Bubble Sort

FINAL PROJECT PSA

PRESENTED BY

Mehaboobjan Shaik

Table of Contents

PROJECT REVIEW REPORT

Proposal

In this report the focus is to understand the basic understanding of bubble sort how it works, examples, implementation and visualization of algorithm using JavaScript.

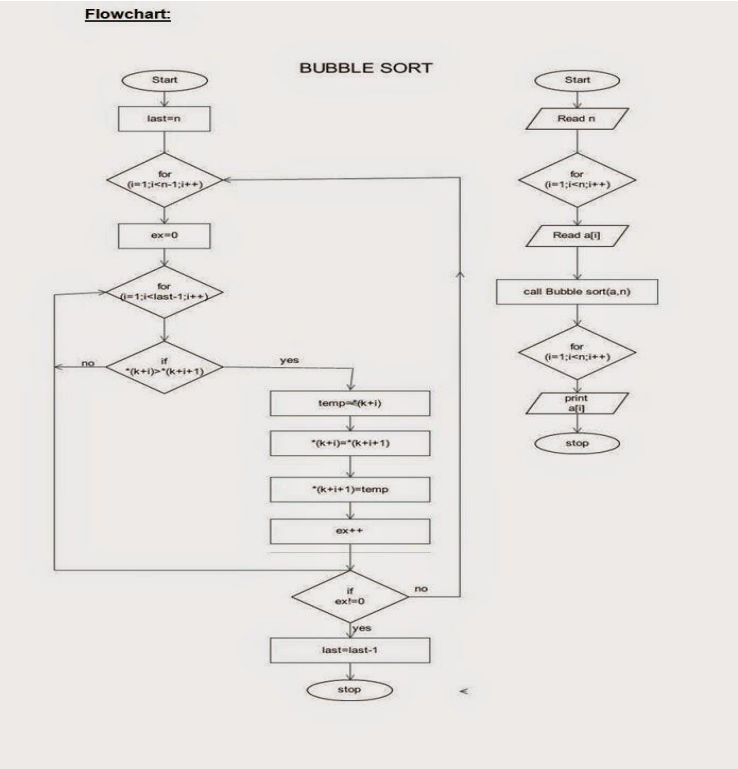
Introduction

Bubble sort algorithm is mainly used for sorting a string of numbers or elements in an order where comparing each element with adjacent elements starting from left to right once per iteration.

How it works:

·This algorithm works with comparing the first element with adjacent element at the next position and if the element at current position is greater than position swap two elements, if the element is not greater we will not swap its position.
·Now, we have to just move on to next position, and we have to repeat the same process as above; after reaching the last element then we are done with one pass off the array, repeat the same for (n – 1) iterations or pass to sort the list where n is the total number of elements in the list

Flow chart:



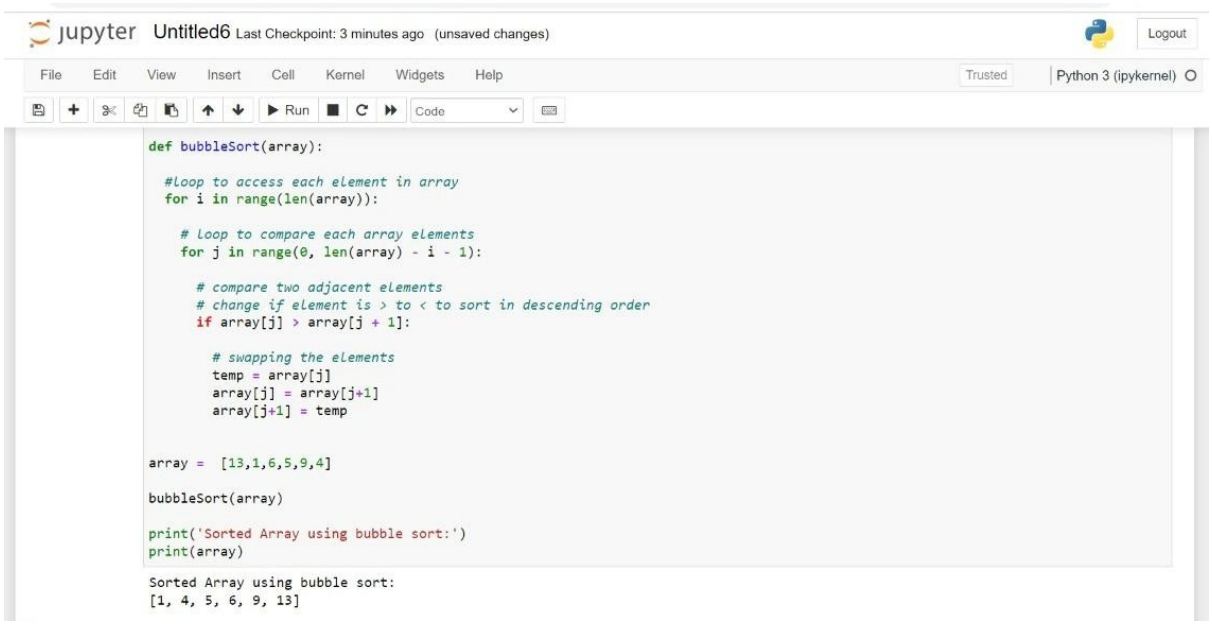
pseudo Code:

```
def bubbleSort(array):

  #loop to access each element in array
  for i in range(len(array)):

    # loop to compare each array elements
    for j in range(0, len(array) - i - 1):

      # compare two adjacent elements
      # change if element is > to < to sort in descending order
      if array[j] > array[j + 1]:

        # swapping the elements
        temp = array[j]
        array[j] = array[j+1]
        array[j+1] = temp
```

