Result in COE Linux:

```
-bash-4.2$ ./a.out
Insertion Sort:
Best
moves: 0 comps: 999
Sorted
Average
moves: 260634 comps: 261628
Sorted
Worst
moves: 499500 comps: 499500
Sorted

Heap Sort:
Best
moves: 29127 comps: 20418
Sorted
Average
moves: 27162 comps: 19108
Sorted
Worst
moves: 24951 comps: 17634
Sorted

Quick Sort:
Best
moves: 1501497 comps: 499500
Sorted
Average
moves: 17610 comps: 10257
Sorted
Worst
moves: 751497 comps: 499500
Sorted
```

Result in sort.txt

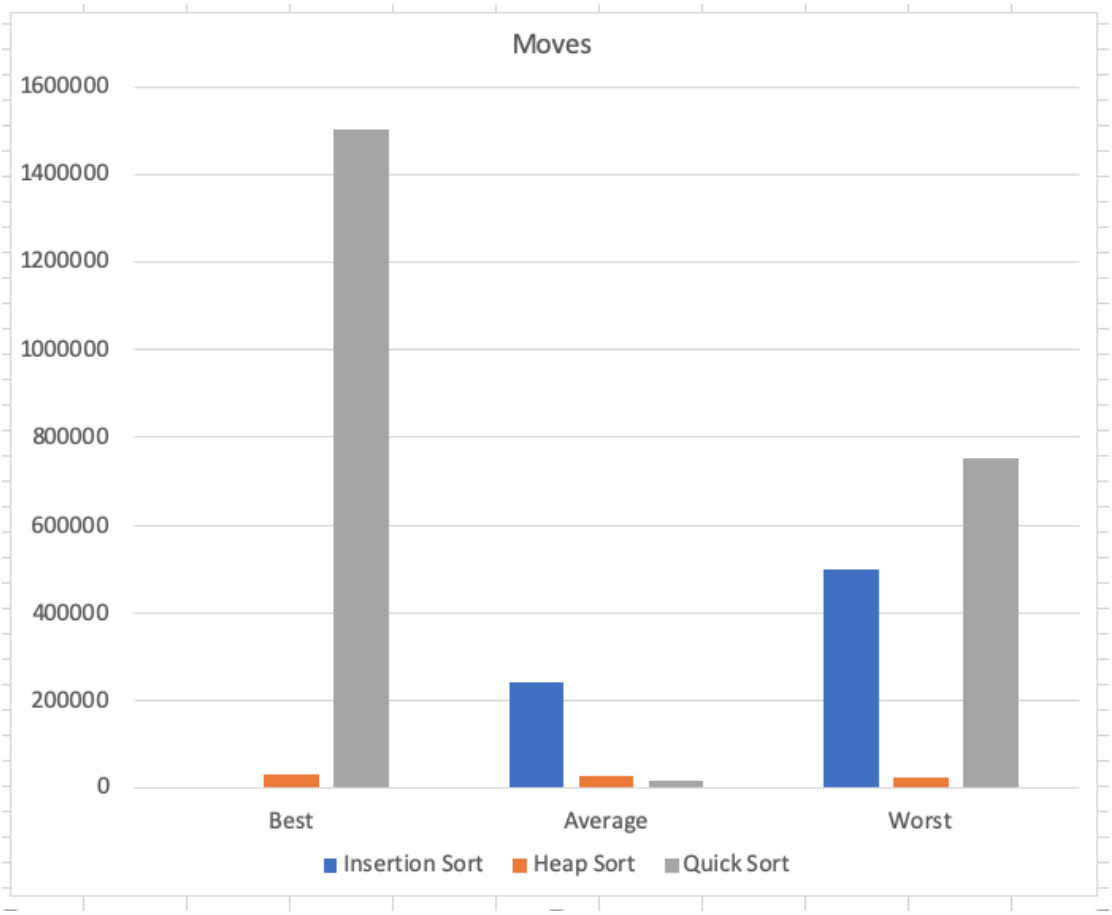
```
Insertion Sort:
Best
moves: 0 comps: 999
Average
moves: 260634 comps: 261628
Worst
moves: 499500 comps: 499500

Heap Sort:
Best
moves: 29127 comps: 20418
Average
moves: 27162 comps: 19108
Worst
moves: 24951 comps: 17634

Quick Sort:
Best
moves: 1501497 comps: 499500
Average
moves: 17610 comps: 10257
Worst
moves: 751497 comps: 499500

~
~
~
~
~
~
~
~
~
"sort.txt" 24L, 345C                                    1,1           全部
```

