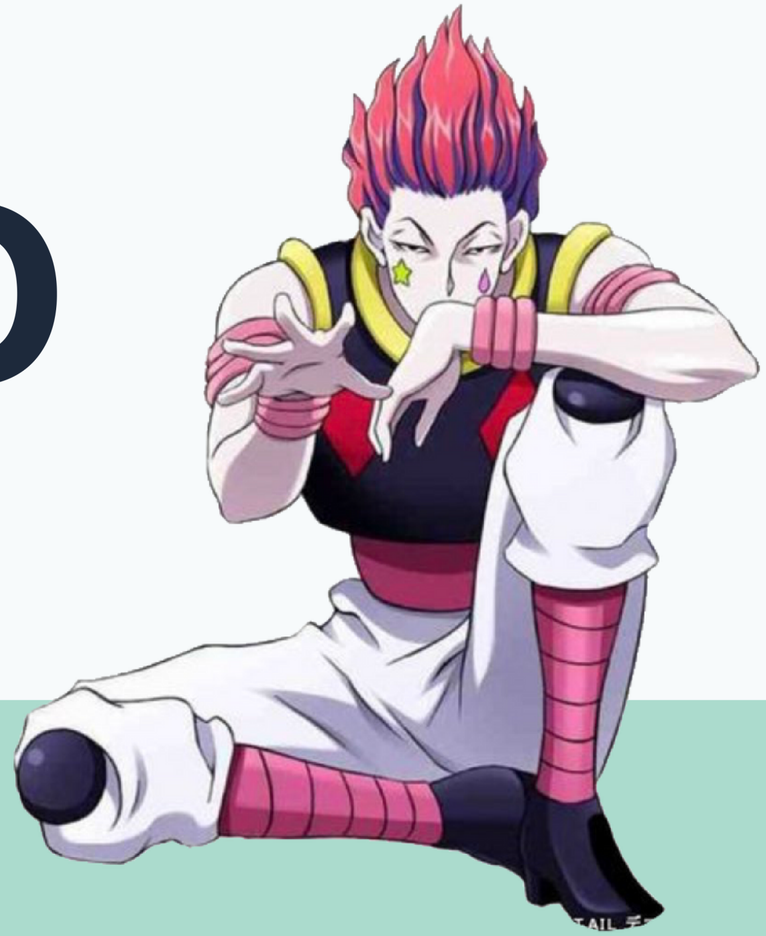
WASM

Grupo 4

DEMO



Principal Algoritmo

JS

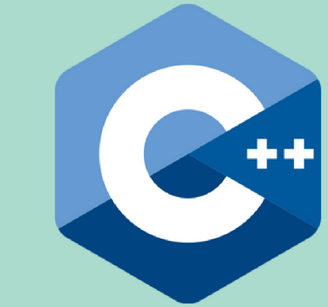
```js
function js_solver(jobs, n_clusters) {
  // sort jobs in descending order
  jobs.sort((a, b) => b - a);

  // initialize clusters with 0 seconds
  let clusters = Array.from({ length: n_clusters }, () => 0);

  // assign each job to the cluster with the least seconds
  for (let j = 0; j < jobs.length; j++) {
    let best_cluster = clusters.indexOf(Math.min(...clusters));
    clusters[best_cluster] += jobs[j];
  }

  return Math.max(...clusters);
}
```

Cuál Sort elegir?



```
extern "C" int cpp_solver(int jobs[], int n_jobs, int n_clusters) {
  // sort jobs in descending order
    ??????????????????

  int clusters[n_clusters];
  // initialize all clusters to 0
  for (int c = 0; c < n_clusters; c++) {
      clusters[c] = 0;
  }
  // assign each job to the cluster with the least seconds
  for (int j = 0; j < n_jobs; j++) {
      int best_cluster = index_of_min(clusters, n_clusters);
      clusters[best_cluster] += jobs[j];
  }
  // look for the cluster with the most seconds
  int total_seconds = clusters[index_of_max(clusters, n_clusters)];
  return total_seconds;
}
```

