

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
selectionSort(A):	
for $i = 0$ to $A.size$	1
$min = selectMin(A, i, A.size)$	$2n$
$swap(A[min], A[i]);$	$n(1 + t_1)$
return	nt_2
	1

```
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
selectionSort(A):	
for $i = 0$ to $A.size$	1
$min = selectMin(A, i, A.size)$	$2n$
$swap(A[min], A[i]);$	$n(1 + t_1)$
return	nt_2
	1
selectMin(A, start, end):	
$min = start;$	1
for $i = start$ to end	1
if $A[i] < A[min]$	$2n$
$min = i$	n
return min	$n, 0$
	1
swap(a, b):	
$aux = a$	
$a = b$	
$b = aux$	
return	

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

	Repeticiones
selectionSort(A):	1
for i = 0 to A.size	2n
min = selectMin(A, i, A.size)	$n(1 + t_1)$
swap(A[min], A[i]);	nt_2
return	1
 selectMin(A, start, end):	1
min = start;	1
for i = start to end	2n
if A[i] < A[min]	n
min = i	n, 0
return min	1
 swap(a, b):	1
aux = a	1
a = b	1
b = aux	1
return	1



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

	Repeticiones	
selectionSort(A):	1	} $T(n) = n^2c_4 + nc_5 + c_6$ $T(n) \in \Theta(n^2)$
for $i = 0$ to $A.size$	$2n$	
$min = selectMin(A, i, A.size)$	$n(1 + t_1)$	
