BUBBLE SORT

UNIT 3: SORTING & SEARCHING

WHAT IS BUBBLE SORT ?

- Bubble sort is a simple sorting algorithm that repeatedly steps through a list, compares adjacent elements, and swaps them if they are in the wrong order.
- The pass through the list is repeated until the list is sorted.

- The name "bubble sort" comes from the way smaller elements "bubble" to the top of the list, similar to how bubbles rise to the surface of a liquid.
- In each pass, the largest (or smallest, depending on the sorting order) element gradually "bubbles" up to its correct position.

WORKING OF BUBBLE SORT

1) Start at the beginning of the list.
2) Compare the first and second elements. If they are in the wrong order, swap them.
3) Move to the next pair of adjacent elements and repeat step 2. Continue until reaching the end of the list.
4) If any swaps were made during the pass, repeat steps 2 and 3 for the entire list again. Otherwise, the list is already sorted, and the algorithm terminates.
5) Repeat steps 1-4 until the list is fully sorted.

The algorithm's time complexity is O(n^2) in the worst and average case, where n is the number of elements in the list.

It is inefficient for large lists, but it can be useful for small lists or partially sorted lists.

Bubble Sort in 2

Bubble sort in 2 minutes, (Jul. 26, 2016). Accessed: Jul. 07, 2023. [Online Video]. Available: https://www.youtube.com/watch?v=xli_FI7CuzA

IMPLEMENTATION OF BUBBLE SORT

```
def bubble_sort(arr):
    n = len(arr)
    for i in range(n):
        # Perform one pass through the list
        for j in range(0, n - i - 1):
            if arr[j] > arr[j + 1]:
                # Swap elements if they are in the wrong order
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
```

1.The function **bubble_sort** takes an input list **arr** as a parameter.

2.n = len(arr) stores the length of the list in the variable n.

3.The outer loop **for i in range(n):** iterates n times, representing the number of passes through the list.

4.Inside the outer loop, the inner loop for **j in range(0, n - i - 1):** iterates over the unsorted portion of the list.

5.The condition **if arr[j] > arr[j + 1]:** checks if the current element is greater than the next element in the list.

6.If the condition is true, it means the elements are in the wrong order, and the following swap operation is performed:

7.After the inner loop completes one pass, the largest element in the unsorted portion "bubbles" up to its correct position at the end of the list.

8.This process repeats for each pass of the outer loop until the entire list is sorted in ascending order.

Download Lecture Slides & Executable Programs from

https://github.com/ashiqirphan-AI/AD3251-DSD-Programs