

Actividad 1.2: Algoritmos de Búsqueda y Ordenamiento

Complejidad: $O(n^2)$

```
void ordenaIntercambio(vector<int> &nums)
{
    int i, j, aux;
    for (i = 0; i <= nums.size() - 2; i++)
    {
        /*se esta haciendo una comparación signed con unsigned int en
        el condicional del for. ¿Cómo se repara? ¿Cómo hago para que la función
        .size() me regrese un signed int? */
        for (j = i + 1; j <= nums.size() - 1; j++)
        {
            if (nums[i] > nums[j])
            {
                swapElements(nums, i, j);
            }
        }
    }
}
```

Analisis de algoritmos: tarea 1.2 $\sum_{k=1}^n k = \frac{n(n+1)}{2}$

Ordena Intercambio

int i, j, aux

for (i = 0 to n-2)

{

for (j = i+1 to n-1) $\sum_{j=i+1}^{n-1} n - (i+1) = \frac{n(n+1)}{2} - 3 - (i+1)$

{

if (nums[i] > nums[j]) $\sum_{j=i+1}^{n-1} (n-1) - (i+1) = \frac{n(n+1)}{2} - 4$

{

swap(i, j)

$\sum_{j=i+1}^{n-1} (n-1) - (i+1) = \frac{n(n+1)}{2} - 5$

}

}

}

$T(n) = C_1 + C_2(n-1) + C_3\left(\frac{n(n+1)}{2}\right) + C_4\left(\frac{n(n+1)}{2}\right) + C_5\left(\frac{n(n+1)}{2}\right)$

$T(n) = C_1 + C_2n - C_1 + n^2\left(\frac{C_3 + C_4 + C_5}{2}\right)$

$O(n^2)$

Complejidad: $O(n^2)$

```
void ordenaBurbuja(vector<int> &nums)
{
    int i, j;
    for (i = 0; i < nums.size() - 1; i++)
    {
        for (j = 0; j < nums.size() - i - 1; j++)
        {
            if (nums[j] > nums[j + 1])
            {
                /* Nos interesa hacer el cambio de valores del arreglo en una posición después de que se
                cumpla la condición 'j > j + 1'. Es por esto que el cambio lo hacemos a la variable j + 1.
                */
                swapElements(nums, j, j + 1);
            }
        }
    }
}
```

ordenar Burbuja

∇ se hace n veces porque chequea si $i > n-1$, siempre se hace una iteración más?

for ($i=0$ to $n-1$; $i++$) $C_1 \cdot n$

{

for ($j=0$ to $n-i-1$; $j++$) $C_2 + \sum_{j=0}^{n-i-1}$

{

if ($nums[j] > nums[j+1]$) $\sum_{j=0}^{n-i-1} (n-i-2) \cdot C_3$

if swap($j, j+1$) $\sum_{j=0}^{n-i-1} (n-i-2) \cdot C_4$

}

}

$$T(n) = C_1 n + C_2 \left(\frac{n(n+1)}{2} - i - 1 \right) + C_3 \left(\frac{n(n+1)}{2} - i - 2 \right) + C_4 \left(\frac{n(n+1)}{2} - i - 2 \right)$$

$$T(n) = C_1 n + C_2 \left(\frac{n^2+n}{2} - i - 1 \right) + C_3 \left(\frac{n^2+n}{2} - i - 2 \right) + C_4 \left(\frac{n^2+n}{2} - i - 2 \right)$$

$$n(C_1 + C_2 + C_3 + C_4) + n^2(C_2 + C_3 + C_4) - i(C_2 + C_3 + C_4) - C_1 - 2(C_3 + C_4)$$

$$T(n) = n + n^2 \rightarrow \boxed{O(n^2)}$$

Complejidad: $O(n \log n)$

```
void ordenaMerge(vector<int> &nums, int Low, int high)
{
    int mid = (Low + high) / 2;
    if (Low < high)
    {
        ordenaMerge(nums, Low, mid);
        ordenaMerge(nums, mid + 1, high);
        merge(nums, Low, mid, high);
    }
}
```