Driver code to test above:

```
array = [13,1,6,5,9,4]
bubbleSort(array)
print('Sorted Array using bubble sort:')
print(array)
```

Output:



Time complexity of Bubble sort:

Where C is running time of each loop

T(n) = (n-1) × (n-1) × C
= Cn^2 – 2Cn + 1
= Polynomial expression of time then time complexity be O(n^2)

Therefore time complexity of Bubble sort be O(n^2)

Average Case

- Bubble sort requires (n/2) passes and O(n) comparisons for each pass.
- Therefore, the average case time complexity of bubble sort is O(n/2 x n) = O(n/2 x n) = O(n/2 x n) = O(n/2 x n) = O (n^2).

Best Case

- In the best-case, the array is already sorted, bubble sort performs O(n) comparisons.
- Therefore the time complexity of bubble sort in the best-case is O(n).

Worst Case

- In the worst-case, the outer loop runs O(n) times.
- Hence, the worst-case time complexity of bubble sort is O(n x n) = O (n^2).

The Space Complexity of the Bubble Sort Algorithm:

- Bubble sort requires fixed amount of extra space for the flag, i, and size variables.
- Therefore, the space complexity of bubble sort is O(1).

Example:

Let us take a array A with 6 elements indexes 0 to 5

0	1st.	2nd.	3rd.	4th.	5th
2	7	4	1	5	3

- To rearrange the elements in the array in increasing order, we are going to scan the array from left to right (n-1) times, where n is number of elements.
- first compare the 0th index to next element and then the element at 2nd index and so on of course, if the element at the current position is greater than the element that the next position we will swap the 2 elements in this case 2 is not greater than 7 so we will not swap the 2 elements we will just move on to the next position.

0	1st.	2nd.	3rd.	4th.	5th
2	4	7	1	5	3

- Here, once again we will compare the elements at this position with its next element and if it is greater we'll swap the 2 elements in this case 7 is greater than 4 so we'll swap the position of these 2 elements so 7 will move to index 2 and 4 will move to index one.

0	1st.	2nd.	3rd.	4th.	5th
2	4	1	7	5	3

- Now we will move to index 2 the number index 2 at this stage will be 7 once again we will look at the next element, 7 is again greater than 1 so we'll swap.

0	1st.	2nd.	3rd.	4th.	5th
2	4	1	5	7	3

- and now we will move to index 3 once again at this stage the number at index 3 is 7 we will compare it with 5 and we need to swap again so 7 will go to index 4 and 5 will move to index 3.

0	1st.	2nd.	3rd.	4th.	5th
2	4	1	5	3	7

- and we'll go to index 4 as you can see is it step this is at each step when we're at position 4 once again 7 is greater than 3 so we'll swap there is no next element for index 5, this whole process is pushing number 7 which is the largest in the array towards higher end.
- and at this stage we are done with one pass on the array and what has happened after this one passes that 7 which is the largest in the array is at it's appropriate position it deserves to be at position at index 5 in the sorted array. So, 7 has kind of bubbled up to the right most position in the array.

