} a = a \sum_{i=0}^{n-1} \sum_{j=1}^{i} 1 = a \sum_{i=0}^{n-1} i = a \frac{n(n-1)}{2}$$

Algoritmo de inserción - Elena



Algoritmo de inserción - Antonio

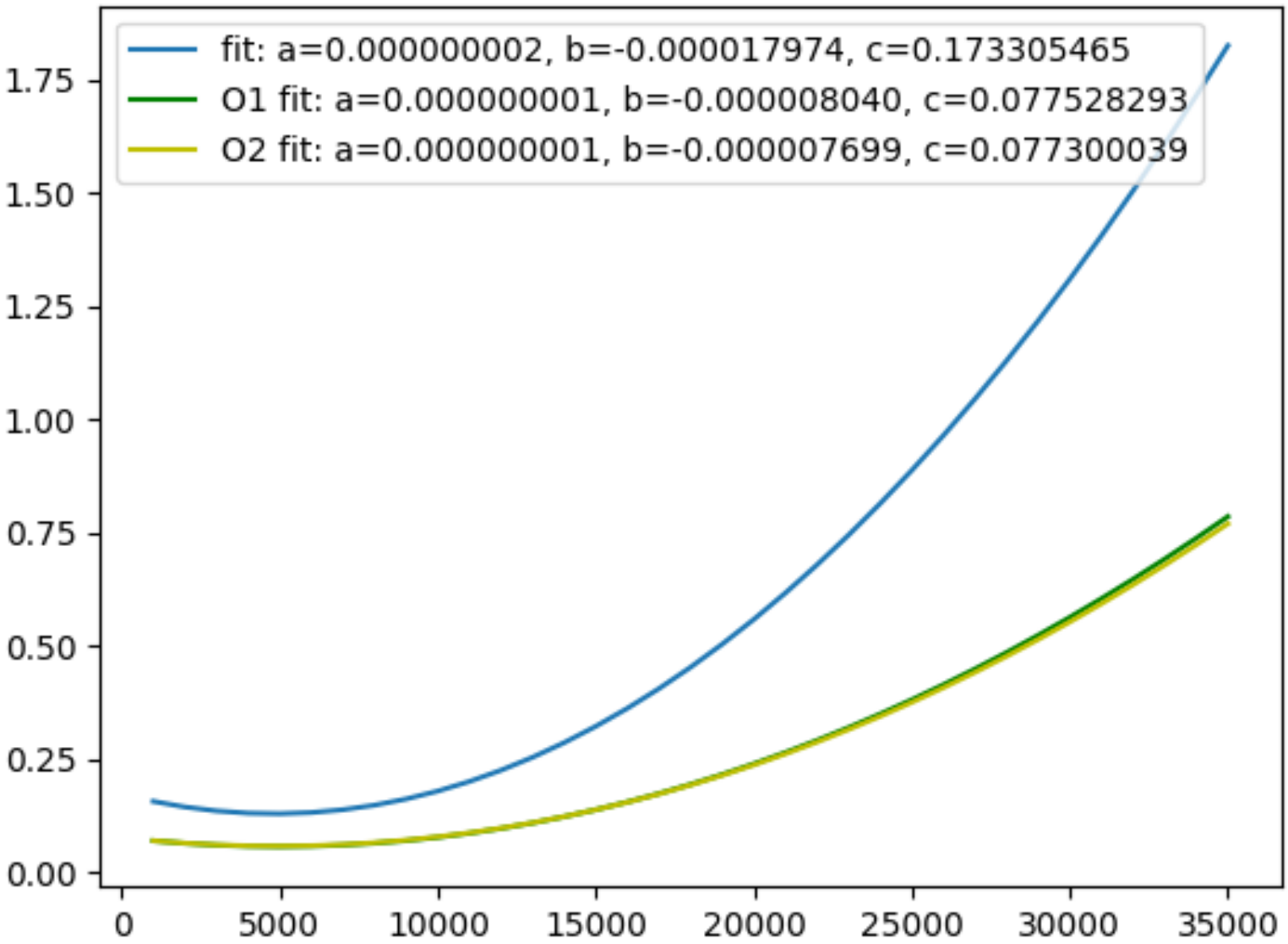


Algoritmo de selección – Análisis teórico

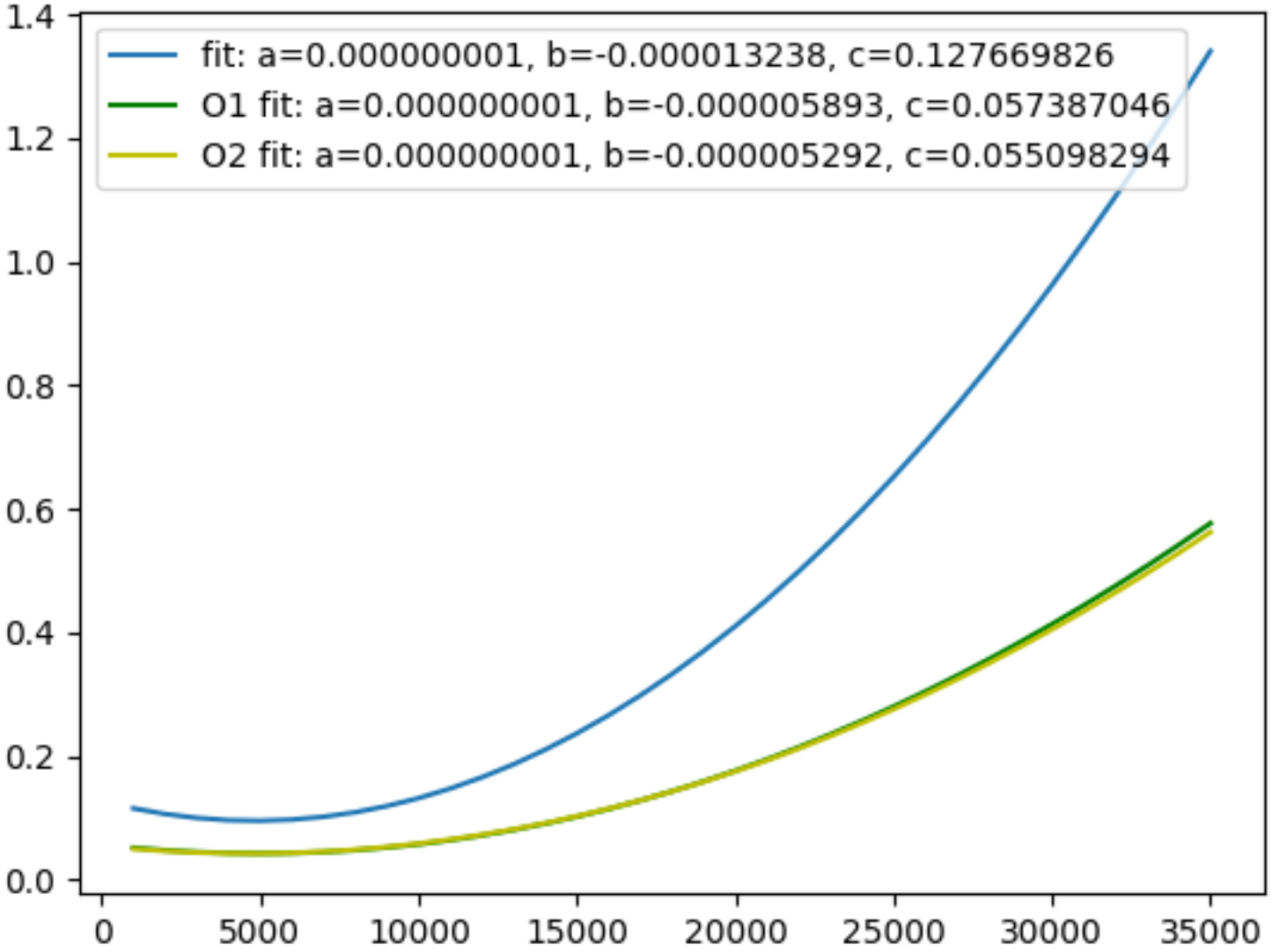
```
1  void seleccion(int T[], int num_elem)
2  {
3    seleccion_lims(T, 0, num_elem);
4  }
5
6  static void seleccion_lims(int T[], int inicial, int final)
7  {
8    int i, j, indice_menor;
9    int menor, aux;
10   for (i = inicial; i < final - 1; i++) {
11     indice_menor = i;
12     menor = T[i];
13     for (j = i; j < final; j++)
14       if (T[j] < menor) {
15       indice_menor = j;
16       menor = T[j];
17       }
18     aux = T[i];
19     T[i] = T[indice_menor];
20     T[indice_menor] = aux;
21   };
22 }
```

$$\sum_{i=0}^{n-1}(n-i-1) = \sum_{i=0}^{n-1}(n-1) - \sum_{i=0}^{n-1} i = (n-1)n - \frac{n(n+1)}{2} = \frac{n^2}{2} - \frac{n}{2}$$

