Understanding Bubble Sort

Introduction:

Bubble Sort is a simple sorting algorithm that works by repeatedly stepping through the list, comparing adjacent elements, and swapping them if they are in the wrong order. The pass through the list is repeated until the list is sorted. The algorithm is named for the way smaller elements 'bubble' to the top of the list.

Bubble Sort has a time complexity of O(n^2) due to its nested loops, which makes it inefficient for large datasets. However, it is easy to understand and implement, making it useful for educational purposes.

How Bubble Sort Works:

1. Starting with the first element, compare it with the next element.
2. If the current element is greater than the next element, swap them.
3. Move to the next element and repeat the process for the entire list.
4. After one complete pass through the list, the largest element is at its correct position.
5. Repeat steps 1 to 4 for the remaining elements, excluding the last sorted elements in each pass.

Python Code Example:

```
def bubble_sort(arr):
    n = len(arr)
    for i in range(n):
        for j in range(0, n-i-1):
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]
```

```
# Example usage:
arr = [64, 34, 25, 12, 22, 11, 90]
bubble_sort(arr)
print('Sorted array:', arr)
```

Time Complexity:

The time complexity of Bubble Sort is O(n^2) in the worst and average cases, where n is the number of elements in the list. This is because there are two nested loops, each running n times. In the best case, if the list is already sorted, the time complexity is O(n).

Additional Explanations for Bubble Sort

Why Does Bubble Sort Repeatedly Compare Adjacent Elements?

Bubble Sort repeatedly compares adjacent elements because it works by ensuring that each pair of neighboring elements is in order. By doing this for every element in each pass, the algorithm allows the largest unsorted element to 'bubble' to the end of the list. This process continues, each time excluding the last sorted elements from the next pass, until the entire list is sorted. The repetitive comparison is key to gradually moving each unsorted largest element to its correct position at the end of the list.

How to Swap Elements in Bubble Sort

In Bubble Sort, elements are swapped when an element is found to be greater than the next element. Swapping is essential to move the larger element towards the end of the list.

In Python, swapping can be done in one line using multiple assignment, as shown below:

```
arr[j], arr[j+1] = arr[j+1], arr[j]
```

This line simultaneously assigns the value of arr[j+1] to arr[j] and arr[j] to arr[j+1], effectively swapping the two elements without needing a temporary variable.