

# Selection Sort & HeapSort

- Airam Navas Simón
- Ruymán Rodríguez Martín
- Jorge Sierra Acosta

[Repositorio Github](#)

# Selection Sort -- $\Theta(n^2)$

```
selectionSort(A):
```


```
    for i = 0 to A.size
        min = selectMin(A, i, A.size)
        swap(A[min], A[i]);
    return
```

```
selectMin(A, start, end):
```

```
    min = start;
    for i = start to end
        if A[i] < A[min]
            min = i
    return min
```

```
swap(a, b):
```

```
    aux = a
    a = b
    b = aux
    return
```



# Selection Sort -- $\Theta(n^2)$

	Repeticiones
<code>selectionSort(A):</code>	<code>1</code>
<code>for i = 0 to A.size</code>	<code>n</code>
<code>min = selectMin(A, i, A.size)</code>	<code>t<sub>1</sub></code>
<code>swap(A[min], A[i]);</code>	<code>t<sub>2</sub></code>
<code>return</code>	<code>1</code>

```
selectMin(A, start, end):  
    min = start;  
    for i = start to end  
        if A[i] < A[min]  
            min = i  
    return min
```

```
swap(a, b):  
    aux = a  
    a = b  
    b = aux  
    return
```



# Selection Sort -- $\Theta(n^2)$

	Repeticiones
<b>selectionSort(A):</b>	
<b>for</b> $i = 0$ to $A.size$	$n$
$min = selectMin(A, i, A.size)$	$t_1$
$swap(A[min], A[i]);$	$t_2$
<b>return</b>	$1$
<b>selectMin(A, start, end):</b>	$1$
$min = start;$	$1$
<b>for</b> $i = start$ to $end$	$n$
<b>if</b> $A[i] < A[min]$	$1$
$min = i$	$1, 0$
<b>return</b> $min$	$1$
<b>swap(a, b):</b>	
$aux = a$	
$a = b$	
$b = aux$	
<b>return</b>	

# Selection Sort -- $\Theta(n^2)$

	Repeticiones
<b>selectionSort(A):</b>	1
for i = 0 to A.size	n
min = selectMin(A, i, A.size)	$t_1$
swap(A[min], A[i]);	$t_2$
return	1
 <b>selectMin(A, start, end):</b>	1
min = start;	1
for i = start to end	n
if A[i] < A[min]	1
min = i	1, 0
return min	1
 <b>swap(a, b):</b>	1
aux = a	1
a = b	1
b = aux	1
return	1



# Selection Sort -- $\Theta(n^2)$

	Repeticiones	
<code>selectionSort(A):</code>	1	}
<code>for i = 0 to A.size</code>	$n$	
<code>min = selectMin(A, i, A.size)</code>	$t_1$	
<code>swap(A[min], A[i]);</code>	$t_2$	
<code>return</code>	1	$T(n) \in \Theta(n^2)$
<code>selectMin(A, start, end):</code>	1	}
<code>min = start;</code>	1	
<code>for i = start to end</code>	$n$	
<code>if A[i] &lt; A[min]</code>	1	
<code>min = i</code>	$1, 0$	$t_1 = nc_2 + c_1$
<code>return min</code>	1	
<code>swap(a, b):</code>	1	}
<code>aux = a</code>	1	
<code>a = b</code>	1	
<code>b = aux</code>	1	
<code>return</code>	1	
		$t_2 = c_3$



# Selection Sort -- $\Theta(n^2)$

8	5	2	6	9	3	1	4	0	7
---	---	---	---	---	---	---	---	---	---



# HeapSort -- $\Theta(n \log(n))$

```
heapSort(A):  
    for i = (A.size/2) to 1  
        sink(A, i)  
    for i = A.size to 2  
        swap(A[1], A[i])  
        sink(A, i)  
    return
```





# HeapSort -- $\Theta(n \log(n))$

	Repeticiones
<b>heapSort(A):</b>	1
<b>for</b> $i = (A.size/2)$ to 1	$n/2$
<b>sink</b> (A, $i$ )	$t_1$
<b>for</b> $i = A.size$ to 2	$n-1$
<b>swap</b> (A[1], A[ $i$ ])	$t_2$
<b>sink</b> (A, $i$ )	$t_1$
<b>return</b>	1



