**Análisis de Complejidad Temporal y Espacial**

***Selection Sort***

Public void sort(int arr[]) {

int n = arr.length;

for (int i = 0; i < n-1; i++) {

int min\_idx = i;

for (int j = i+1; j < n; j++) {

if (arr[j] < arr[min\_idx]) {

min\_idx = j;

}

}

int temp = arr[min\_idx];

arr[min\_idx] = arr[i];

arr[i] = temp;

}

}

***Complejidad Temporal***

|  |  |
| --- | --- |
| ***# Línea*** | ***#Veces que se repite*** |
| 2 | 1 |
| 3 | N |
| 4 | N-1 |
| 5 | ((N(N+1))/2)+N |
| 6 | ((N(N+1))/2) |
| 7 | ((N(N+1))/2) |
| 10 | N-1 |
| 11 | N-1 |
| 12 | N-1 |

Complejidad Temporal O ()

***Complejidad Espacial***

*Insertion Sort*

Public void insertionSort(int arr[]) {

int n = arr.length;

for (int i = 1; i < n; ++i) {

int key = arr[i];

int j = i - 1;

while (j >= 0 && arr[j] > key) {

arr[j + 1] = arr[j];

j = j - 1;

}

arr[j + 1] = key;

}

}

}

***Complejidad Temporal***

|  |  |
| --- | --- |
| ***# Línea*** | ***#Veces que se repite*** |
| 2 | 1 |
| 3 | N |
| 4 | N-1 |
| 5 | N-1 |
| 6 | ((N(N+1))/2)+N |
| 7 | ((N(N+1))/2) |
| 8 | ((N(N+1))/2) |
| 10 | N-1 |

Complejidad Temporal O ()

***Complejidad Espacial***