Complejidad Temporal:

Insertion Sort (orderInStands):

**int** n = order.length;

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

**int** intTemp=levels[i];

String temp=games[i];

**int** key = order[i];

**int** j = i - 1;

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

levels[j+1] = levels[j];

games[j+1] = games[j];

order[j + 1] = order[j];

j = j - 1;

}

levels[j+1]=intTemp;

games[j+1]= temp;

order[j + 1] = key;

}

|  |  |
| --- | --- |
| **int** n = order.length; | 1 |
| **for** (**int** i = 1; i < n; ++i) { | n |
| **int** intTemp=levels[i]; | n - 1 |
| String temp=games[i]; | n - 1 |
| **int** key = order[i]; | n - 1 |
| **int** j = i - 1; | n - 1 |
| **while** (j >= 0 && order[j] > key) { | n(n+1)/2 – 1 |
| levels[j+1] = levels[j]; | n(n+1)/2 – 2 |
| games[j+1] = games[j]; | n(n+1)/2 – 2 |
| order[j + 1] = order[j]; | n(n+1)/2 – 2 |
| j = j - 1; } | n(n+1)/2 – 2 |
| levels[j+1]=intTemp; | n - 1 |
| games[j+1]= temp; | n - 1 |
| order[j + 1] = key; } | n - 1 |

Bubble Sort (orderInStands):

n = levels.length;

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

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

**if** (levels[j] > levels[j+1]&& order[j]==order[j+1]){

String stringTemp=games[j];

games[j]=games[j+1];

games[j+1]=stringTemp;

**int** temp = levels[j];

levels[j] = levels[j+1];

levels[j+1] = temp;

}

}

return games;

|  |  |
| --- | --- |
| n = levels.length; | 1 |
| **for** (**int** i = 0; i < n-1; i++) { | n |
| **for** (**int** j = 0; j < n-i-1; j++) { |  |
| **if** (levels[j] > levels[j+1]&& order[j]==order[j+1]){ | n(n+1)/2 -2 |
| String stringTemp=games[j]; | n(n+1)/2 -2 |
| games[j]=games[j+1]; | n(n+1)/2 -2 |
| games[j+1]=stringTemp; | n(n+1)/2 -2 |
| **int** temp = levels[j]; | n(n+1)/2 -2 |
| levels[j] = levels[j+1]; | n(n+1)/2 -2 |
| levels[j+1] = temp; | n(n+1)/2 -2 |
| return games; | 1 |

Bubble Sort (orderClientsByTime):

**for**(**int** c=0;c<clients.length-1;c++) {

**for**(**int** i=c;i <clients.length-c-1;i++) {

**if**(clients[i].getTime()>clients[i+1].getTime()) {

Client temp = clients[i];

clients[i] = clients[i+1];

clients[i+1] = temp;

}

}

}

Si clients.length = n

|  |  |
| --- | --- |
| **for**(**int** c=0;c<clients.length-1;c++) { | n |
| **for**(**int** i=c;i <clients.length-c-1;i++) { | 2n |
| **if**(clients[i].getTime()>clients[i+1].getTime()) { | 2n - 1 |
| Client temp = clients[i]; | 2n - 1 |
| clients[i] = clients[i+1]; | 2n - 1 |
| clients[i+1] = temp; | 2n - 1 |

Complejidad Especial:

|  |  |  |  |
| --- | --- | --- | --- |
| Tipo | Variable | Tamaño de 1 valor atómico | Cantidad de valores atómicos |
| Entrada | games  order  levels | 16 bits  32 bits  32 bits | n (games.length)  n (order.length)  n (levels.length) |
| Auxiliar | n  i  j  intTemp  temp | 32 bits  32 bits  32 bits  32 bits  16 bits | 1  1  1  1  1 |
| Salida | None | None | None |

Complejidad Espacial Total = Entrada+ Auxiliar + Salida =

Complejidad Espacial Auxiliar =

Complejidad Espacial Auxiliar + Salida =

|  |  |  |  |
| --- | --- | --- | --- |
| Tipo | Variable | Tamaño de 1 valor atómico | Cantidad de valores atómicos |
| Entrada | games  order  levels | 16 bits  32 bits  32 bits | n (games.length)  n (order.length)  n (levels.length) |
| Auxiliar | n  i  j  stringTemp  temp | 32 bits  32 bits  32 bits  16 bits  32 bits | 1  1  1  1  1 |
| Salida | games | 16 bits | n (games.length) |

Complejidad Espacial Total = Entrada+ Auxiliar + Salida = 4n + 5 =

Complejidad Espacial Auxiliar =

Complejidad Espacial Auxiliar + Salida =

|  |  |  |  |
| --- | --- | --- | --- |
| Tipo | Variable | Tamaño de 1 valor atómico | Cantidad de valores atómicos |
| Entrada | None | None | None |
| Auxiliar | c  i  temp | 32 bits  32 bits  128 bits | 1  1  1 |
| Salida | None | None | None |

Complejidad Espacial Total = Entrada+ Auxiliar + Salida =

Complejidad Espacial Auxiliar =

Complejidad Espacial Auxiliar + Salida =