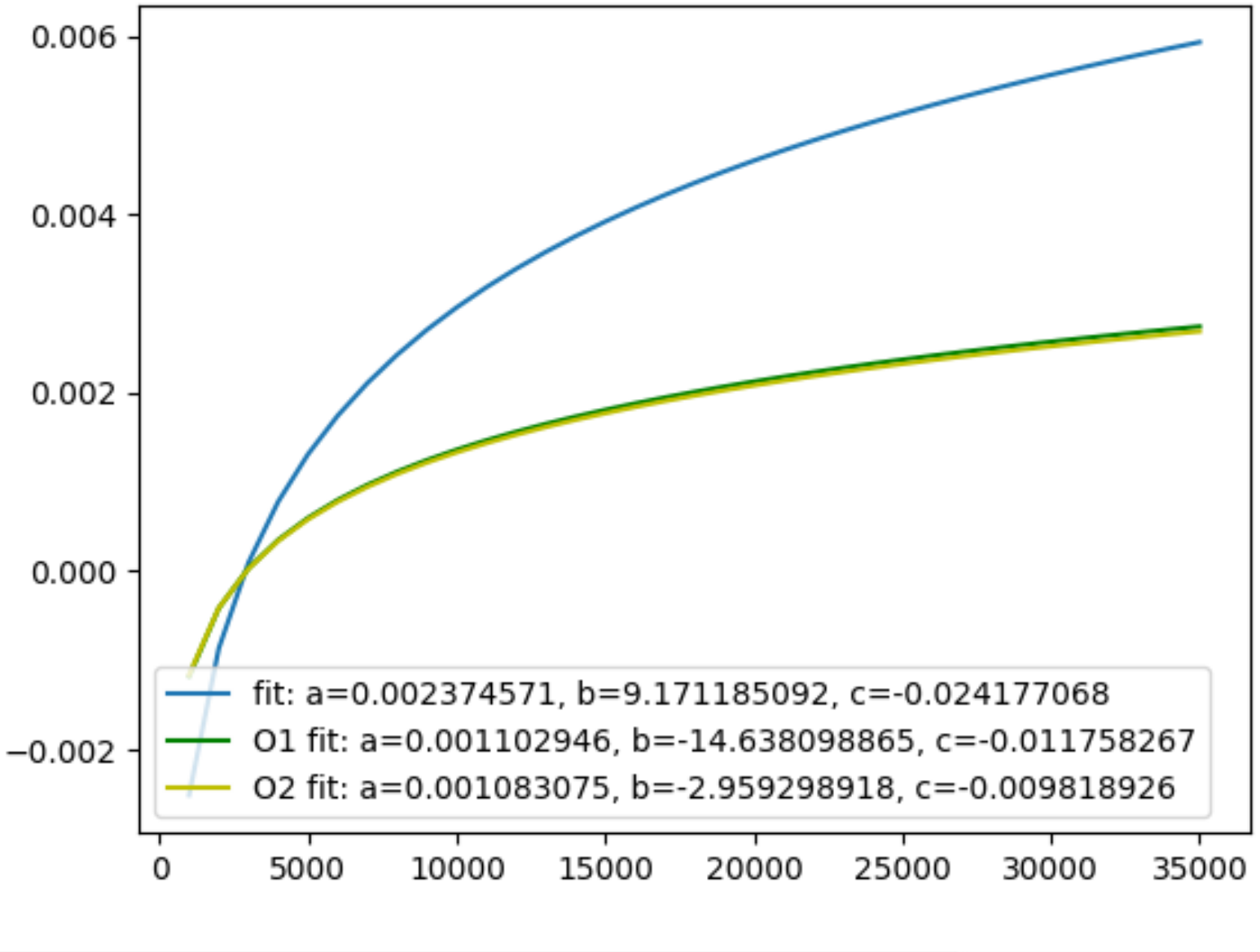
Algoritmo de selección - Elena



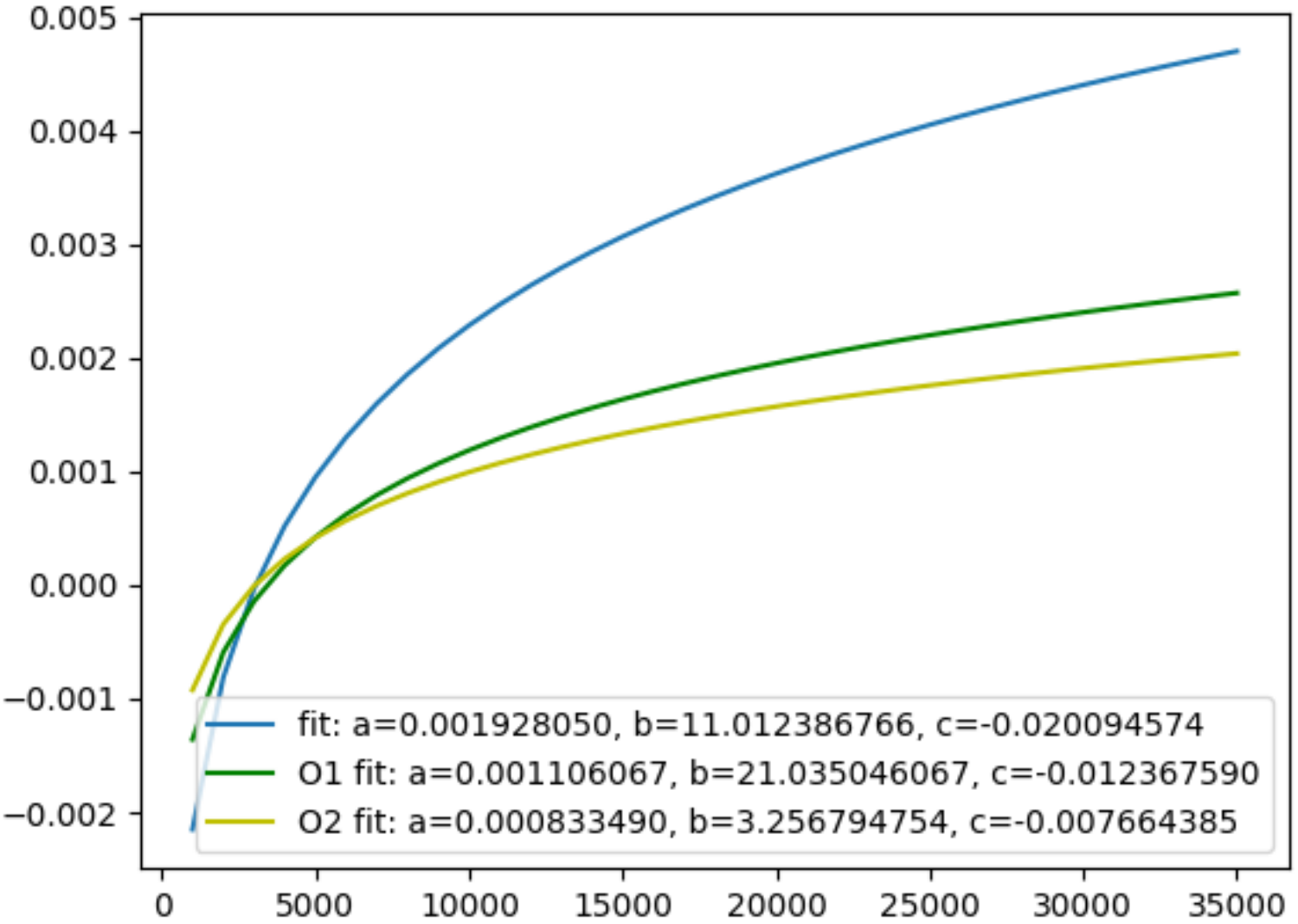
Algoritmo de selección - Antonio



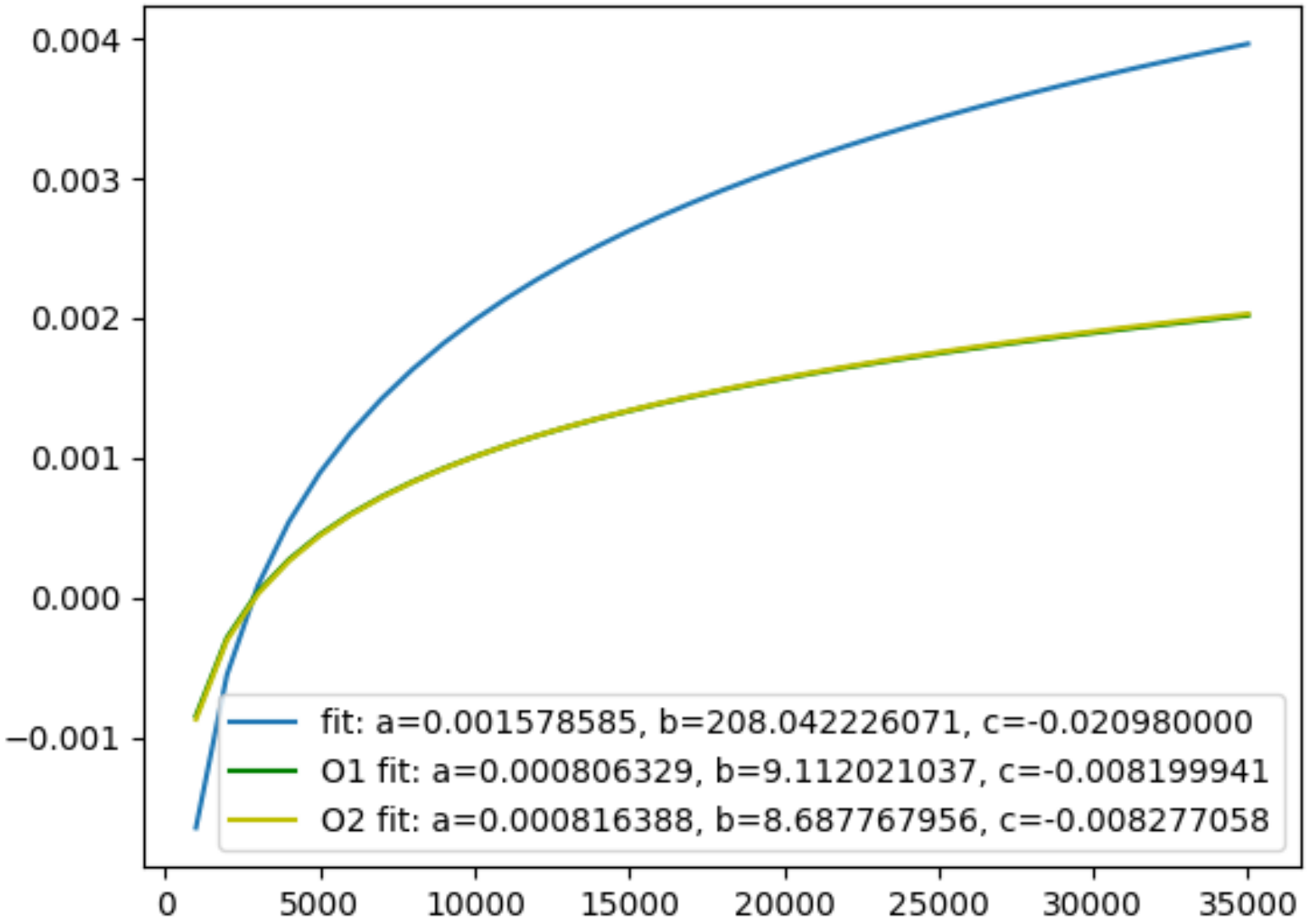
Algoritmo mergesort - Elena



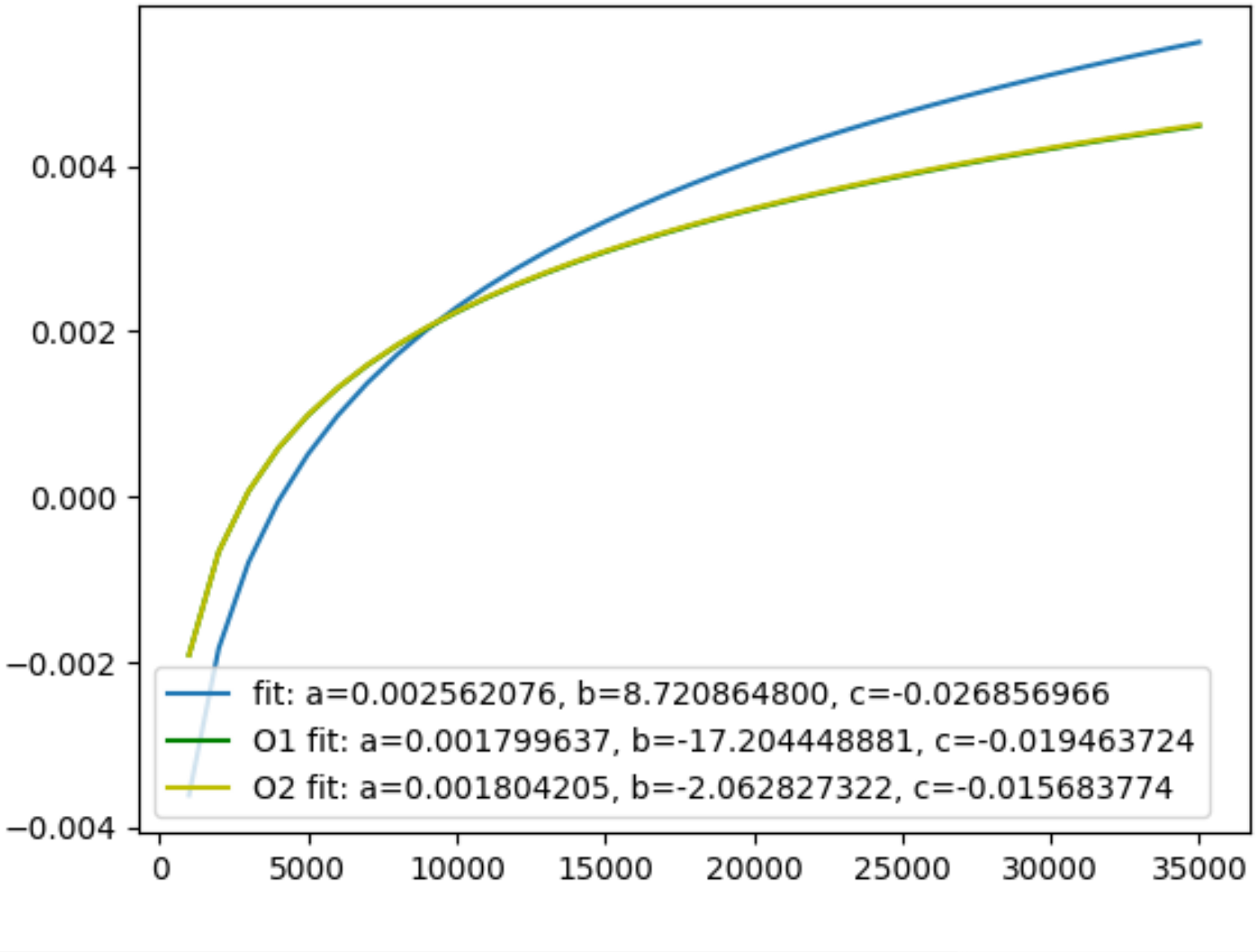
Algoritmo mergesort - Antonio



Algoritmo quicksort - Elena



Algoritmo quicksort - Antonio

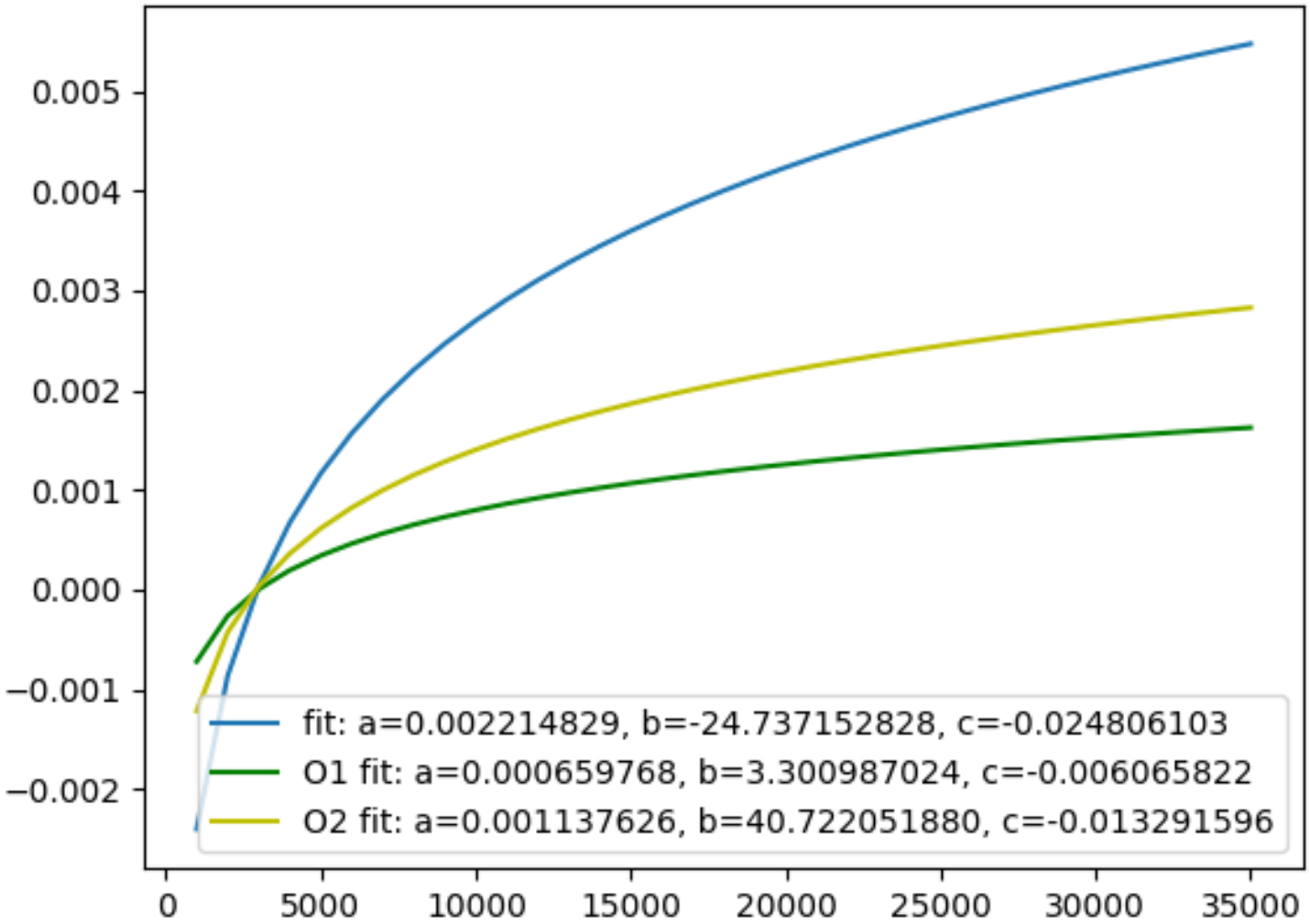


Algoritmo heapsort – Análisis teórico

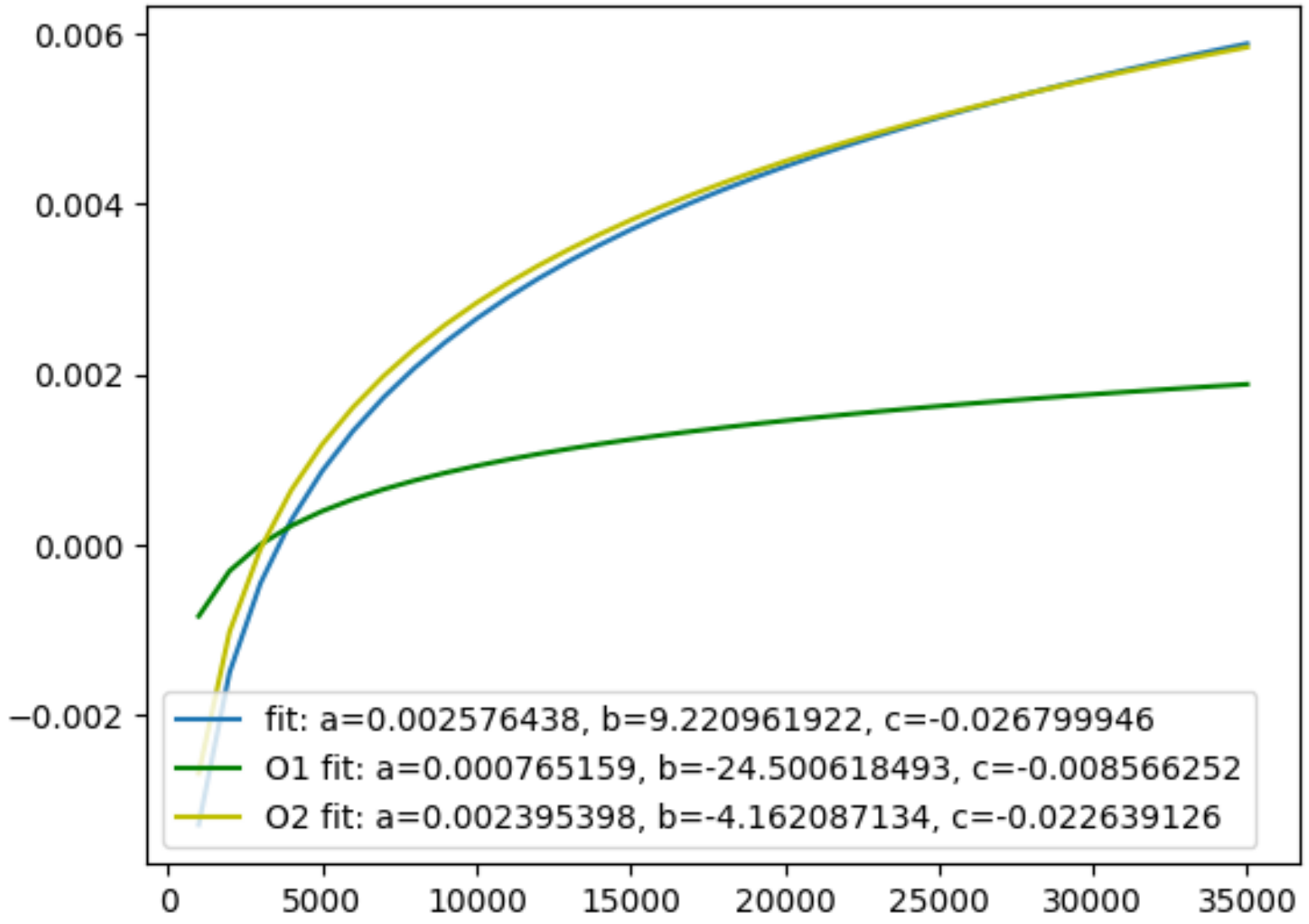
```
2  static void heapsort(int T[], int num_elem)
3  {
4    int i;
5    for (i = num_elem/2; i >= 0; i--)
6      reajustar(T, num_elem, i);
7    for (i = num_elem - 1; i >= 1; i--)
8      {
9        int aux = T[0];
10       T[0] = T[i];
11       T[i] = aux;
12       reajustar(T, i, 0);
13     }
14 }
15
16 static void reajustar(int T[], int num_elem, int k)
17 {
18   int j;
19   int v;
20   v = T[k];
21   bool esAPO = false;
22   while ((k < num_elem/2) && !esAPO)
23     {
24       j = k + k + 1;
25
26       if ((j < (num_elem - 1)) && (T[j] < T[j+1])) j++;
27       if (v >= T[j]) esAPO = true;
28
29       T[k] = T[j];
30       k = j;
31     }
32   T[k] = v;
33 }
```