# HeapSort -- $\Theta(n \log(n))$

	Repeticiones	
heapSort(A):	1	}
for i = (A.size/2) to 1	$n/2$	
sink(A, i)	$t_1$	
for i = A.size to 2	$n-1$	
swap(A[1], A[i])	$t_2$	
sink(A, i)	$t_1$	}
return	1	

$$T(n) = nt_1c_3 + nc_4 + c_5$$



# HeapSort -- $\Theta(n \log(n))$

```
sink(A, i):  
    x = A[i-1]  
    while 2i <= A.size  
        left = 2i  
        right = 2i+1  
        if (left == A.size) or (A[left] > A[right])  
            max = left  
        else  
            max = right  
        if (A[max] <= x)  
            break  
        else  
            swap(A[i], A[max])  
            i = max  
    return
```



# HeapSort -- $\Theta(n \log(n))$

	Repeticiones
<code>sink(A, i):</code>	1
<code>x = A[i-1]</code>	1
<code>while 2i &lt;= A.size</code>	$\log n$
<code>left = 2i</code>	1
<code>right = 2i+1</code>	1
<code>if (left == A.size) or (A[left] &gt; A[right])</code>	1
<code>max = left</code>	1
<code>else</code>	
<code>max = right</code>	1
<code>if (A[max] &lt;= x)</code>	1
<code>break</code>	1
<code>else</code>	
<code>swap(A[i], A[max])</code>	$t_2$
<code>i = max</code>	1
<code>return</code>	

# HeapSort -- $\Theta(n \log(n))$

	Repeticiones
<code>sink(A, i):</code>	1
<code>x = A[i-1]</code>	1
<code>while 2i &lt;= A.size</code>	$\log n$
<code>left = 2i</code>	1
<code>right = 2i+1</code>	1
<code>if (left == A.size) or (A[left] &gt; A[right])</code>	1
<code>max = left</code>	1
<code>else</code>	
<code>max = right</code>	1
<code>if (A[max] &lt;= x)</code>	1
<code>break</code>	1
<code>else</code>	
<code>swap(A[i], A[max])</code>	$t_2$
<code>i = max</code>	1
<code>return</code>	

