**Universidad ICESI**

**Sean Quintero - Bryan Guapacha**

**Paola Osorio - John Kennedy**

**Análisis de complejidad temporal de algoritmos de ordenamiento.**

**Insertion sort:**

public static void insertionSortImperative(int[] input) {

for (int i = 1; i < input.length; i++) {

int key = input[i];

int j = i - 1;

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

input[j + 1] = input[j];

j = j - 1;

}

input[j + 1] = key;

}

}

|  |  |
| --- | --- |
| **Instrucción** | **Veces que se repite (Big O)** |
| 1. for (int i = 1; i < input.length; i++) { | n |
| 1. int key = input[i]; | n-1 |
| 1. int j = i - 1; | n-1 |
| 1. while (j >= 0 && input[j] > key) { | (n\*(n-1))/2 |
| 1. input[j + 1] = input[j]; | ((n\*(n-1))/2)-1 |
| 1. j = j - 1; | ((n\*(n-1))/2)-1 |
| 1. input[j + 1] = key; | n-1 |
| **Total:** | n² |

**Merge sort:**

public static void mergeSort(int[] a, int n) {

if (n < 2) {

return;

}

int mid = n / 2;

int[] l = new int[mid];

int[] r = new int[n - mid];

for (int i = 0; i < mid; i++) {

l[i] = a[i];

}

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

r[i - mid] = a[i];

}

*mergeSort*(l, mid);

*mergeSort*(r, n - mid);

*merge*(a, l, r, mid, n - mid);

}

public static void merge(int[] a, int[] l, int[] r, int left, int right) {

int i = 0, j = 0, k = 0;

while (i < left && j < right) {

if (l[i] <= r[j]) {

a[k++] = l[i++];

} else {

a[k++] = r[j++];

}

}

while (i < left) {

a[k++] = l[i++];

}

while (j < right) {

a[k++] = r[j++];

}

}

|  |  |
| --- | --- |
| **Instrucción** | **Veces que se repite (Big O)** |
| 1. if (n < 2) { |  |
| 1. return; |  |
| 1. int mid = n / 2; |  |
| 1. int[] l = new int[mid]; |  |
| 1. int[] r = new int[n - mid]; |  |
| 1. for (int i = 0; i < mid; i++) { |  |
| 1. l[i] = a[i]; |  |
| 1. for (int i = mid; i < n; i++) { |  |
| 1. r[i - mid] = a[i]; |  |
| 1. *mergeSort*(l, mid); |  |
| 1. *mergeSort*(r, n - mid); |  |
| 1. *merge*(a, l, r, mid, n - mid); |  |
| **Total:** |  |

|  |  |
| --- | --- |
| **Instrucción** | **Veces que se repite (Big O)** |
| 1. int i = 0, j = 0, k = 0; |  |
| 1. while (i < left && j < right) { |  |
| 1. if (l[i] <= r[j]) { |  |
| 1. a[k++] = l[i++]; |  |
| 1. while (i < left) { |  |
| 1. a[k++] = l[i++]; |  |
| 1. while (j < right) { |  |
| 1. a[k++] = r[j++]; |  |
| **Total:** |  |