Algorithm SelectionSort(arr): n = length of arr // 1 for i from 0 to n-1: // n-1 minIndex = i // 1 for j from i+1 to n: // n - 1 if arr[j] < arr[minIndex]: // 1 minIndex = j // 1 swap(arr[minIndex], arr[i]) // 1

The outer loop runs for n-1 iterations, where n is the number of elements in the array. The inner loop runs for n-i-1 iterations in each iteration of the outer loop.

Worst Case: The worst-case scenario occurs when the array is in reverse order. In each pass of the outer loop, the algorithm finds the minimum element in the unsorted part of the array and performs a swap. This results in a time complexity of O(n^2).

Best Case: The best-case scenario occurs when the array is already sorted. In each pass of the outer loop, the algorithm finds the minimum element but doesn't need to perform a swap because the minimum element is already in the correct position. However, the algorithm still needs to check for the minimum element in each pass, resulting in a time complexity of Ω(n^2).

Average Case: The average-case time complexity is also O(n^2). Although the actual number of comparisons and swaps may be lower than the worst case, the overall growth is still quadratic.

Final Result: Worst Case: O(n^2) Best Case: Ω(n^2) Average Case: Θ(n^2)