

Implementing Sorting Algorithms

data array

- A 2-D array with a size of (n_sort, size)
- Each 1-D array with a size of size should be same

The diagram illustrates a 2-D array structure. A blue bracket on the left, labeled 'n_sort', spans four rows. A blue bracket on top, labeled 'size', spans five columns. The array contains the following values:

34	80	15	39	...
34	80	15	39	...
34	80	15	39	...
34	80	15	39	...

An Example Output

```
A part of the initial data:  
  18914 10315 18164 23804 9675 18397 21656 23618 26534 6789  
  
quick sort  
Time : 0.002 sec  
  
merge sort  
Time : 0.004 sec  
same  
  
insertion sort  
Time : 0.005 sec  
same  
  
stooge sort  
Time : 24.366 sec  
same
```

Useful Libraries

```
#include <ctime> // time  
#include <cstdlib> // srand, rand  
#include <climits> // INT_MAX
```

$x = \text{INT_MAX};$

$x \leftarrow \infty$

Excluding an Algorithm

```
run_quick_sort(size, data[0]);

run_merge_sort(size, data[1]);
compare_results(size, data[1], data[0]); //

run_insertion_sort(size, data[2]);
compare_results(size, data[2], data[0]);

// run_stooge_sort(size, data[3]);
// compare_results(size, data[3], data[0]);
```