**SORTING TABLE FOR 100 ELEMENTS**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Inverse** | | **Random** | | **Almost sorted** | |
| **Algorithm** | **Comparisons** | **Swaps** | **Comparisons** | **Swaps** | **Comparisons** | **Swaps** |
| Selection | 4950 | 59 | 4950 | 95 | 4950 | 31 |
| Insertion | 4901 | 99 | 2496 | 99 | 118 | 99 |
| Heap | 1077 | 515 | 1247 | 593 | 1310 | 626 |
| Merge | 413 | 1360 | 552 | 1360 | 393 | 1360 |
| Quick | 244 | 384 | 370 | 500 | 219 | 351 |
| Shell | 250 | 753 | 417 | 920 | 58 | 561 |

**Conclusion:**

Based on the number of comparison (4950), selection sort perform almost the same regardless of the data arrangement (in reverse order, random of almost sorted). However selection sort has the property to minimize the number of swaps.

Insertion sort perform better if the data is almost sorted. Heap sort perform almost the same regardless of the data arrangement with slightly less comparison if the data is in reverse order. Merge sort and quick sort are well stable sorts. They perform well with less comparisons but more swaps. Shell sort perform well for a list nearly sorted with more swaps than comparisons.

**SORTING TABLE FOR 1000 ELEMENTS**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Inverse** | | **Random** | | **Almost sorted** | |
| **Algorithm** | **Comparisons** | **Swaps** | **Comparisons** | **Swaps** | **Comparisons** | **Swaps** |
| Selection | 499500 | 603 | 499500 | 988 | 499500 | 638 |
| Insertion | 498988 | 999 | 250054 | 999 | 12171 | 999 |
| Heap | 18788 | 8336 | 20561 | 9069 | 21268 | 9435 |
| Merge | 4938 | 19974 | 8710 | 19974 | 6547 | 19974 |
| Quick | 4266 | 5664 | 4671 | 6025 | 3715 | 5037 |
| Shell | 4469 | 12475 | 7217 | 15223 | 1734 | 9740 |

**Conclusion:**

With a list of thousands elements, selection sort performs the worst regardless of the data arrangement. Insertion sort perform a best sort if the data is almost sorted. Heap sort perform almost the same regardless of the data arrangement with slightly less comparison if the data is in reverse order. For a size of thousands elements, merge sort give a best result if the data is in reverse order. Quick sort is a well stable sort and performs the same. Shell sort perform well for a list nearly sorted with more swaps than comparisons.