$$t_1 = \log(n)c_1 + c_2$$



# HeapSort -- $\Theta(n \log(n))$

$$T(n) = nt_1c_3 + nc_4 + c_5$$

$$t_1 = \log(n)c_1 + c_2$$

$$T(n) \in \Theta(n \log(n))$$

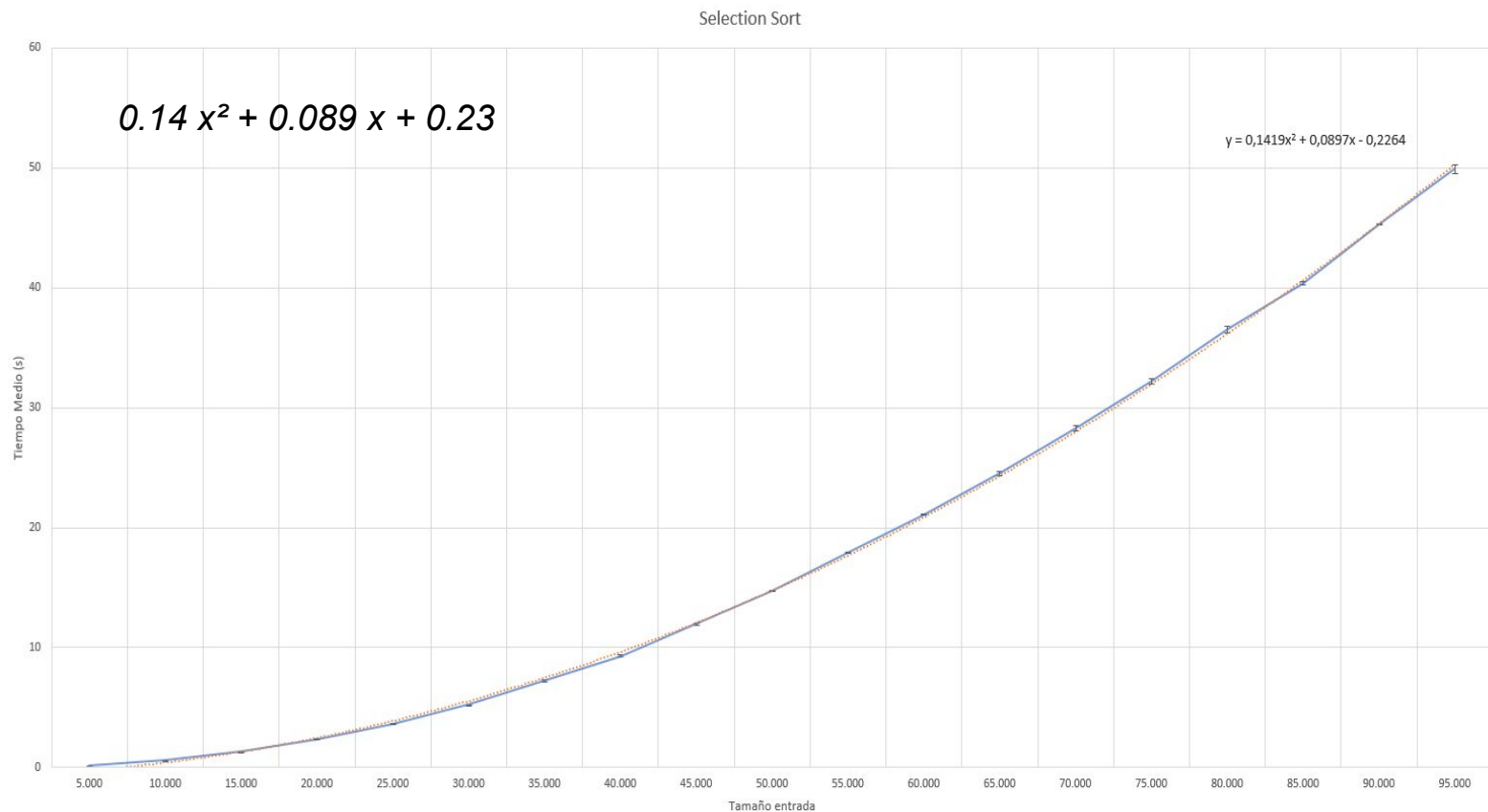


# HeapSort -- $\Theta(n \log(n))$

6 5 3 1 8 7 2 4



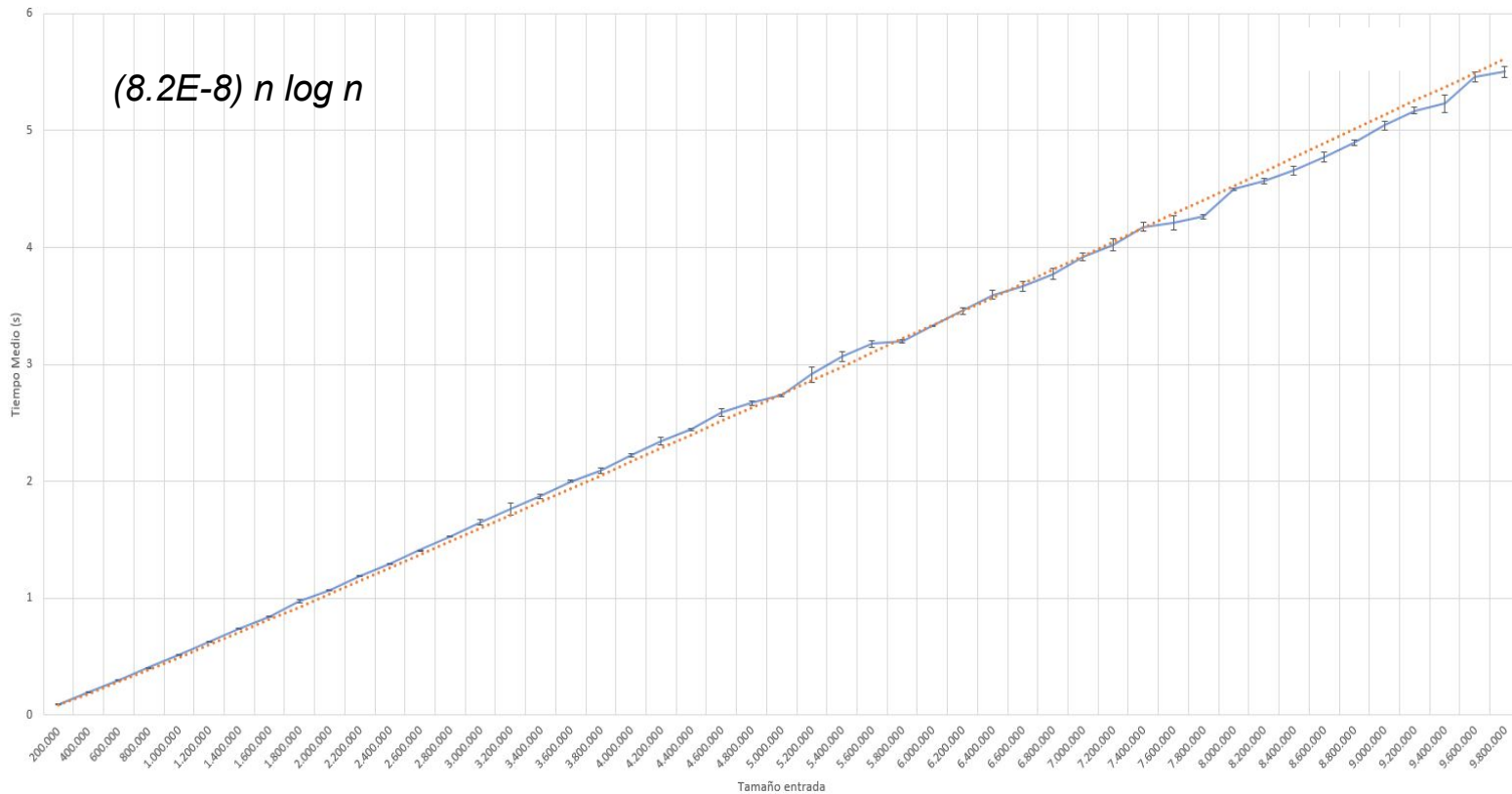
# Caso real -- Selection Sort -- $\Theta(n^2)$