merge sort

$$\text{int mid} = (\text{low} + \text{high}) / 2 \quad C_1 \quad 1$$

if (low < mid)

$$\left\{ \begin{array}{l} \text{merge sort}(\text{nums}, \text{low}, \text{mid}); \quad \sum T_j \left(\frac{n}{2} \right) \quad C_3 \\ \text{merge sort}(\text{nums}, \text{mid} + 1, \text{high}); \quad \sum T_j \left(\frac{n}{2} \right) \quad C_4 \\ \text{merge}(\text{nums}, \text{low}, \text{mid}, \text{high}) \quad n \quad C_5 \end{array} \right.$$
$$T(n) = \begin{cases} 1 & n=1 \\ 2T\left(\frac{n}{2}\right) + n & n>1 \end{cases} \quad T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{n}{2}\right) + n$$
$$2 \left[2T\left(\frac{n}{2^2}\right) + \frac{n}{2} \right] + n \quad \frac{n}{2^k} = 1$$
$$2^2 T\left(\frac{n}{2^2}\right) + 2n \quad n = 2^k$$
$$2^k T\left(\frac{n}{2^k}\right) + kn \quad n = \log_2(n)$$
$$2^{\log_2(n)} T\left(\frac{n}{2^{\log_2(n)}}\right) + n \log_2(n)$$
$$T(n) = n + n \log_2 n$$
$$\boxed{O(n \log n)}$$

Complejidad: $O(n)$

```
int busqSecuencial(vector<int> &nums, int n)
{
    for (int i = 0; i < nums.size(); i++)
    {
        if (n == nums[i])
        {
            // devuelve el indice donde está ubicado el elemento 'n'.
            return i;
        }
    }

    // Devuelvme -1 si el valor 'n' no se encuentra en el vector.
    return -1;
}
```

busc Secuencial

```
for (int i=0 to n)      C1 n
{
    if n == array[i] {   C2 1
        return i; }     C3 1
}
return -1;              C4 1
```

$$T(n) = C_1 n + C_2 + C_3 + C_4$$

$$\boxed{O(n)}$$

Complejidad: $O(\log n)$

```
int busqBinaria(vector<int> &nums, int min, int max, int num)
{
    // Regresa el index de un elemento de un arreglo ordenado. Se busca de manera recursiva.

    if (max >= min)
    {
        int key = (min + max) / 2;

        if (num == nums[key])
        {
            return key;
        }

        else if (num <= nums[key])
        {
            max = key - 1;
            return busqBinaria(nums, min, max, num);
        }

        else if (num >= nums[key])
        {
            min = key + 1;
            return busqBinaria(nums, min, max, num);
        }
    }

    return -1; //no se encontró valor
}
```

busc Binario

```

if (max >= min)           C1 1
    int key = (min + max) / 2 C2 1
    if (num == nums[key]) { C3 1
        return key;        C4 1
    }
    else if (num < nums[key]) C5 1
        max = key - 1;      C6 1
        return buscBinario(n) C7  $\sum T(\frac{n}{2}) + 1$ 
    else if (num > nums[key]) C8 1
        min = key + 1       C9 1
        return buscBinario(n) C10  $\sum T(\frac{n}{2}) + 1$ 

```

Estos están separados.
Solo consideras 1.

$$T(n) = \begin{cases} 1 & n=1 \\ T(\frac{n}{2}) + 2 & n > 1 \end{cases} \quad \begin{matrix} \frac{n}{2^k} = 1 \\ k = \log_2 n \end{matrix}$$

$$T(\frac{n}{2}) + 2 \quad T(\frac{n}{2^k}) + k$$

$$[T(\frac{n}{2^k}) + 2] + 2 \quad T(\frac{n}{2^{\log_2 n}}) + \log_2 n$$

$$T(\frac{n}{2^k}) + k \quad + 1 + \log_2 n$$

$$O(\log_2 n)$$