B: Mr Evil??

D: SelectionSort

Unstable Sorts: B,D,E

Sorter A:

Cost for sorted and inverse sorted is the same

Sorter C:

Cost for sorted array is way cheaper than inverse sorted array (possibly bubble sort and insertion sort)

Comparing Sorter C and Sorter F, when given the same array, it took Sorter C longer which could mean that sorter C is **bubble sort.**

Sorter D:

Cost for sorted and inverse sorted is the same.

By comparing with a sorter E,

Sorter E:

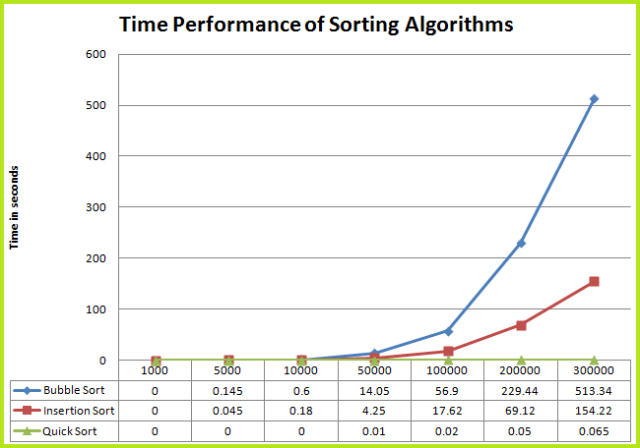
Cost for sorted and inverse sorted is the same

Sorter F:

Cost for sorted array is way cheaper than inverse sorted array (possibly bubble sort and insertion sort)

Comparing Sorter C and Sorter F, when given the same array, it took Sorter C longer which could mean that sorter F is **insertion sort.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sort Name | Best Case | Worst case | Stable | Possible one |
| Bubble Sort | O(n)  When already sorted or nearly sorted | O(n2)  Inverse sorted | Yes | Sorter C |
| Selection Sort | O(n2) | O(n2) | No | Sorter D |
| Insertion Sort | O(n)  When already sorted or nearly sorted | O(n2)  Inverse sorted | Yes | Sorter F |
| Merge Sort | O(n log n) | O(n log n) | Yes | Sorter A |
| Quick Sort |  |  | No | Sorter E |
| Mr Evil |  |  | Probably No | Sorter B |

Bubble sort takes more time for larger inputs

Bubble sort has “greater steeps”