$$T(n) = \frac{nlog(n)}{2} + (n-1)$$

Algoritmo heapsort - Elena



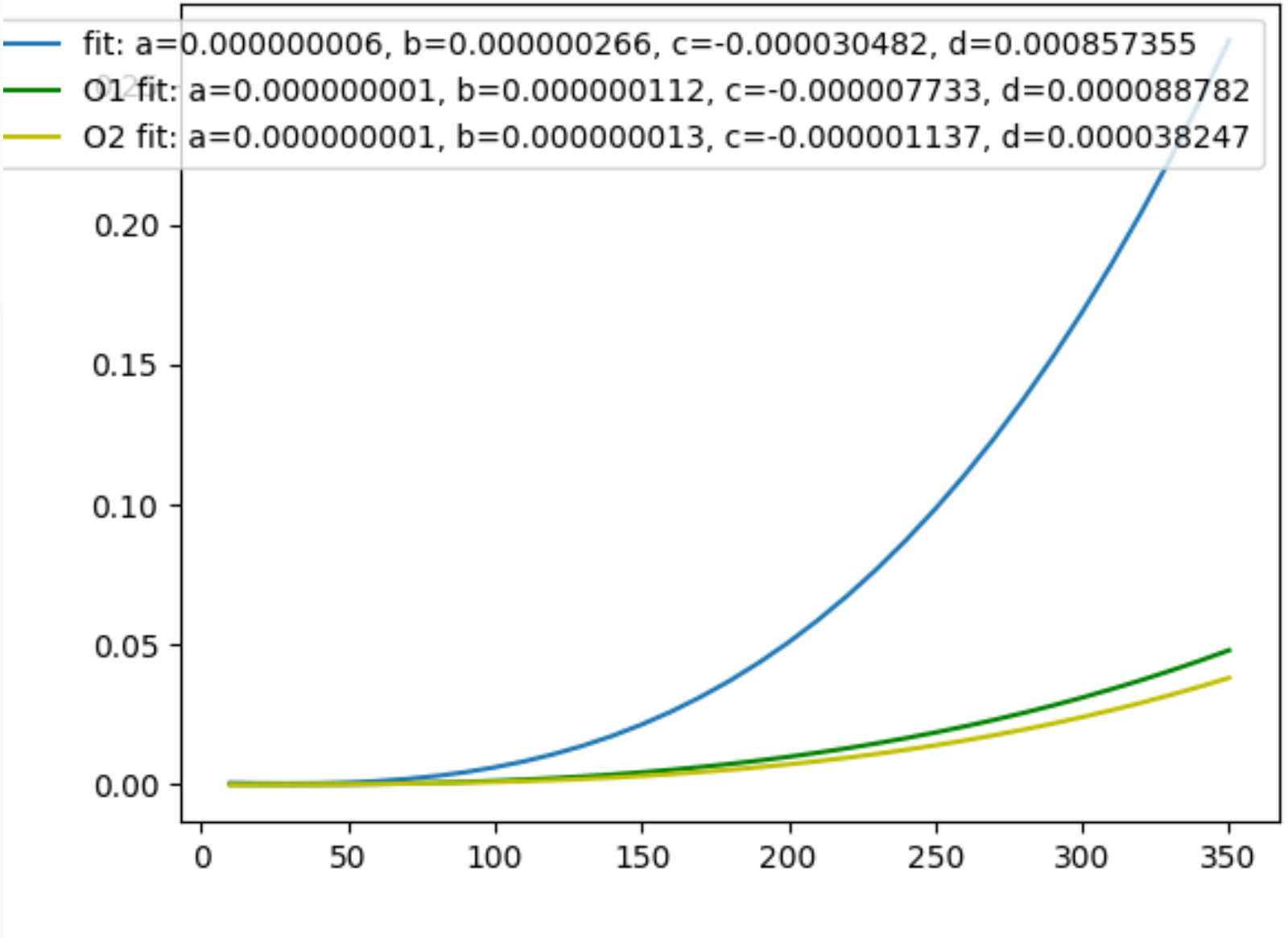
Algoritmo heapsort - Antonio



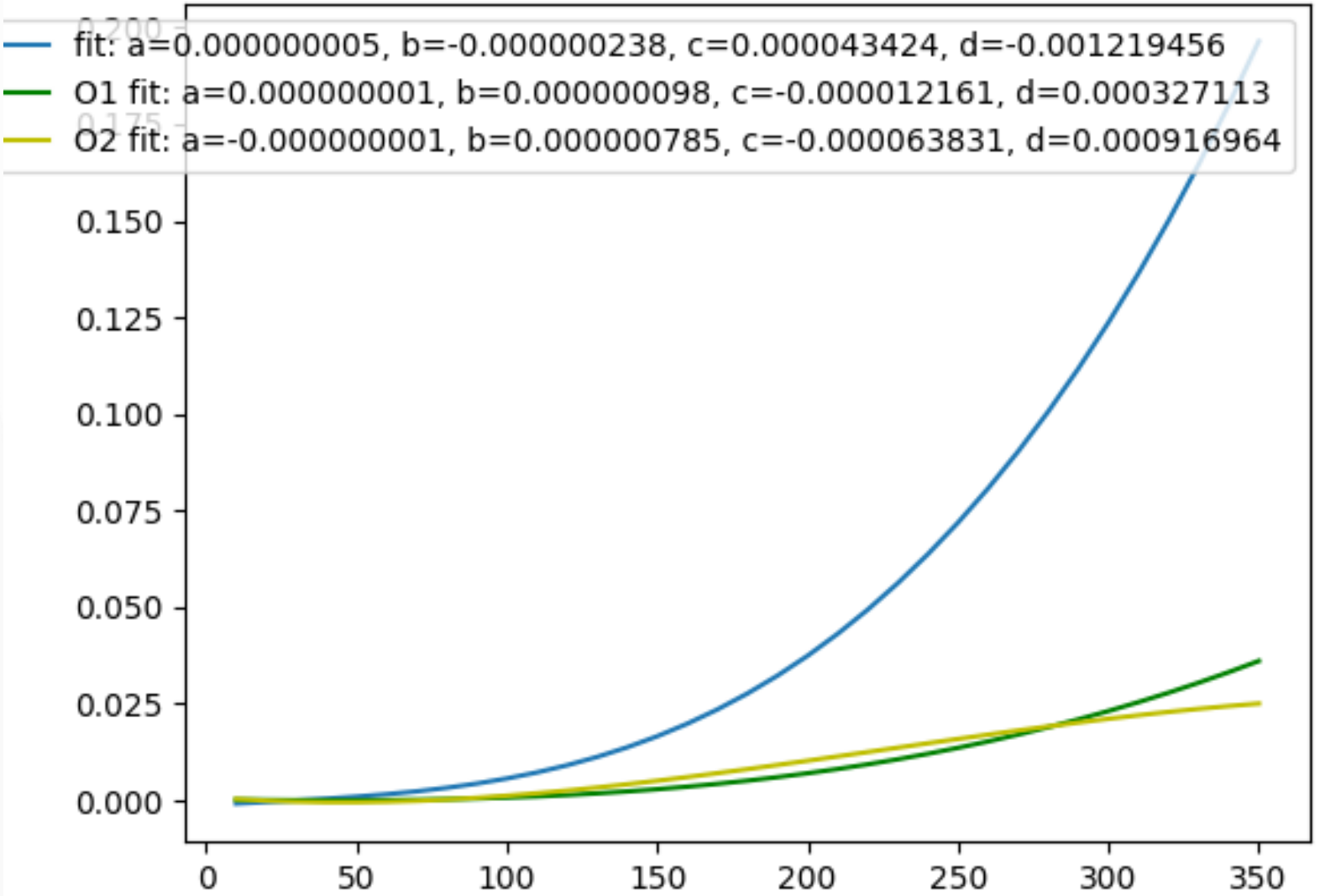
Algoritmo de floyd – Análisis teórico

```
1  void Floyd(int **M, int dim)
2  {
3    for (int k = 0; k < dim; k++)
4      for (int i = 0; i < dim; i++)
5        for (int j = 0; j < dim; j++)
6          {
7          int sum = M[i][k] + M[k][j];
8          M[i][j] = (M[i][j] > sum) ? sum : M[i][j];
9          }
10 }
```

Algoritmo de floyd - Elena



Algoritmo de floyd - Antonio



Algoritmo de hanoi – Análisis teórico

```
1 void hanoi (int M, int i, int j)
2 {
3   if (M > 0)
4     {
5       hanoi(M-1, i, 6-i-j);
6       hanoi (M-1, 6-i-j, j);
7   }
8 }
```

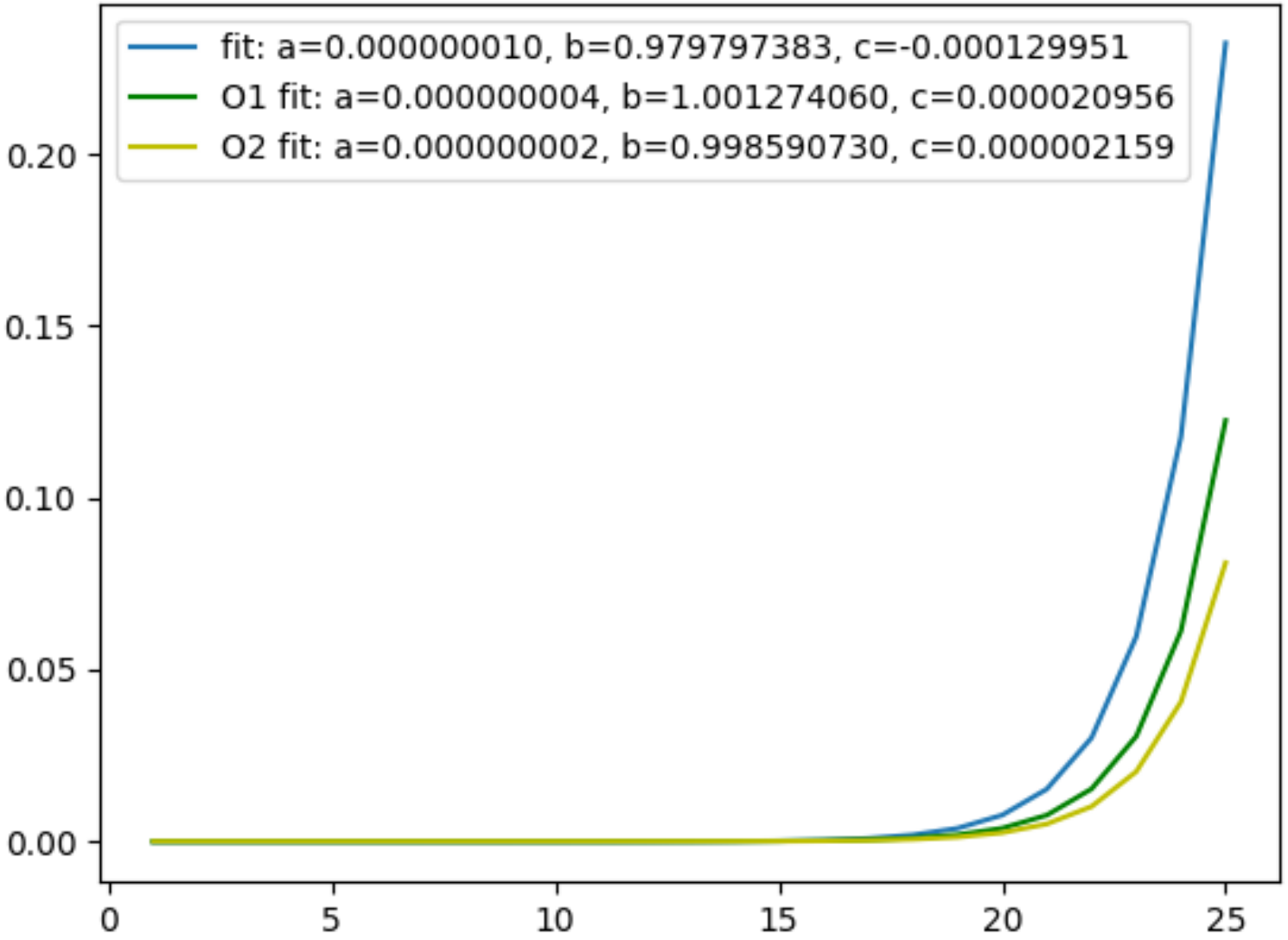
$$T(n) = \begin{cases} 2T(\frac{n}{2}) + 1 & n \geq 1 \\ 1 & n = 1 \end{cases}$$

$$\begin{aligned} T(n) &= 2\overbrace{T(n-1)}^{2T(n-2)+1} + 1 && n > 1 \\ T(n) &= 2^2 T(n-2) + 2 + 1 && n > 2 \\ T(n) &= 2^i T(n-i) + (2^{i-1} + \cdots + 2^2 + 2 + 1) && n > i \end{aligned}$$