Principal Algoritmo

```
function js_solver(jobs, n_clusters) {
  // sort jobs in descending order
  jobs.sort((a, b) => b - a);

  // initialize clusters with 0 seconds
  let clusters = Array.from({ length: n_clusters }, () => 0);

  // assign each job to the cluster with the least seconds
  for (let j = 0; j < jobs.length; j++) {
    let best_cluster = clusters.indexOf(Math.min(...clusters));
    clusters[best_cluster] += jobs[j];
  }

  return Math.max(...clusters);
}
```

??

Sorting for num arrays

Browser	Sort used
Chrome	Merge+Insertion
Safari	QuickSort
Firefox	MergeSort
M. Edge	Quick+Heap+Insertion

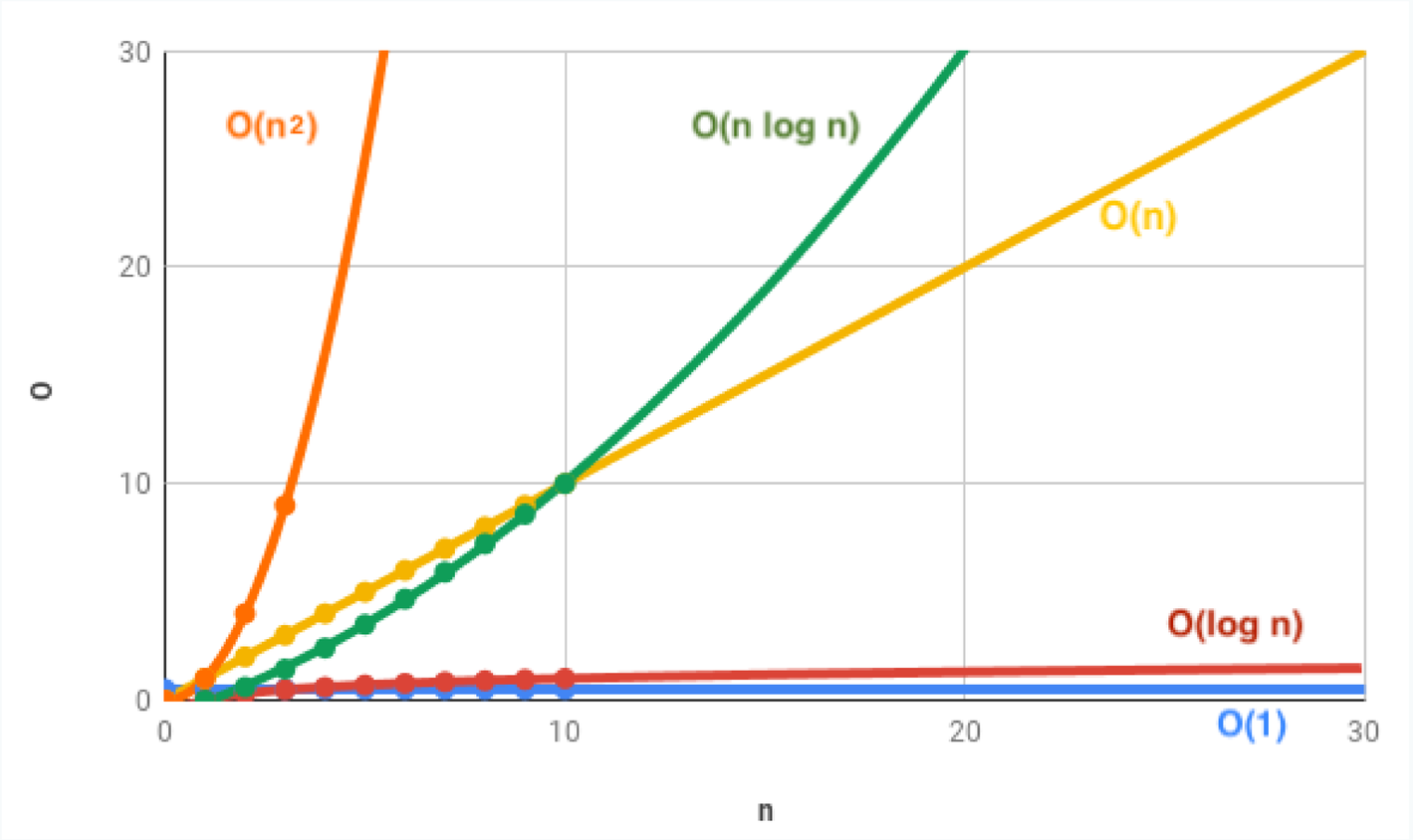
Fuentes: https://v8.dev/blog/array-sort
https://learn.microsoft.com/dotnet/api/system.array.sort
https://stackabuse.com/sorting-arrays-in-javascript

