Algorithm Analysis

2 min

Given a moderately unsorted data-set, bubble sort requires multiple passes through the input before producing a sorted list. Each pass through the list will place the next largest value in its proper place.

We are performing n-1 comparisons for our inner loop. Then, we must go through the list n times in order to ensure that each item in our list has been placed in its proper order.

The n signifies the number of elements in the list. In a worst case scenario, the inner loop does n-1 comparisons for each n element in the list.

Therefore we calculate the algorithm's efficiency as:

O(n(n−1))=O(n(n))=O(n2)$O(n(n-1))=O(n(n))=O(n2)$

The diagram analyzes the

Preview: Docs Loading link description

pseudocode

implementation of bubble sort to show how we draw this conclusion.

When calculating the run-time efficiency of an

Preview: Docs Loading link description

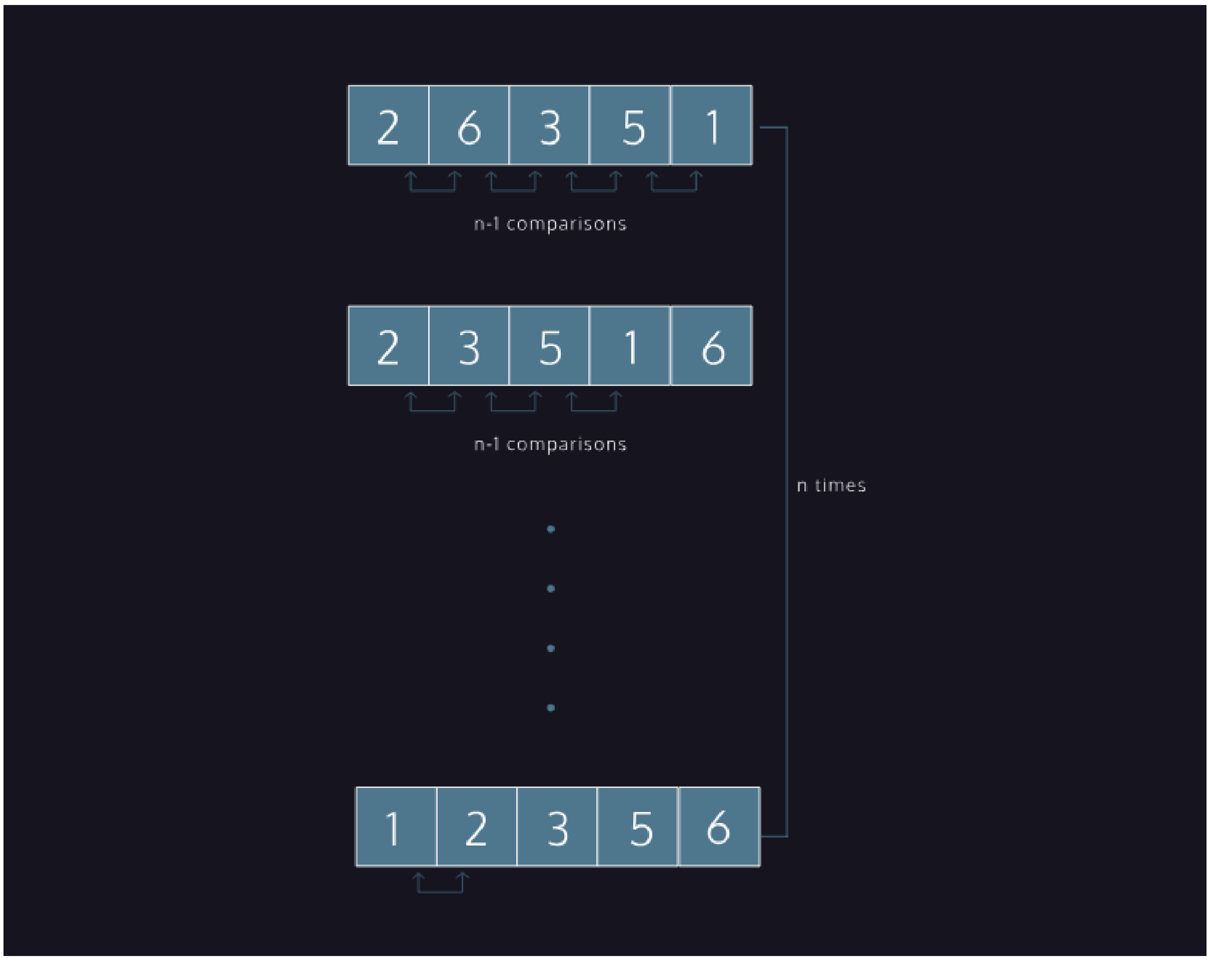
algorithm

, we drop the constant (-1), which simplifies our inner loop comparisons to n.

This is how we arrive at the algorithm's runtime: O(n^2).

Instructions

What input to bubble sort would produce the worst possible runtime?


2 6 3 5 1
n-1 comparisons
2 3 5 1 6
n-1 comparisons
n times
1 2 3 5 6