# Laboratorio 2: Algunas soluciones

## Ejercicio 1: k-ésimo elemento más chico

```
int k_esimo(int a[], int length, int k) {
   int ppiv, lft, rgt;
   lft = 0;
   rgt = length - 1;
   ppiv = partition(a, lft, rgt);
   while (ppiv != k) {
       if (ppiv < k) {
           lft = ppiv + 1;
       } else {
           rgt = ppiv - 1;
       }
       ppiv = partition(a, lft, rgt);
   }
   return a[k];
}
```

## Ejercicio 2: Cima con búsqueda secuencial

### Tiene cima

```
bool tiene_cima(int a[], int length) {
   // primero recorremos la parte creciente
   // frenamos cuando termina el arreglo o deja de ser creciente
   int k = 0;
   while (k < length - 1 && a[k] < a[k+1]) {
       k++;
   }
   // terminamos. acá vale:
   // - arreglo creciente hasta posición k
   // - termina en posición k (k == length-1)
   //   || el que sigue es mayor o igual (a[k] >= a[k+1])

   // ahora recorremos la parte decreciente
   // frenamos cuando termina el arreglo o deja de ser decreciente
```

```
    while (k < length - 1 && a[k] > a[k+1]) {
        k++;
    }

    // tiene cima si y sólo si llegamos hasta el final del arreglo
    return k == length - 1;
}
```

Cima

```
int cima(int a[], int length) {
    // recorremos la parte creciente
    // frenamos cuando termina el arreglo o deja de ser creciente
    int k = 0;
    while (k < length - 1 && a[k] < a[k+1]) {
        k++;
    }
    // dos opciones:
    // 1. llegamos al final del arreglo que es todo creciente
    // 2. dejó de ser creciente entre la posición k y k+1

    // en ambos casos la cima se encuentra en la posición k:
    return k;
}
```

## Ejercicio 3: Cima con divide y vencerás

```
int cima_log(int a[], int length) {
    return cima_rec(a, 0, length-1, length);
}

// PRE: tiene_cima(a, length)
int cima_rec(int a[], int lft, int rgt, int length) {
    int result;
    int mid = (lft + rgt) / 2;
    if (es_cima(a, mid, length)) {
        result = mid;
    } else if (izq_cima(a, mid, length)) {
        result = cima_rec(a, lft, mid-1, length);
```

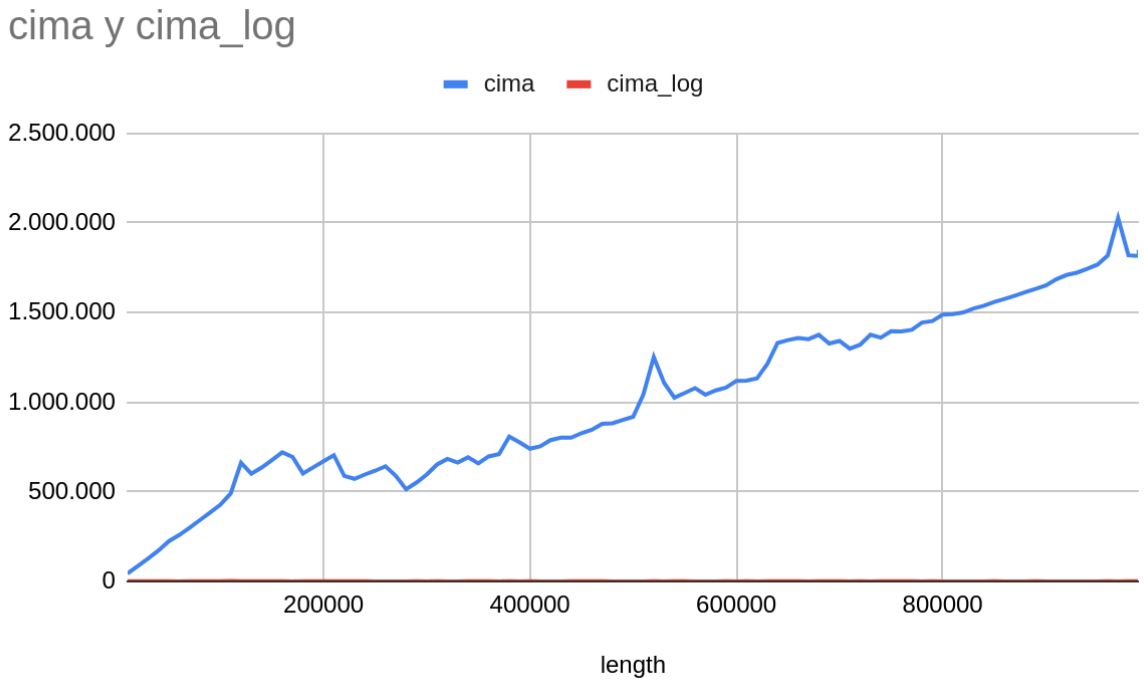
```
    } else if (der_cima(a, mid, length)) {
        result = cima_rec(a, mid+1, rgt, length);
    }
    return result;
}

// PRE: tiene_cima(a, length)
// Indica si la posición i del arreglo a es la cima.
bool es_cima(int a[], int i, int length) {
    return (i == 0 || a[i] > a[i-1]) && (i == length-1 || a[i] >
a[i+1]);
}

// PRE: tiene_cima(a, length)
// Indica si la cima está a la izquierda de la posición i del arreglo
bool izq_cima(int a[], int i, int length) {
    return i > 0 && a[i] < a[i-1];
}

// PRE: tiene_cima(a, length)
// Indica si la cima está a la derecha de la posición i del arreglo
bool der_cima(int a[], int i, int length) {
    return i < length-1 && a[i] < a[i+1];
}
```

## Ejercicio 4: Comparación


cima y cima_log
cima
cima_log
2.500.000
2.000.000
1.500.000
1.000.000
500.000
0
200000
400000
600000
800000
length