Complejidad de un Algoritmo

Notación O()

Sea **f** y **g** funciones

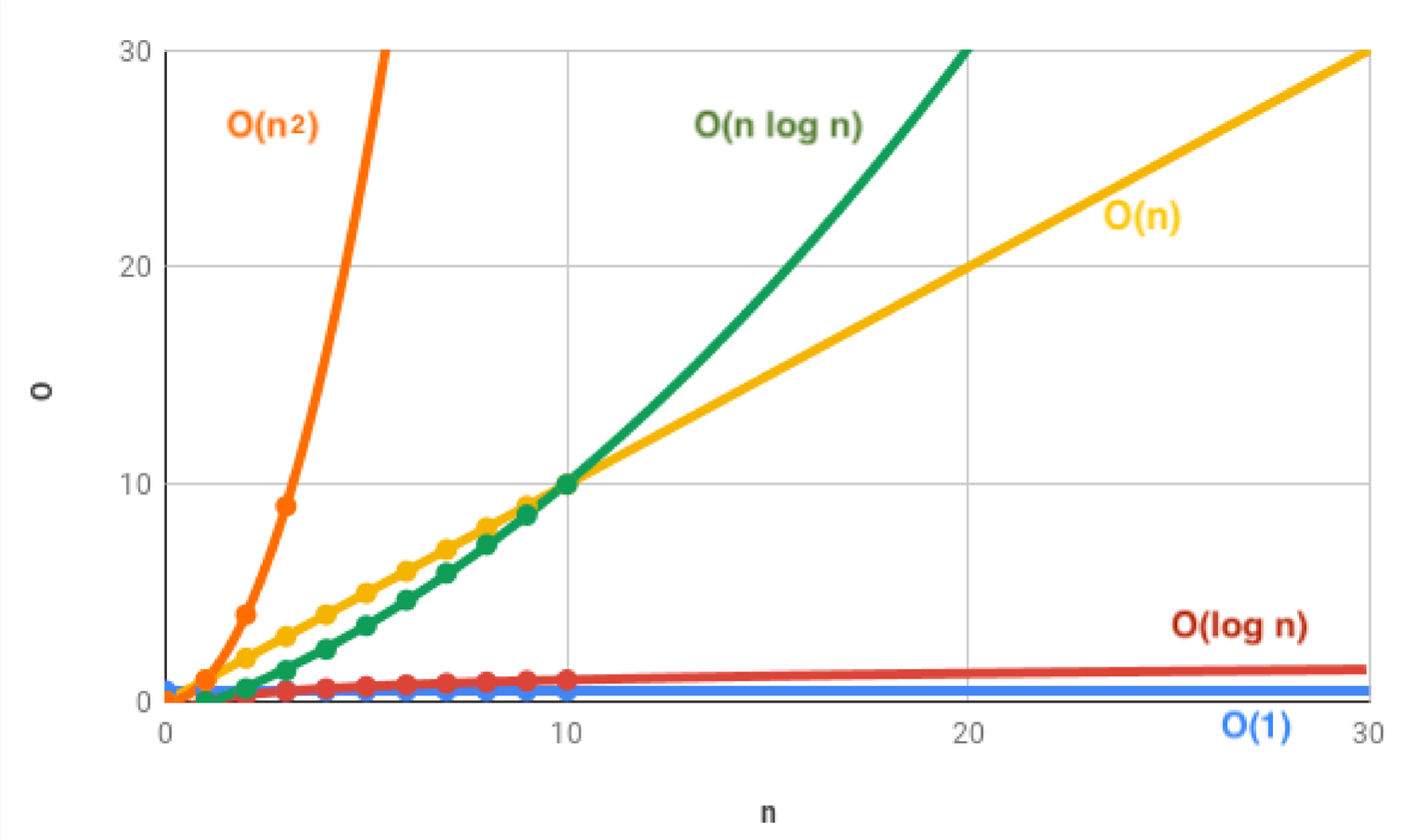
Si **f(n)** es **O(g(n))**, entonces **f** crece más lento o igual que **g**



