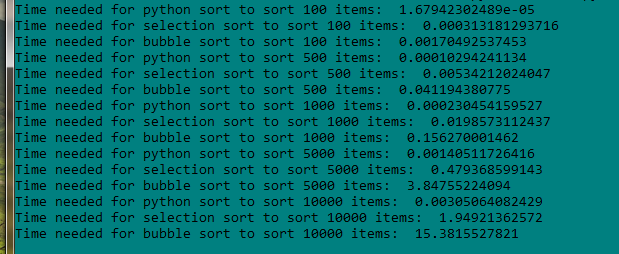
Sample Execution:



Analysis & Conclusion:

1.

2.

|  |  |  |
| --- | --- | --- |
| Method | # of elements | Execution Time |
| Python sort | 100 | 1.6794 e -5 |
| Selection sort | 100 | 0.00031 |
| Bubble sort | 100 | 0.0017 |
| Python sort | 500 | 0.00010 |
| Selection sort | 500 | 0.00534 |
| Bubble sort | 500 | 0.04119 |
| Python sort | 1000 | 0.00023 |
| Selection sort | 1000 | 0.0198 |
| Bubble sort | 1000 | 0.15627 |
| Python sort | 5000 | 0.001405 |
| Selection sort | 5000 | 0.4793 |
| Bubble sort | 5000 | 3.8475 |
| Python sort | 10000 | 0.00305 |
| Selection sort | 10000 | 1.9492 |
| Bubble sort | 10000 | 15.381 |

3. Both bubble sort and selection sort performs n operations in outermost loop.

4.

procedure bubbleSort( A : list of sortable items )

repeat

swapped = false

for i = 1 to length(A) - 1 inclusive do:

*/\* if this pair is out of order \*/*

if A[i-1] > A[i] then

*/\* swap them and remember something changed \*/*

swap( A[i-1], A[i] )

swapped = true

end if

end for

until not swapped

end procedure

*/\* a[0] to a[n-1] is the array to sort \*/*

int i,j;

int iMin;

*/\* advance the position through the entire array \*/*

*/\* (could do j < n-1 because single element is also min element) \*/*

for (j = 0; j < n-1; j++) {

*/\* find the min element in the unsorted a[j .. n-1] \*/*

*/\* assume the min is the first element \*/*

iMin = j;

*/\* test against elements after j to find the smallest \*/*

for ( i = j+1; i < n; i++) {

*/\* if this element is less, then it is the new minimum \*/*

if (a[i] < a[iMin]) {

*/\* found new minimum; remember its index \*/*

iMin = i;

}

}

*/\* iMin is the index of the minimum element. Swap it with the current position \*/*

if ( iMin != j ) {

swap(a[j], a[iMin]);

}

}

The outermost loop of bubble sort performs n operations. In worst case scenario, it performs at *O(n2)* complexity. Selection sort scans n elements to find lowest element. Finding next lowest element requires scanning of n-1 elements. So it’s complexity is ∈ Θ(*n*2). That’s why selection sort performs better than bubble sort in most cases.

5.

|  |  |  |
| --- | --- | --- |
| Method | Time Complexity | Rank |
| Python sort | O(n log n) | 1 |
| Selection sort | Θ(n2) | 2 |
| Bubble sort | O(n2) | 3 |