|  |  |  |  |
| --- | --- | --- | --- |
| Insertion Sort | Minimum Time | Average Time | Maximum Time |
| Sorting 1000 integers | 76100 | 131496,40 | 7583600 |
| Sorting 10000 integers | 6565600 | 7824639,20 | 28879500 |

test array

[6, 4, 7, 5, 2, 1, 70, 84, 93, 107]

[1, 2, 4, 5, 6, 7, 70, 84, 93, 107]

random array

[0, 24, 42, 59, 17, 28, 9, 94, 67, 12]

[0, 9, 12, 17, 24, 28, 42, 59, 67, 94]

* ***86,2759526938239*** is fold difference between the minimum time. This shows us when the input size increases, running time will be longer.

This part was for insertion.

--------------------------------------------------------------------------------------------------------------------------------------

Test array:

[6, 4, 7, 5, 2, 1, 70, 84, 93, 107]

[1, 2, 4, 5, 6, 7, 70, 84, 93, 107]

random array:

[62, 96, 16, 28, 36, 15, 73, 65, 76, 92]

[15, 16, 28, 36, 62, 65, 73, 76, 92, 96]

|  |  |  |  |
| --- | --- | --- | --- |
| Selection Sort | Minimum Time | Average Time | Maximum Time |
| Sorting 1000 integers | 281100 | 381079,30 | 2831100 |
| Sorting 10000 integers | 23343100 | 46477382,40 | 160598000 |

* ***83 was*** fold difference between the minimum time. This shows us when the input size increases, running time will be longer.

--------------------------------------------------------------------------------------------------------------------------------------

Compare the average running times of insertion sort and selection sort. Which algorithm performs better? Can you give any reason?

* İnsertion sort is better. The reason is selection gets through whole array unlike insertion only compares the array if elements are the same, there is no step.