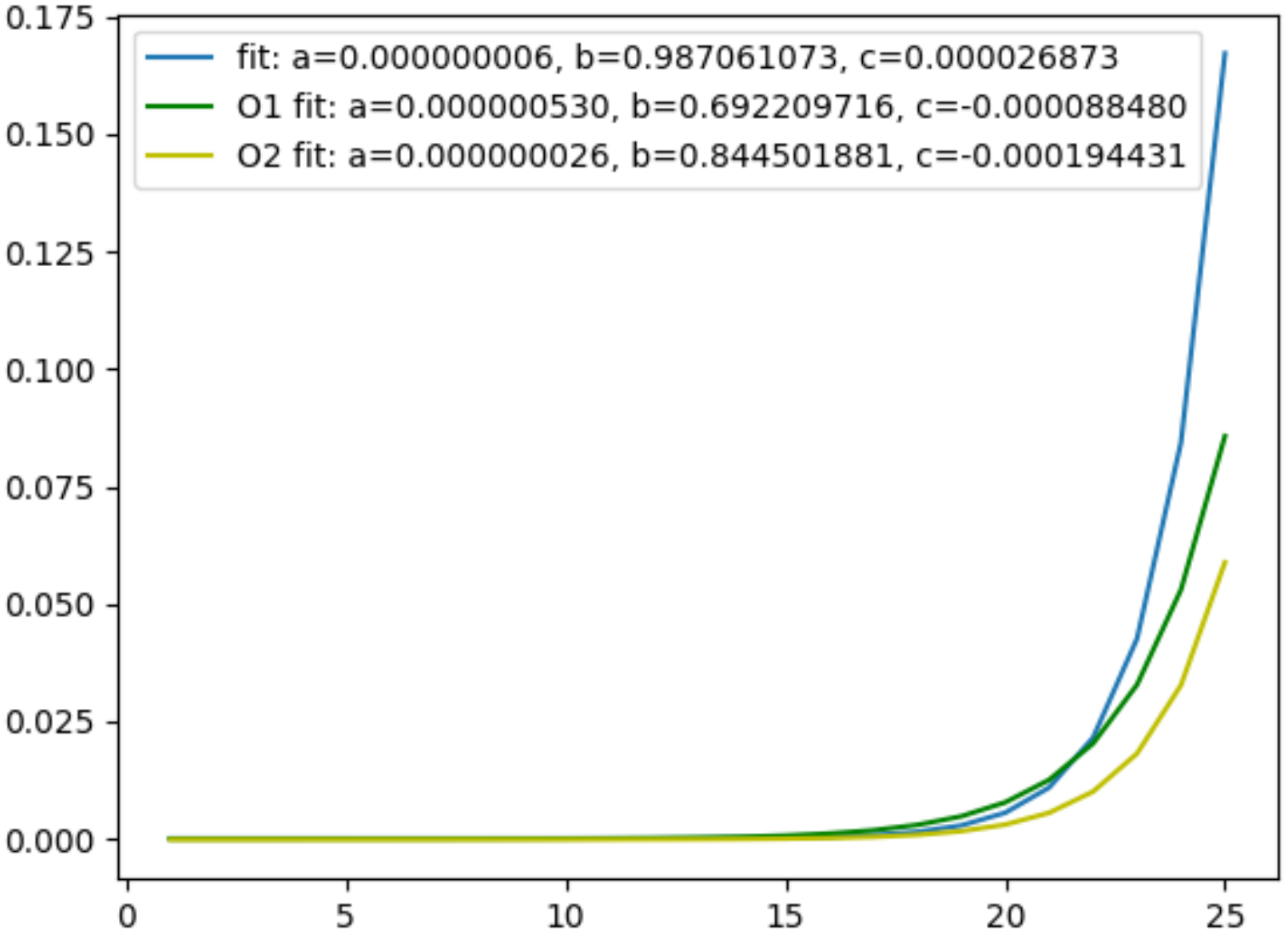
si tomamos $i = n - 1$:

$$T(n) = 2^{n-1} + 2^{n-2} + \cdots + 2^2 + 2 + 1 = \frac{2^{n-1}2 - 1}{2 - 1} = 2^n - 1$$