Fuente: freeCodeCamp

Complejidad de un Algoritmo

Mide la eficiencia de un algoritmo en relación al **tiempo** o **espacio** que demanda su ejecución, teniendo en cuenta el tamaño del input (n).



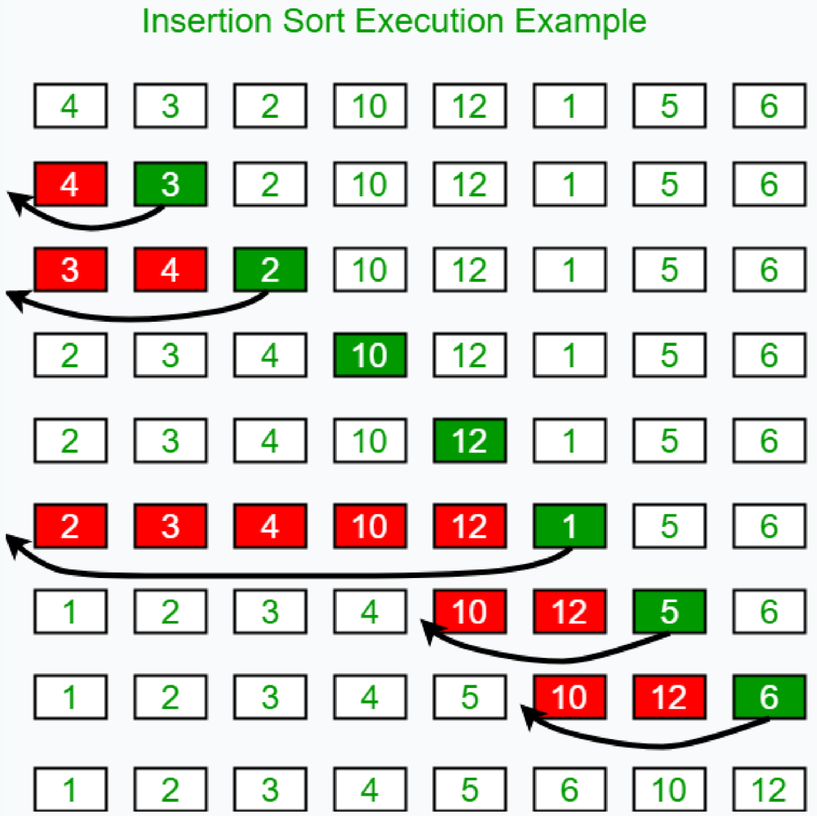
Fuente: freeCodeCamp

Algunos Sort Populares

	Avg Case
Selection Sort	$O(n^2)$
Insertion Sort	$O(n^2)$
Merge Sort	$O(n \cdot \log(n))$
Quick Sort	$O(n \cdot \log(n))$