$swap(A[min], A[i]);$	nt_2	
return	1	
 selectMin(A, start, end):	1	} $t_1 = nc_2 + c_1$
$min = start;$	1	
for $i = start$ to end	$2n$	
if $A[i] < A[min]$	n	
$min = i$	$n, 0$	
return min	1	
 swap(a, b):	1	} $t_2 = c_3$
$aux = a$	1	
$a = b$	1	
$b = aux$	1	
return	1	



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
heapSort(A):	1
for $i = (A.size/2)$ to 1	$2(n/2)$
sink (A, i)	nt_1
for $i = A.size$ to 2	$2(n-1)$
swap (A[1], A[i])	nt_2
sink (A, i)	nt_1
return	1



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

	Repeticiones	
heapSort(A):	1	} $T(n) = n(t_1K_3 + K_2) + K_1$
for i = (A.size/2) to 1	$2(n/2)$	
sink(A, i)	nt_1	
for i = A.size to 2	$2(n-1)$	
swap(A[1], A[i])	nt_2	
sink(A, i)	nt_1	
return	1	



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 <= A.size</code>	$2\log n$
<code>left = 2i</code>	$2\log n$
<code>right = 2i+1</code>	$3\log n$
<code>if (left == A.size) or (A[left] > A[right])</code>	$2\log n$
<code>max = left</code>	$\log n$
<code>else</code>	
<code>max = right</code>	0
<code>if (A[max] <= x)</code>	$\log n$
<code>break</code>	1
<code>else</code>	