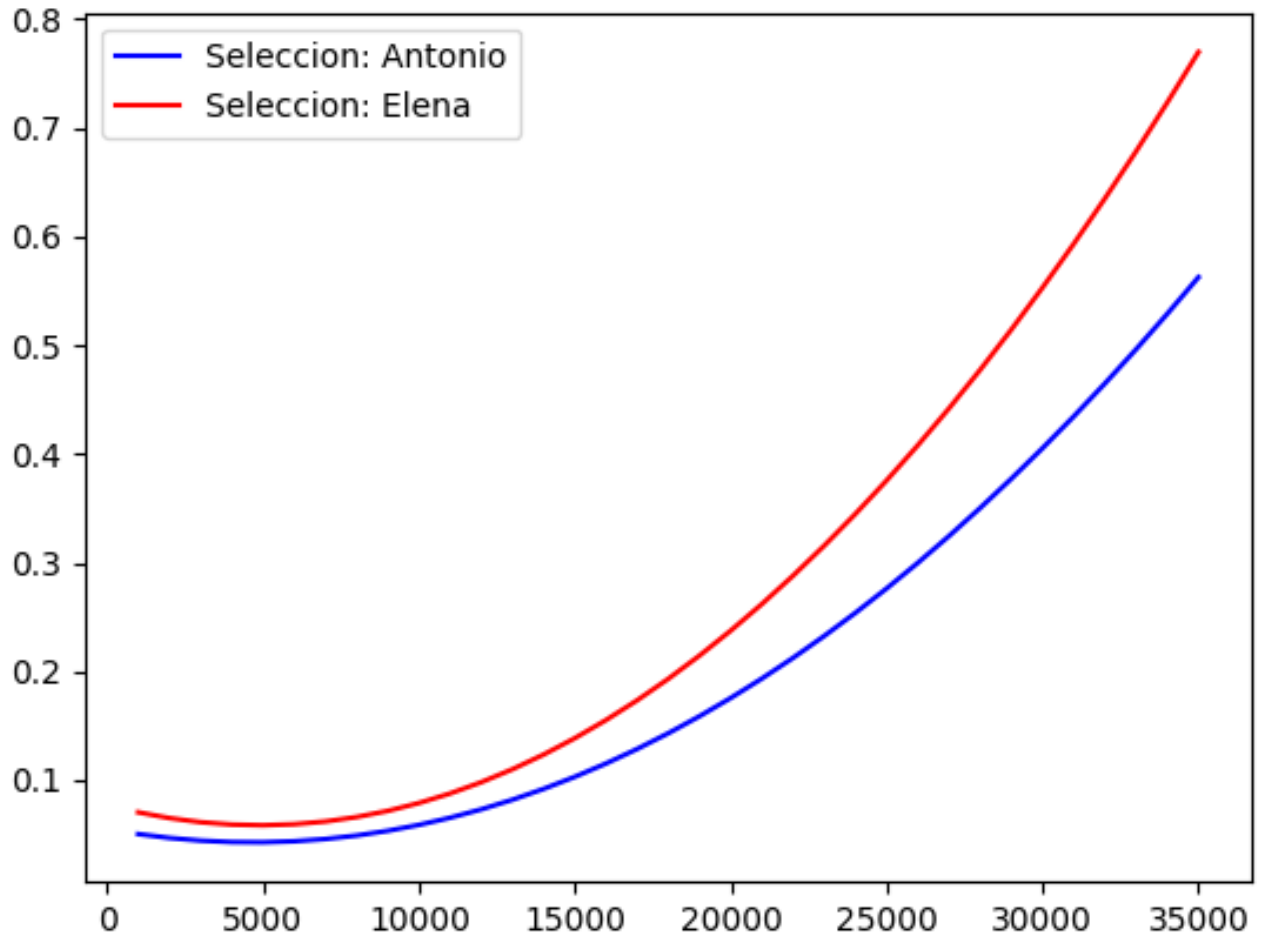
Algoritmo de Hanoi - Elena



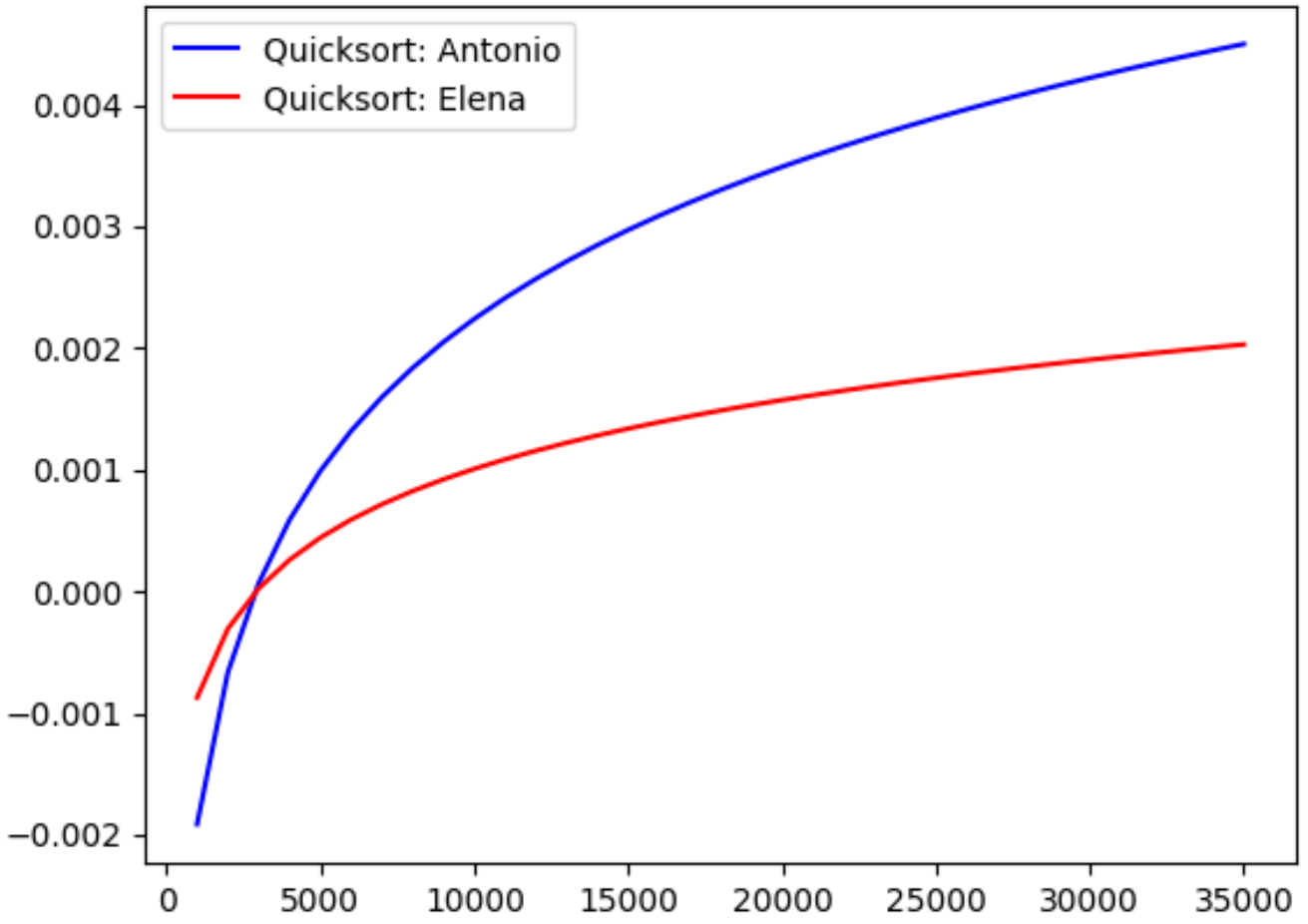
Algoritmo de Hanoi - Antonio



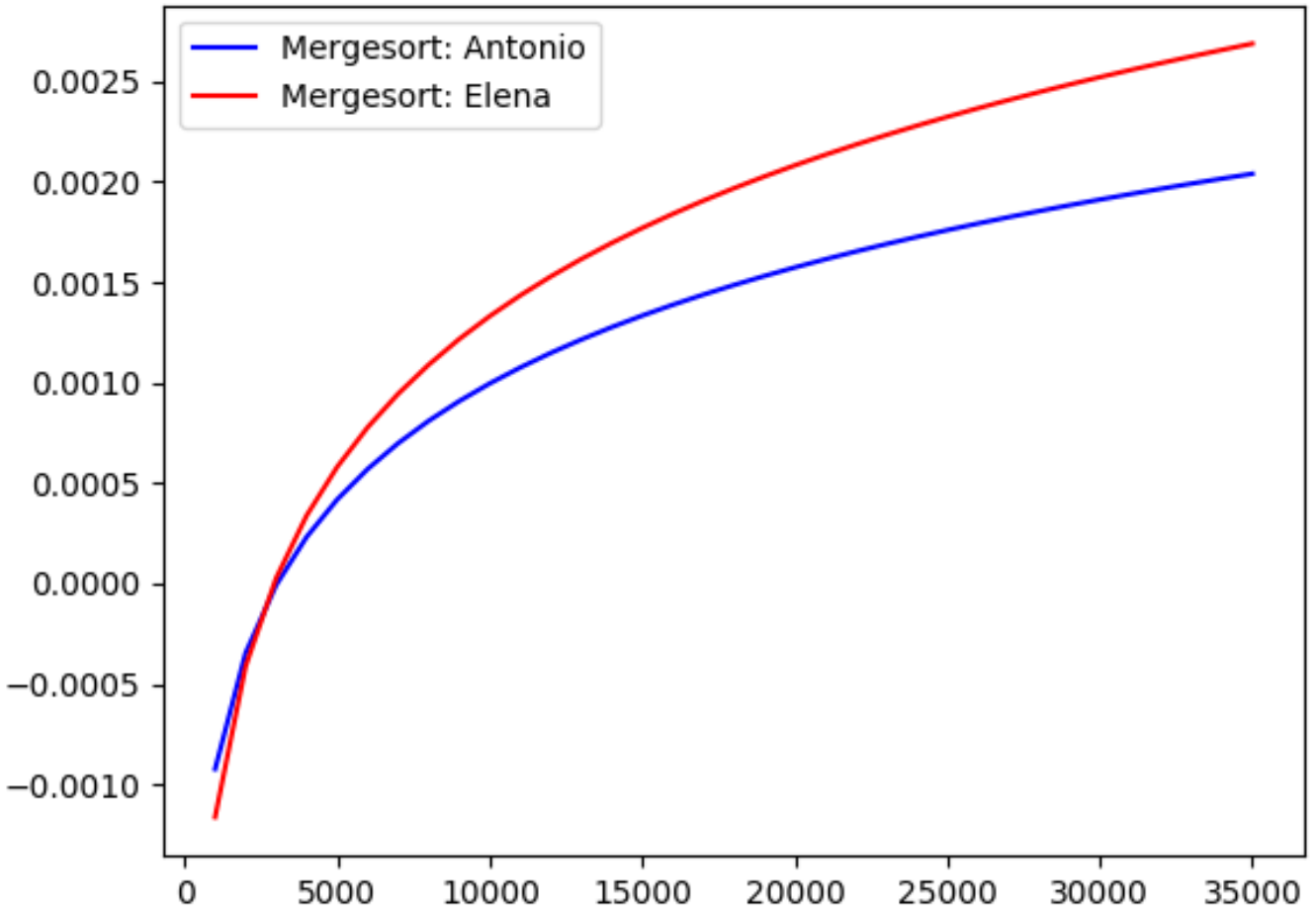
Comparación algoritmos en distintos ordenadores: selección



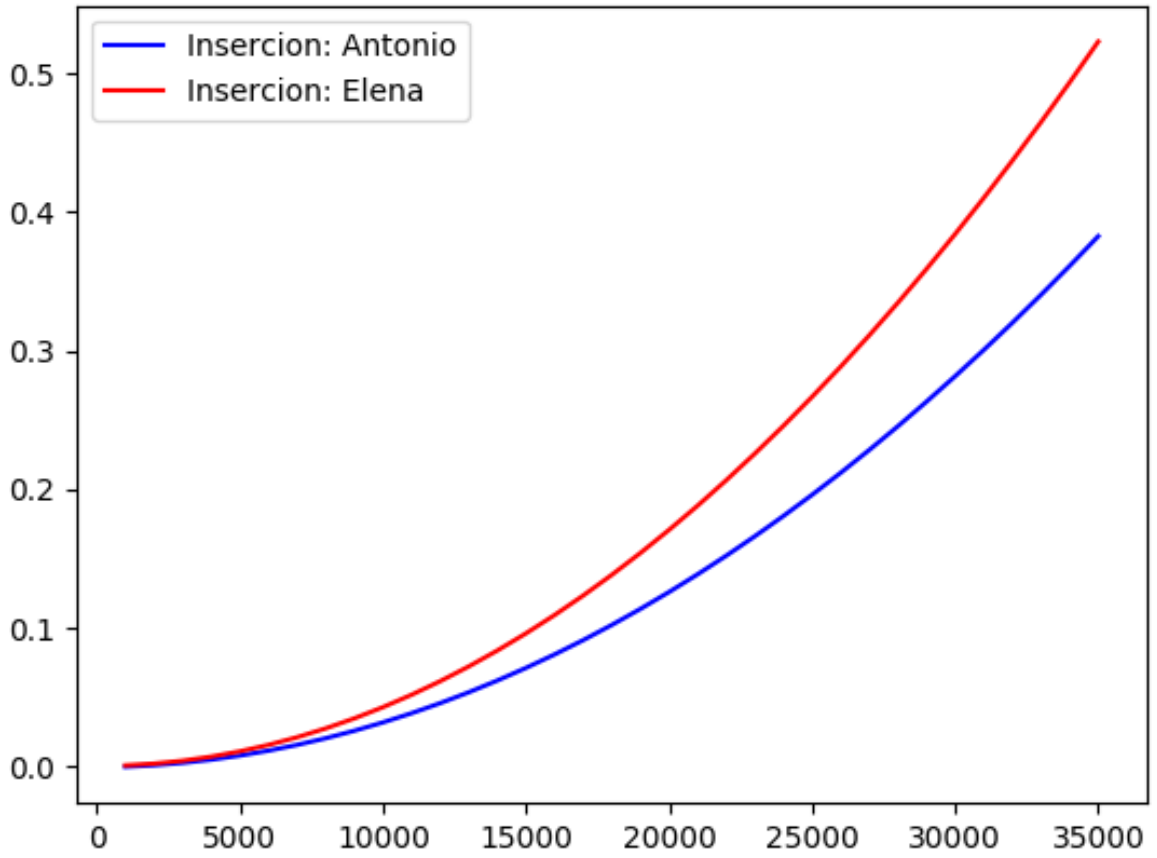
Comparación algoritmos en distintos ordenadores: quicksort



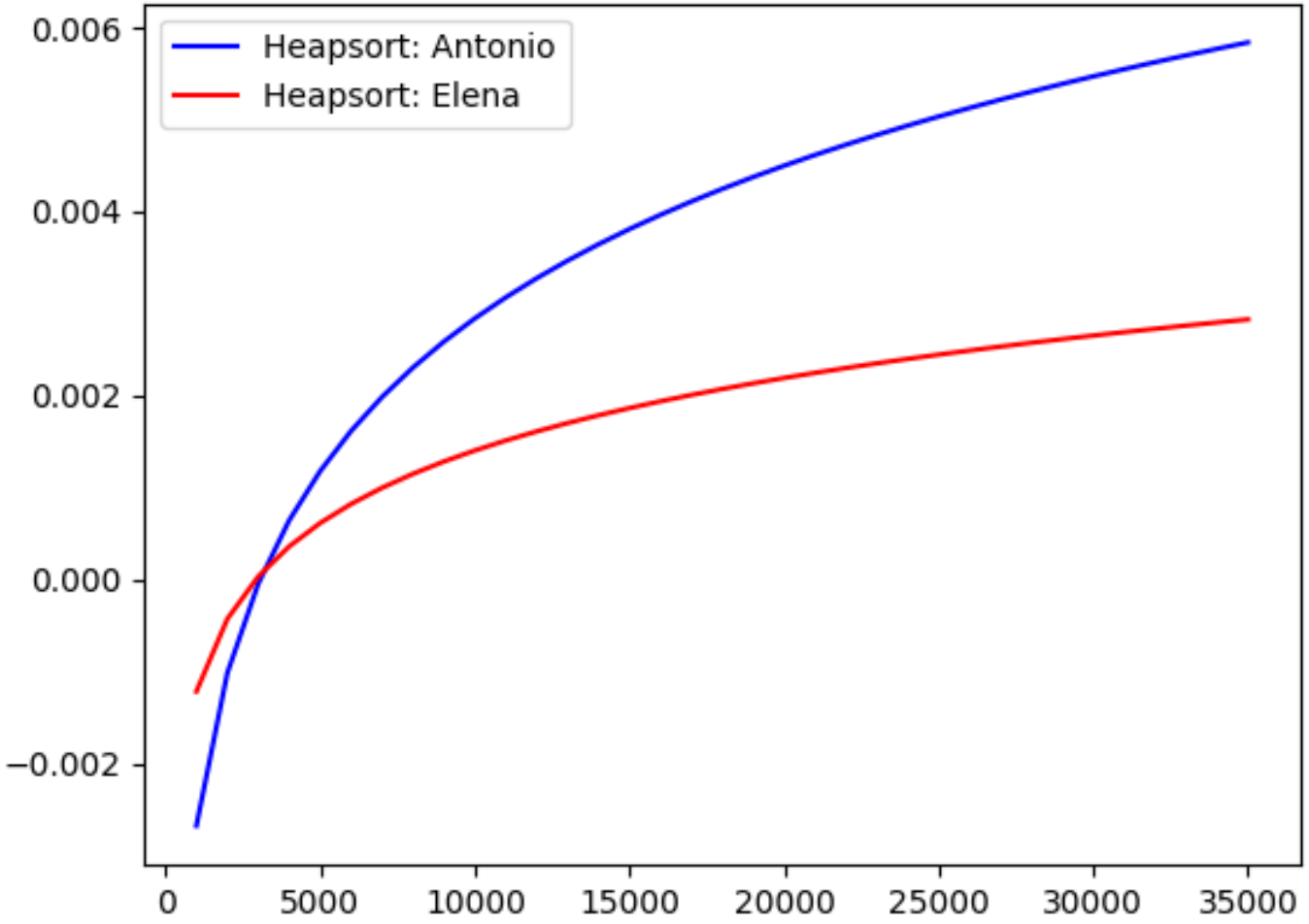
Comparación algoritmos en distintos ordenadores: mergesort