0	1st.	2nd.	3rd.	4th.	5th
2	1	4	3	5	7

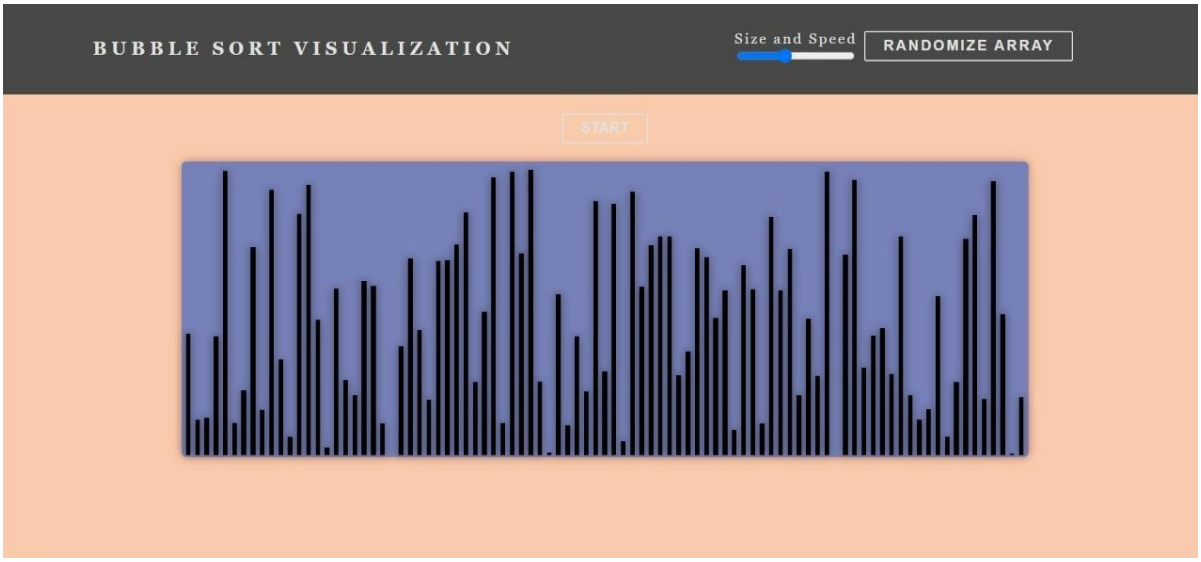
- At this state of the array after one pass we perform another pass and once again keep comparing the adjacent elements and performing swaps if we will do so now the second largest element in this example will end up at index 4 the second largest in the array is number.
- Here we are will be divided into 2 parts, one part will be the sorted part and another part will be unsorted part after 2 passes the part of the array from index 4 till 5 is sorted and the part of the array from index zero till 3 is unsorted with each pass the largest element in the unsorted half will bubble up to the highest index in unsorted half.

0	1st.	2nd.	3rd.	4th.	5th
1	2	3	4	5	7

- so in 3rd pass number 4 should bubble up to position 3 index 3 and while scanning once we reach to the part where we are already sorted there will be no swapping we can actually avoid going to the sorted part.
- it will only improve our algorithm to pass 3 we will be looking like this in fact we are already sorted, in general if we will conduct n -1 i.e; 4such passes for an array of size 5.

Visualization:

1)
Here at first all be bars are un sorted and we can increase the number of bars to be sorted. And can start sorting the bars using START button. And at the end we can RANDOMIZE ARRAY to get the sorted array unsorted.



2)
Here, we have started the sorting the bars using bubble sort algorithm, bars will get compared with their adjacent elements and gets sorted.

3.) Here, we get the final result of all sorted elements.

Application:

- Bubble sort is mainly used to programming television to sort channels based on audience view time.
- One of the main advantages of a bubble sort is it is very simple algorithm to describe to a computer. It has only one task to perform, compare two elements and, if needed, swap them.

Disadvantage:

- The only disadvantage with a bubble sort algorithm is that it takes a very long time to sort a large number of elements in a single array.

Conclusion:

In this report we discussed about bubble sort its example, applications, advantages, disadvantages, and visualization of algorithm & Bubble sort is probably one of the most popular and simple sorting algorithms.

References:

- https://en.wikipedia.org/wiki/Bubble_sort
- https://brilliant.org/wiki/bubble-sort/
- https://www.geeksforgeeks.org/bubble-sort/
- https://www.tutorialspoint.com/data_structures_algorithms/bubble_sort_algorithm.htm