<code>swap(A[i], A[max])</code>	$t_2 \log n$
<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
while <code>2i <= A.size</code>	$2 \log n$
<code>left = 2i</code>	$2 \log n$
<code>right = 2i+1</code>	$3 \log n$
if (<code>left == A.size</code>) or (<code>A[left] > A[right]</code>)	$2 \log n$
<code>max = left</code>	$\log n$
else	
<code>max = right</code>	0
if (<code>A[max] <= x</code>)	$\log n$
break	1
else	
<code>swap(A[i], A[max])</code>	$t_2 \log n$
<code>i = max</code>	1
return	

$$t_1 = \log(n)K_1 + K_2$$



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

$$T(n) = n (t_1 K_3 + K_2) + K_1$$

$$t_1 = \log(n) K_1 + K_2$$

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

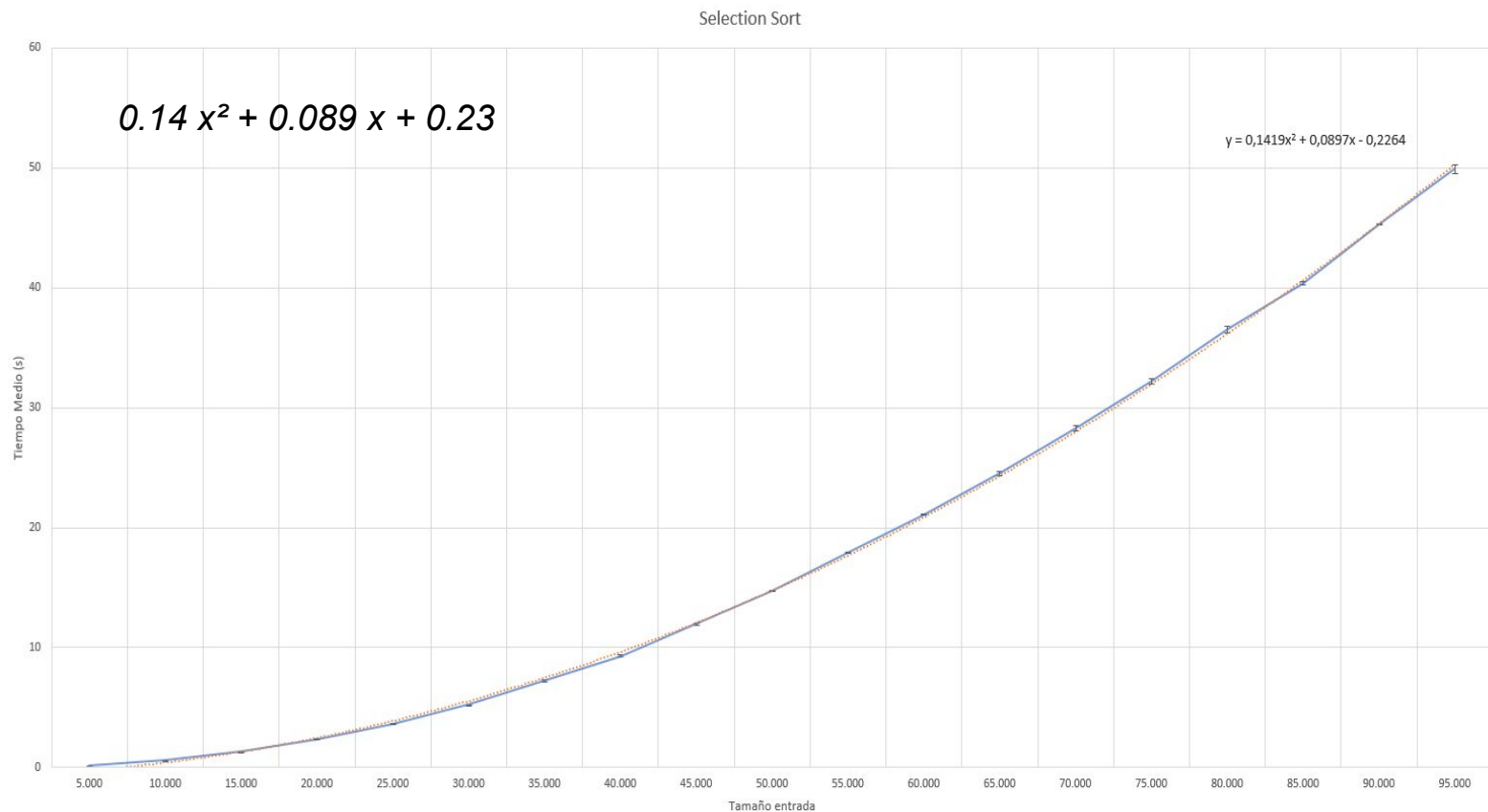


HeapSort -- $\Theta(n \log(n))$

6 5 3 1 8 7 2 4

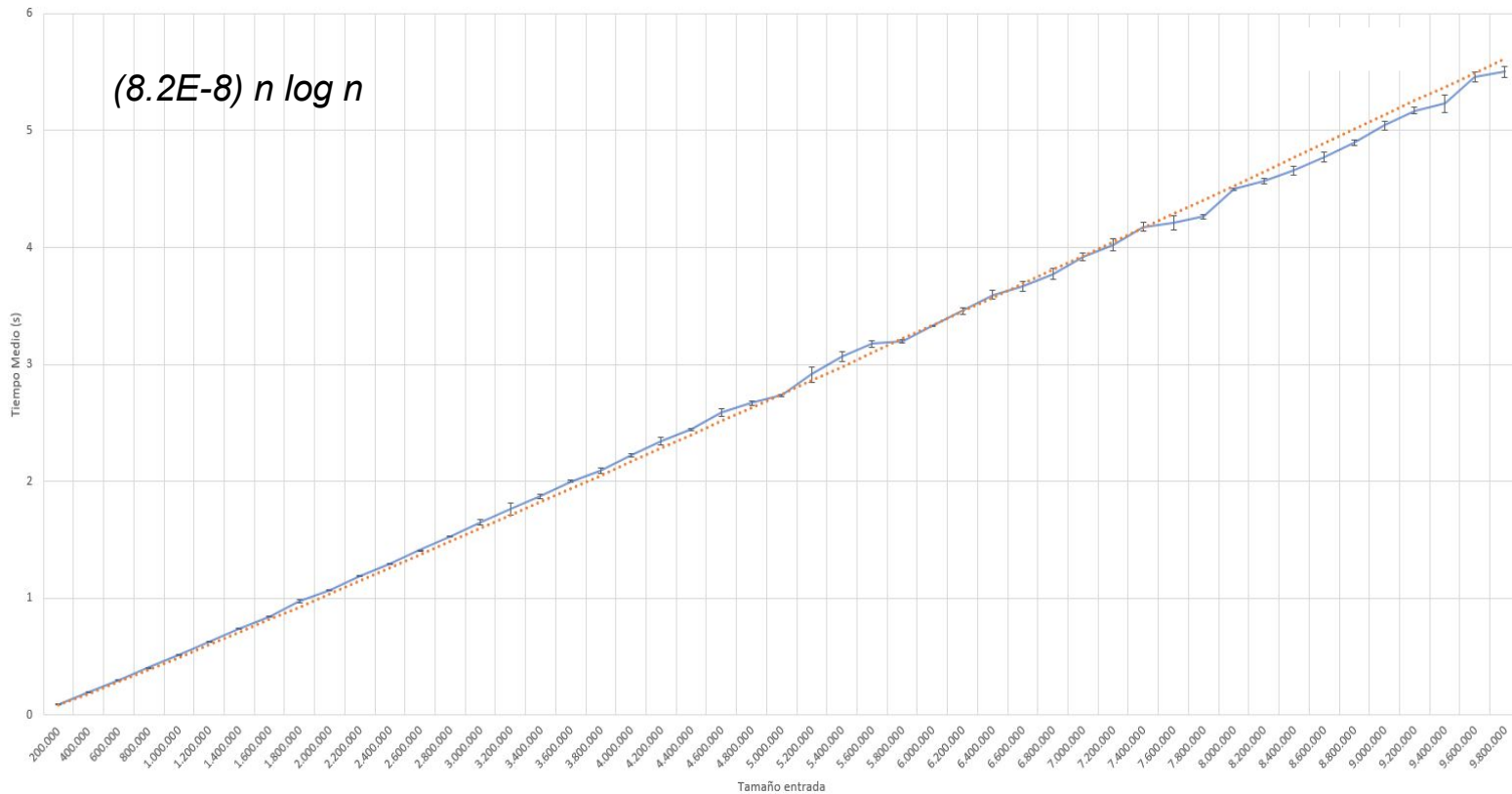


Caso real -- Selection Sort -- $\Theta(n^2)$

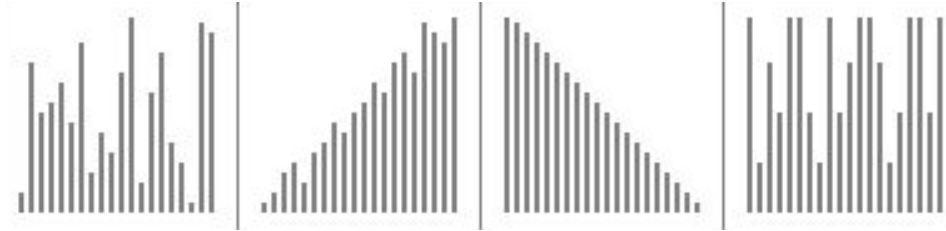


Caso real -- HeapSort -- $\Theta(n \log(n))$

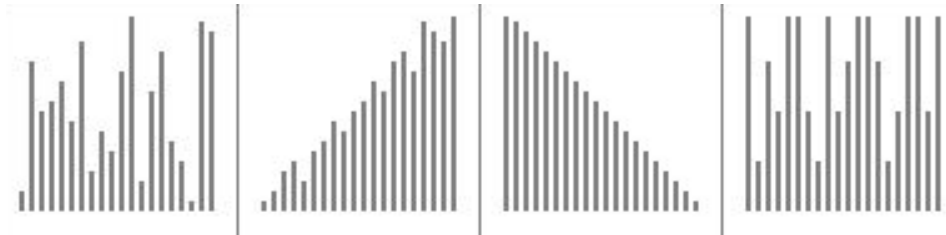
Heap Sort



Comparación



Selection Sort



HeapSort

Random

Nearly sorted

Reversed

Few Unique



FIN