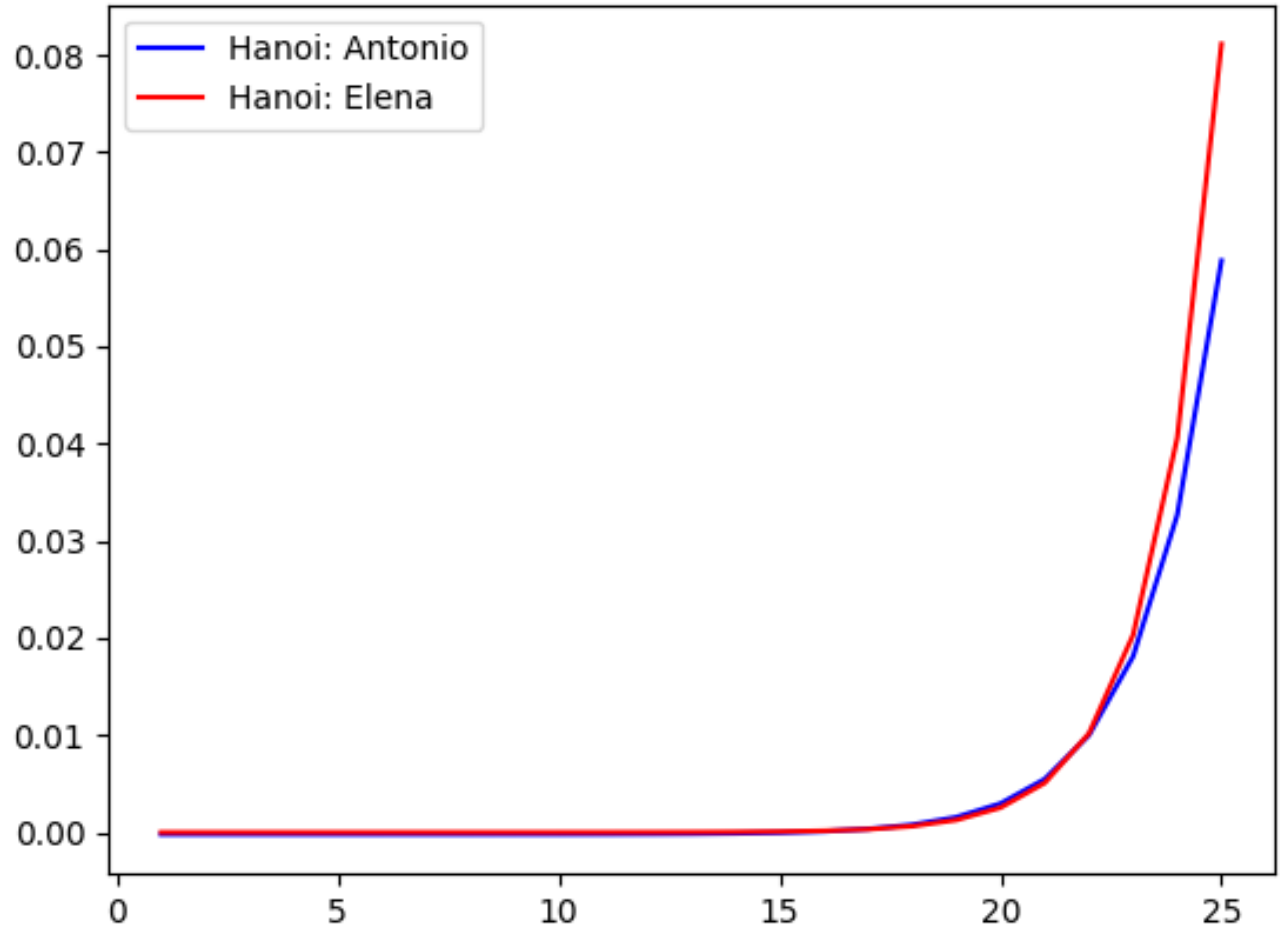
Comparación algoritmos en distintos ordenadores: inserción



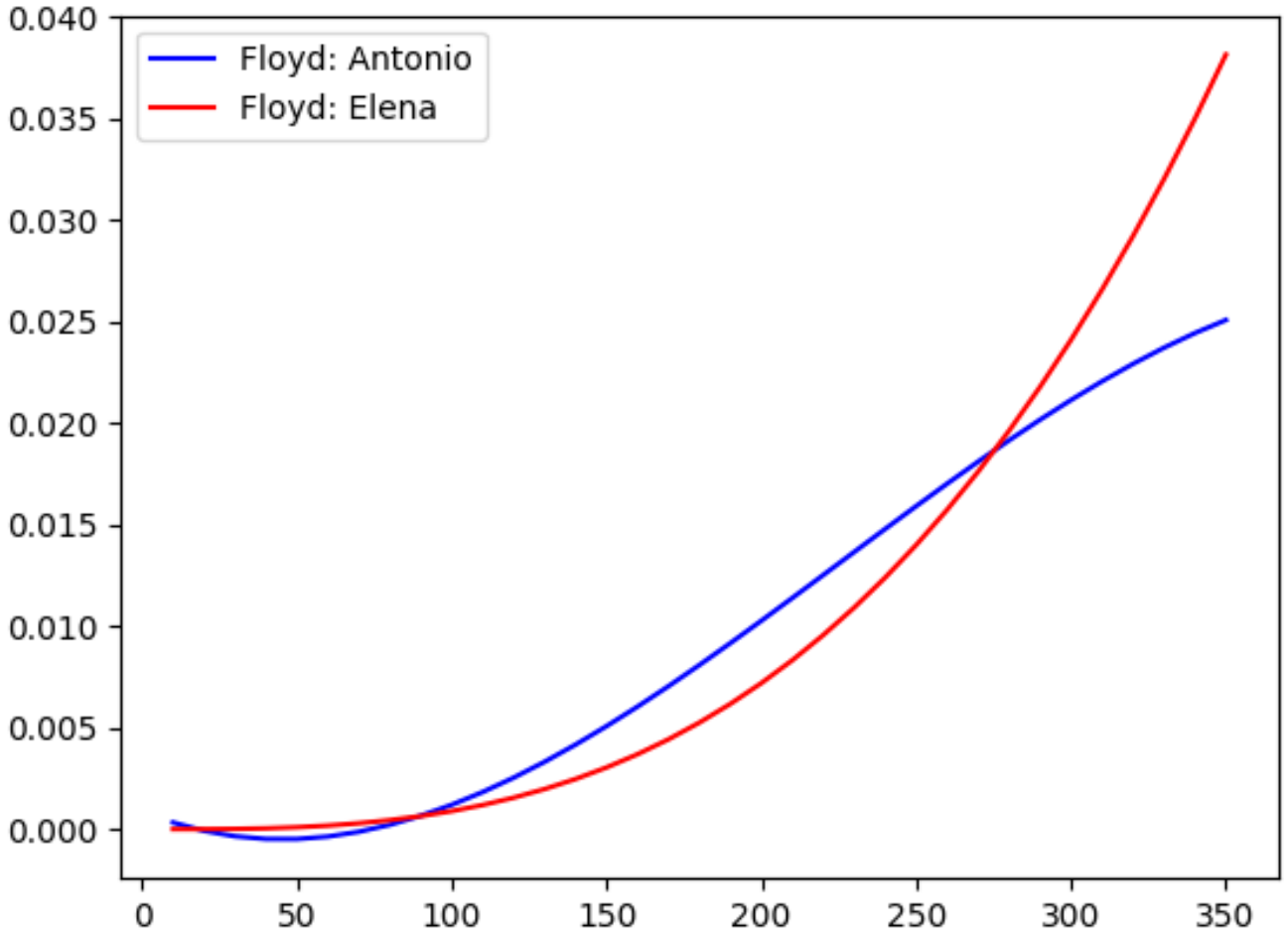
Comparación algoritmos en distintos ordenadores: heapsort



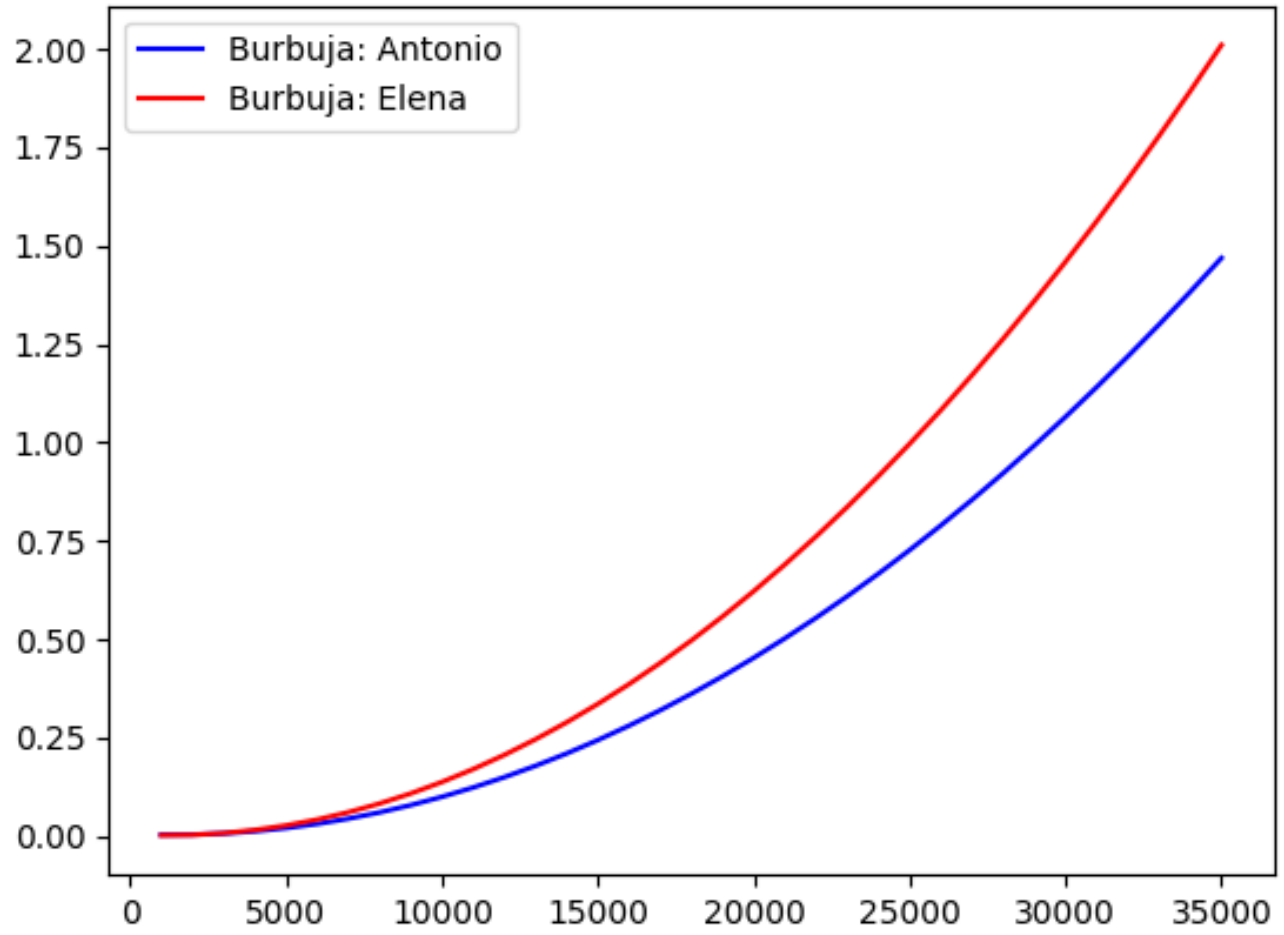
Comparación algoritmos en distintos ordenadores: hanoi



