Bubble Sort

bluesky: @leonlonsdale.dev
discord: https://discord.gg/dhrdFh98UA

How it works

- A sorting algorithm
- Works by looping through a collection and dragging the largest item to the end.
- The range of the loop decreases on each iteration using a variable decrement.
- This avoids rechecking the values at the end of the collection already sorted in previous iterations.

Pseudocode

1. Set `end` to the length of the list (this represents the last unsorted element).
2. Set a flag `sorting` to true (this indicates whether any swaps were made in the current pass).
3. While `sorting` is true:
    - Set `sorting` to false (reset the flag before starting a new pass).
    - Loop through the list from index 1 to `end` (i.e., only check elements up to the current unsorted part of the list):
        - If the current element is smaller than the previous element:
            - Swap the two elements.
            - Set `sorting` to true (indicating a swap was made).
    - Decrease `end` by 1 (because the largest element has now been placed in the correct position at the end).
4. Once `sorting` is false, the list is sorted, and the algorithm ends.

Example Code

```python
def bubble_sort(arr):
    end = len(arr)
    sorting = True

    while sorting:
        sorting = False

        for i in range(1, end):
            if arr[i] < arr[i - 1]:
                arr[i], arr[i - 1] = arr[i - 1], arr[i]
                sorting = True

        end -= 1

    return arr
```

Complexity

Time Complexity

The Time Complexity of Bubble Sort is $O(n^2)$

Worst case

The worst case would be if the list is reversed. In this case:

1. **While Loop ($O(n)$)**: In the worst case the `while` loop will run `n` times because he `sorting` flag is reset to `True` in each iteration of the inner loop.
2. **Inner Loop ($O(n)$)**: The inner loop will iterate through the entire list in each pass. In the worst case, each pass requires O(n) operations, as the largest unsorted element will be moved to its correct position in every iteration.

Overall: $O(n^2)$

Best Case

The best case would be if the list is sorted. I this case:

1. **Inner Loop ($O(n)$)**: The inner loop still needs to iterate through the entire list once to confirm that no changes are necessary. It therefore runs in O(n) time.
2. **While Loop ($O(1)$)**: The `while` loop will only run once because no swaps are made in the first pass. The `sorting` flag will remain `False`, causing the while loop to exit after the first iteration. This gives us O(1).

Overall: $O(n)$

Space Complexity

The space complexity of Bubble Sort is $O(1)$.

- **In-place Sorting:** Bubble Sort is an in-place sorting algorithm, meaning it sorts the elements of the list without requiring additional storage or memory.
- **Constant Space Usage:** The only extra space used is for variables like the `sorting` flag and loop counters (`i`, `j`), which take up a fixed amount of space that does not depend on the size of the input list.

As a result, the space complexity of Bubble Sort is $O(1)$.