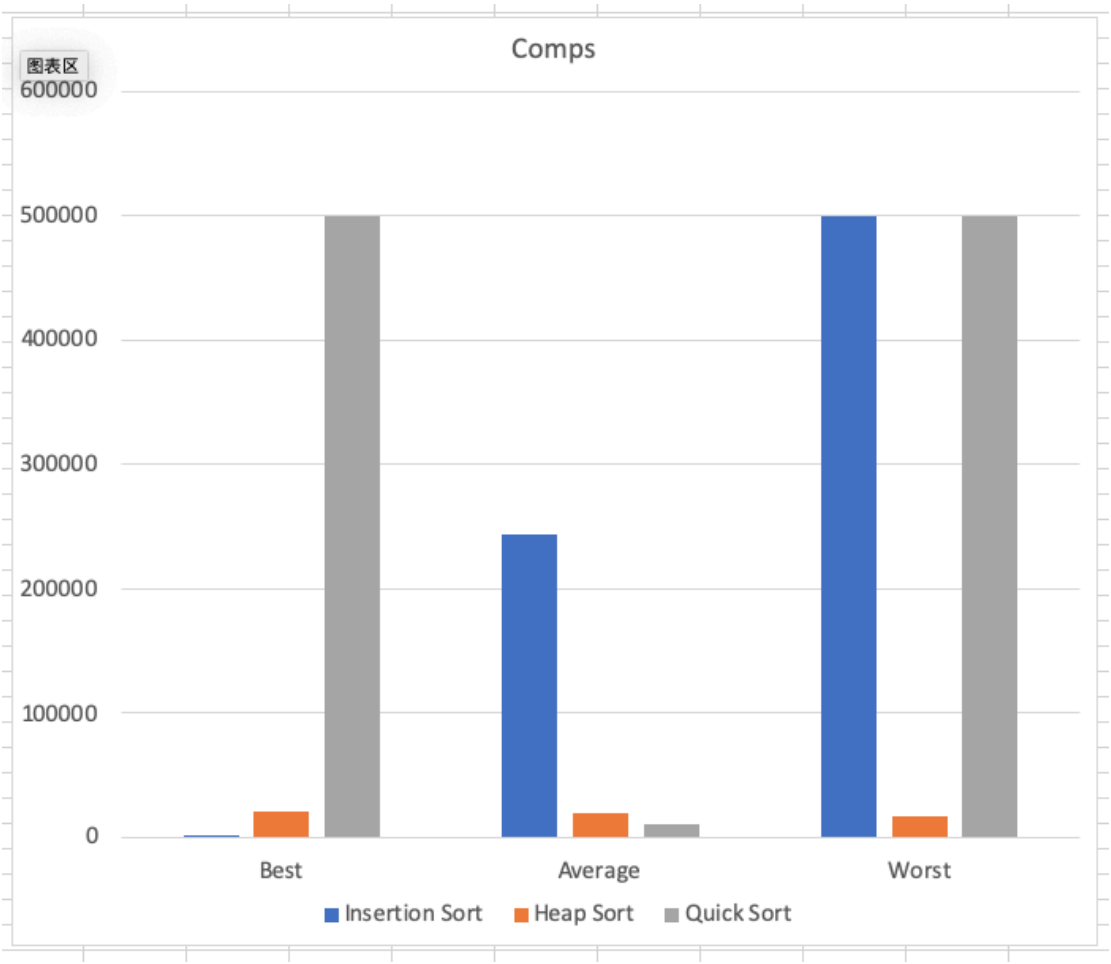
Answer:

i. Because Insertion Sort moves small integer to the front. And when the array is already been sorted, no integer needs to be moved. So moves = 0.

ii. In insertion sort, there are 999 circles to move integers to right situation. When the array is sorted, each circle ends with one comparison. So the comparison times in best situation is 999.

iii. Here are the excel graphs.





In best case, insertion sort performs best and quick sort performs worst in both moves and comparisons.

In average case, quick sort performs best and heap sort performs well, and insertion sort performs very bad in both moves and comparisons.

In worst case, heap sort performs best. Quick sort and insertion sort performs bad, especially in comparison times.

In most of times, the arrays we get are always average or worst case, so heap sort and quick sort should be chosen, and we could choose suitable algorism according to the characteristic of array.