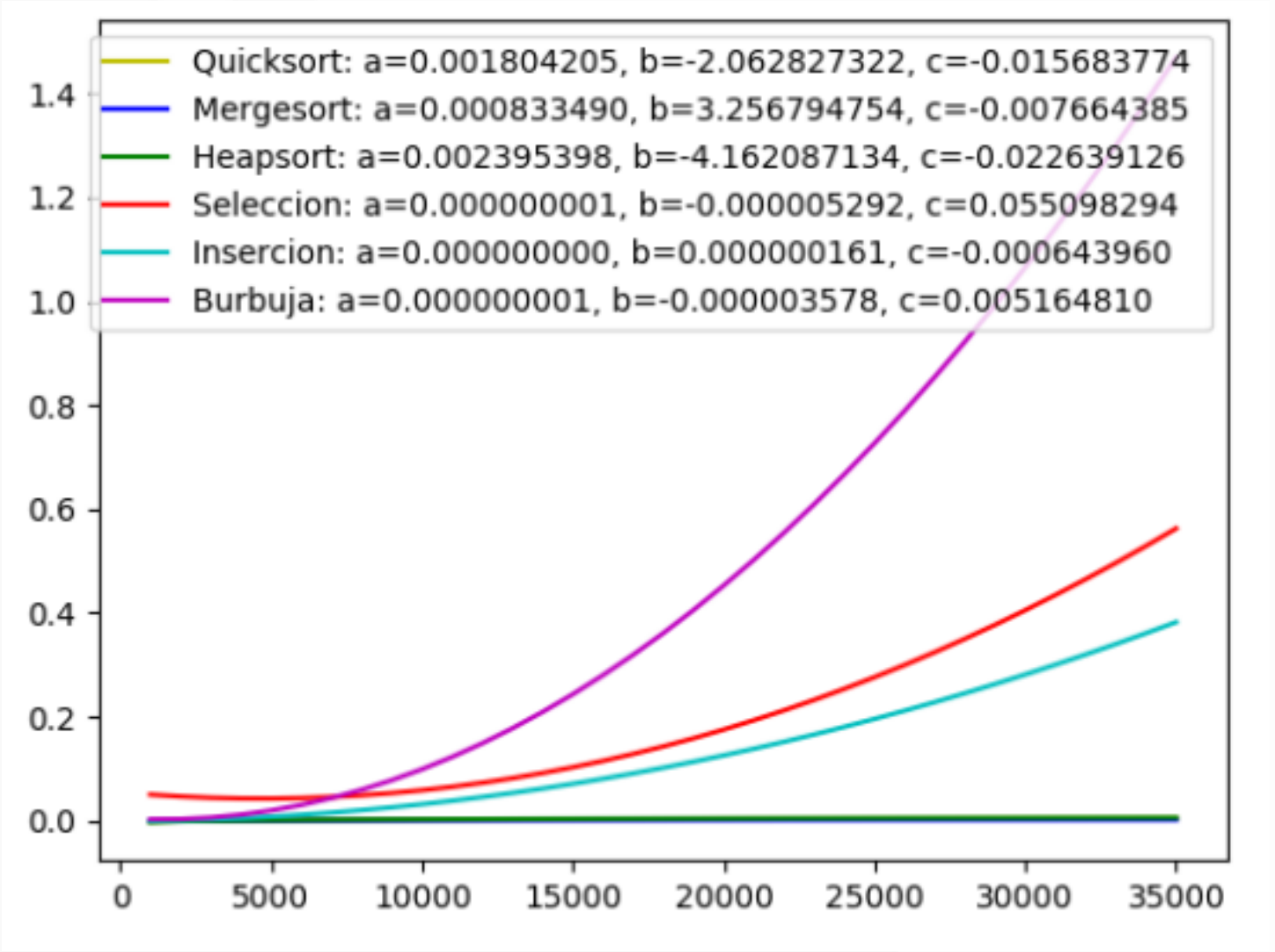
Comparación algoritmos en distintos ordenadores: floyd



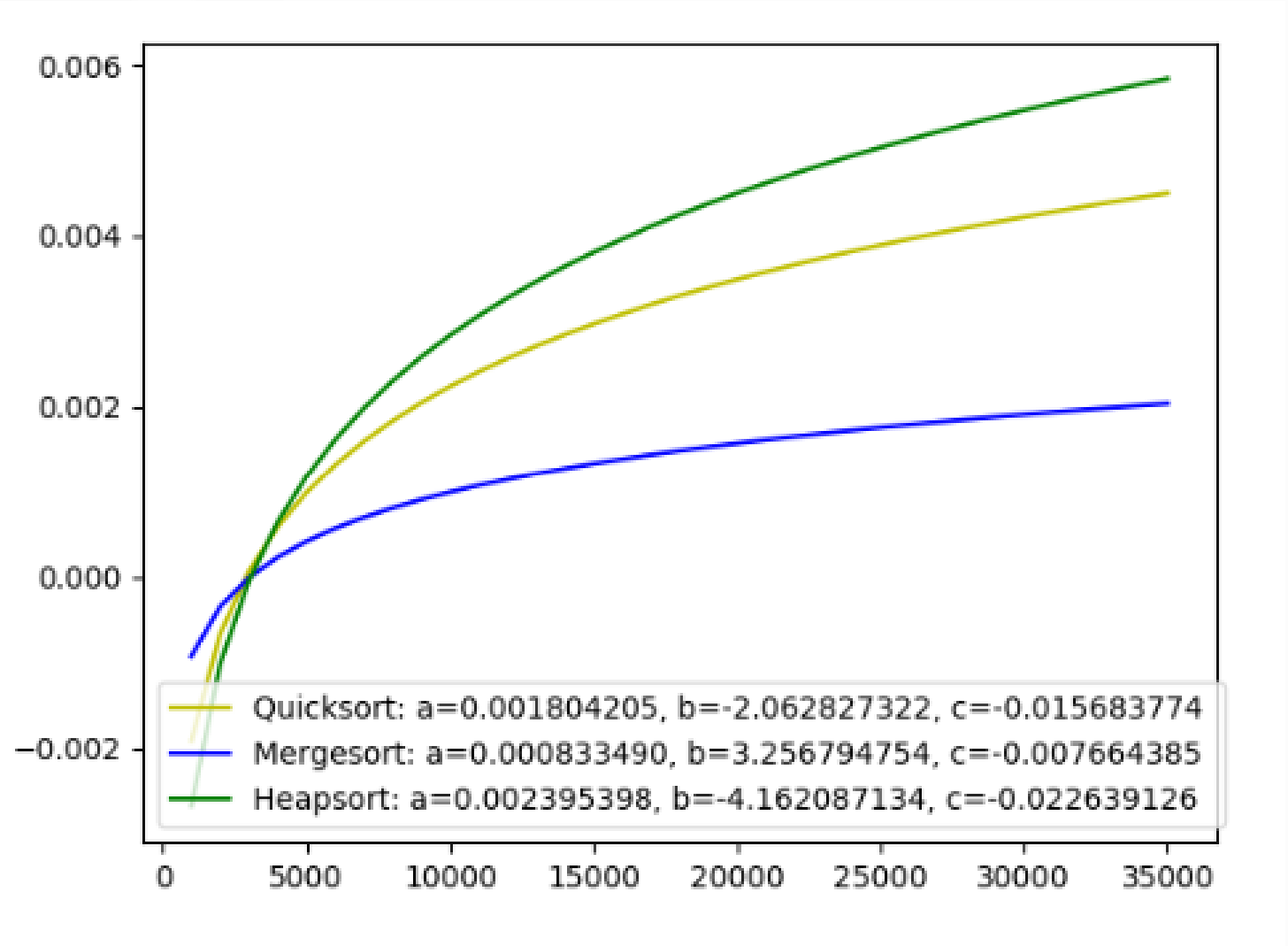
Comparación algoritmos en distintos ordenadores: burbuja



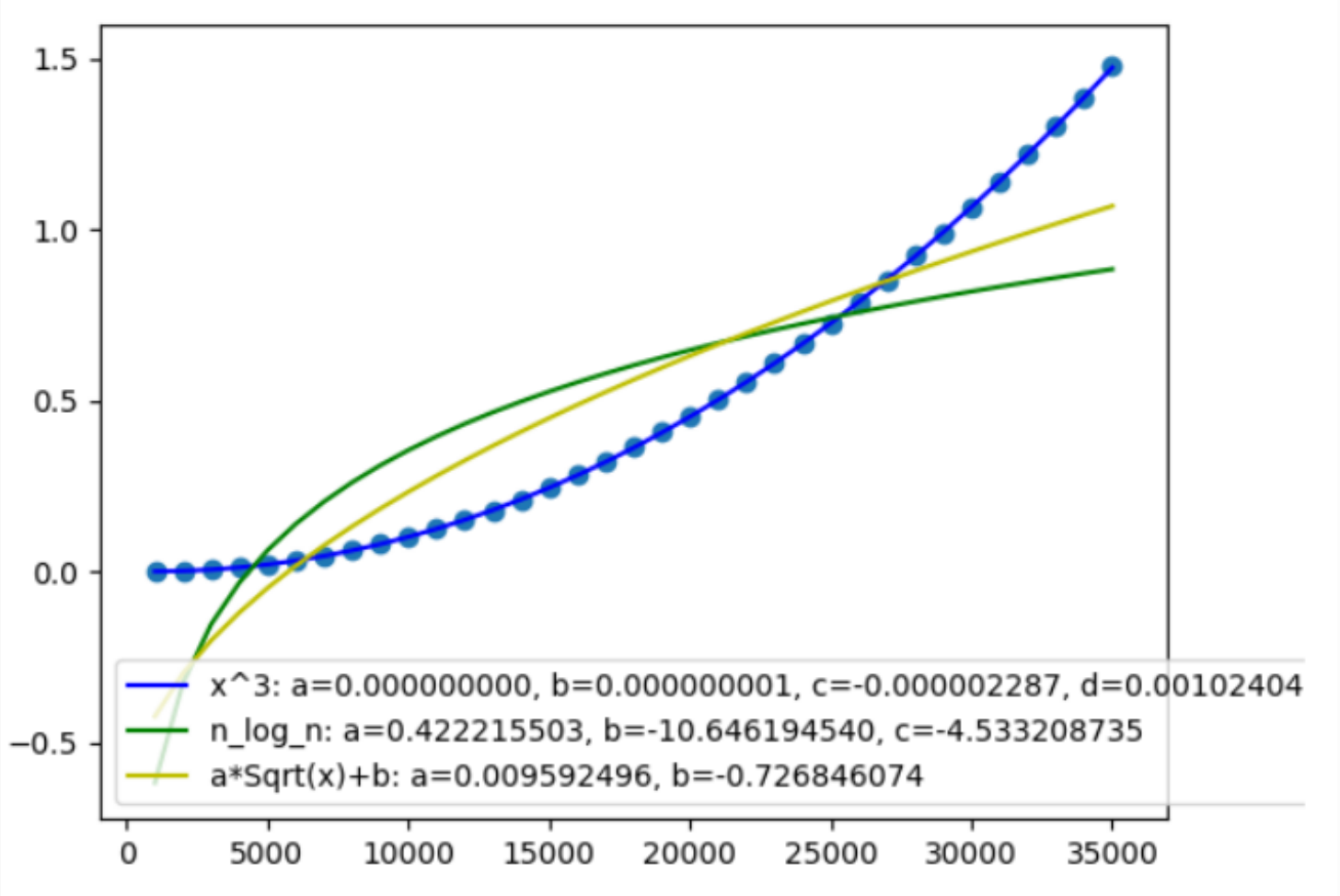
Comparación entre algoritmos de búsqueda



Comparación entre algoritmos *sort



Variación de ajuste según la función-burbuja -02



FIN