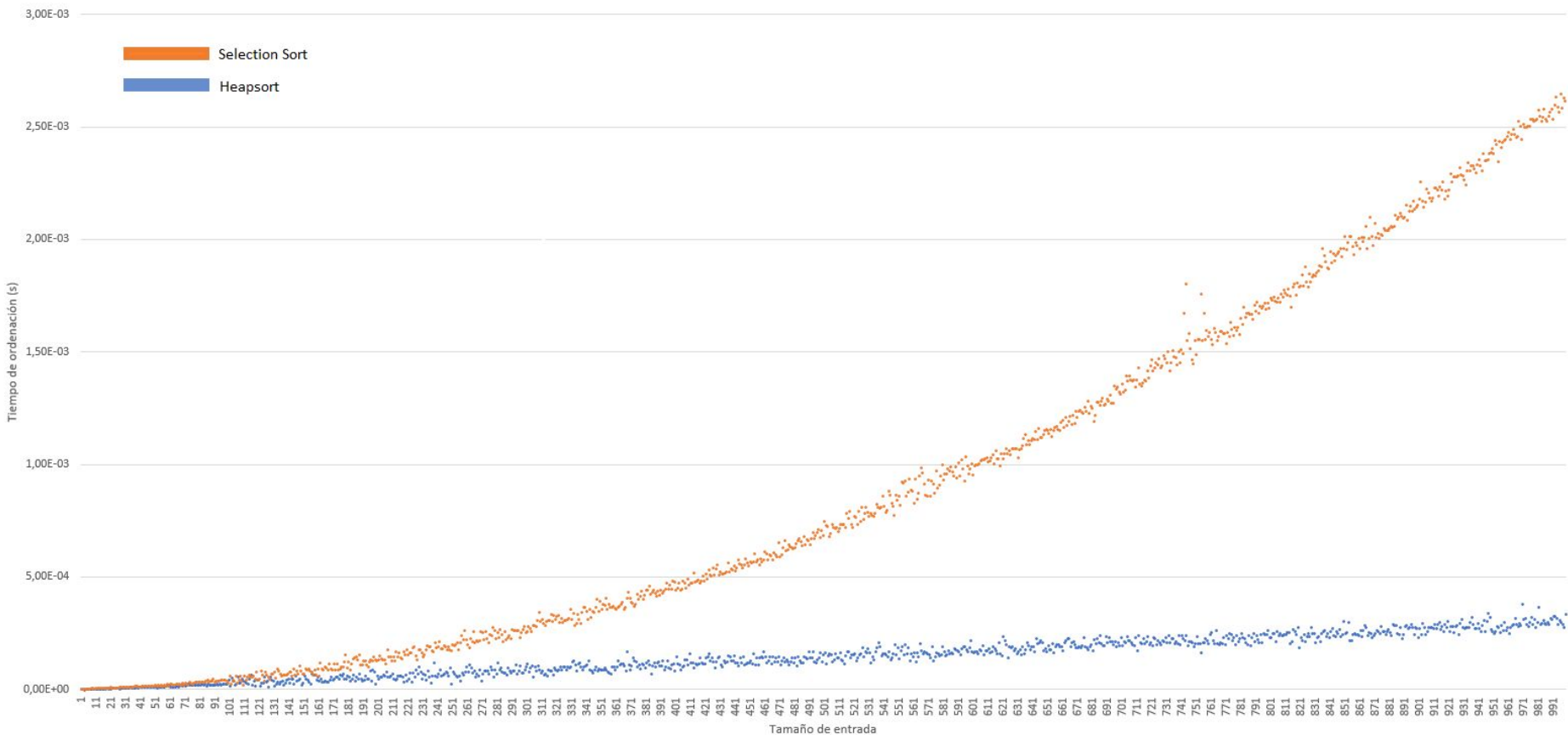


# Caso real -- HeapSort -- $\Theta(n \log(n))$

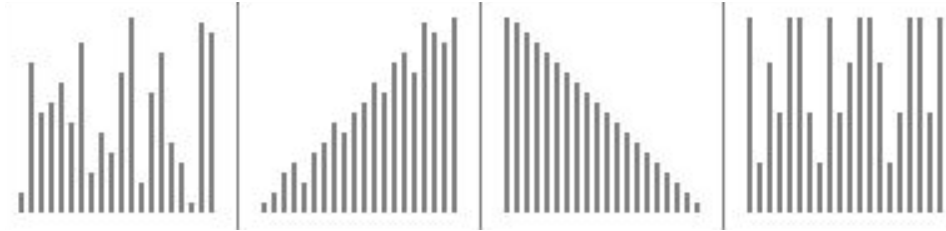
Heap Sort



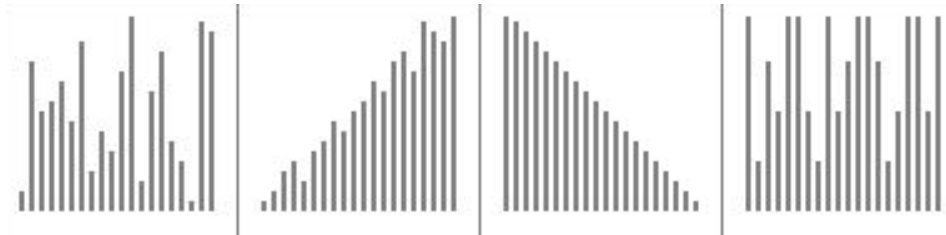
# HeapSort vs Selection Sort



# Comparación



Selection Sort



HeapSort

*Random*

*Nearly sorted*

*Reversed*

*Few Unique*



# Bibliografía

<https://www.toptal.com/developers/sorting-algorithms/>

[https://en.wikipedia.org/wiki/Selection\\_sort](https://en.wikipedia.org/wiki/Selection_sort)

<https://en.wikipedia.org/wiki/Heapsort>

PDF's: Diapositivas Tema 1 DAA



FIN