Insertion Sort

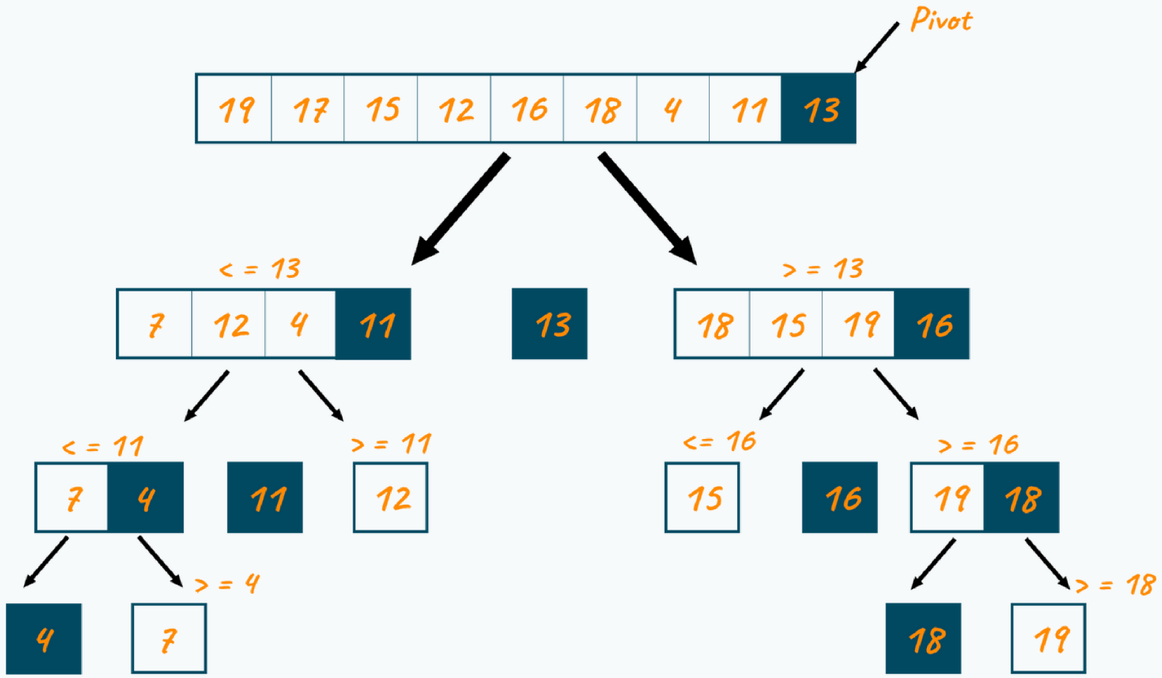
Compara el elemento con el anterior, si este es menor, compara nuevamente con el anterior y así hasta ponerlo en la posición correcta.



Fuente: Geeksforgeeks

QuickSort

Se elige un pivote, luego los elementos que son menores a este valor quedan a la izquierda y los mayores a la derecha, y se repite el proceso hasta que la lista quede ordenada.



Fuente: Favtutor

Diferencia entre Sorts

Random Jobs Generator

N° jobs 100000

Max time 10000

N° Clusters 2

Diferencia entre Sorts

Rand Sorted

InsertionSort

8.028 s

Browser Func.

0.147 s

InsertionSort

3.457 s

QuickSort

0.150 s

Diferencia entre Sorts

Desc Sorted

JS

InsertionSort

0.047 s

JS

Browser Func.

0.062 s

C++

InsertionSort

0.137 s

C++

QuickSort

9.754 s

Diferencia entre Sorts

Asc Sorted

JS

InsertionSort

17.691 s

JS

Browser Func.

0.097 s

InsertionSort

6.224 s

QuickSort

2.491 s

WASM

Grupo 4