**Bubble Sort Vs Heap Sort**

**Submitted by - Anu Thomas**

**Submitted on - 07/09/2022**

**BUBBLE SORT**

Bubble sort is a sorting algorithm that matches two adjacent pairs things and change them until they are in the order you want.

**Working of Bubble Sort:**

1. Initial step

From the first index, compare the first and the second elements. If the first element is greater than the second element, they are swapped. Then, compare the second and the third elements. Swap them if they are not in order. This process goes on until the last element.

1. Remaining Iteration

The same process goes on for the remaining iterations. After each iteration, the largest element among the unsorted elements is placed at the end. In each iteration, the comparison takes place up to the last unsorted element. The array is sorted when all the unsorted elements are placed at their correct positions.

**Algorithm**

**Step 1:** Start

**Step 2:** Get the list number of items to be sorted.

**Step 3:** Determine the number of external passes (n - 1) to be performed. Its length is list minus one.

**Step 4:** Perform inner passes (n – 1) times for outer pass 1. Get the first element value and compare the value with the second value. If the second value is less than the first value, then swap the positions( in the case of ascending order).

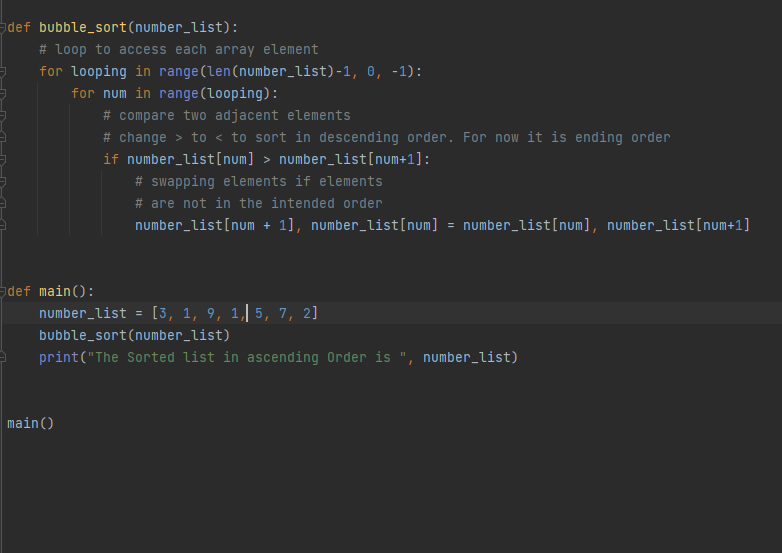
**Step 5:** Repeat step 4 until you reach the outer pass . Get the next element in the list then repeat the process that was performed in step 4 until all the values have been placed in their correct ascending order.

**Step 6:** Return the result when all passes have been done.

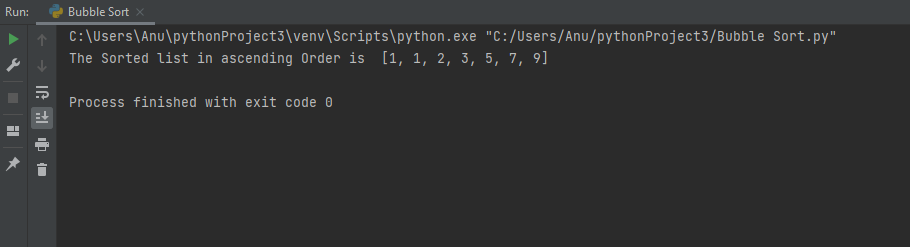
**Step 7:**  Print result.

**Step 8**: Stop

For Example here is the python program for doing bubble sort in ascending order.



And the output is :



**Optimized Bubble Sort:**

If the given list is already sorted, comparing all values is a waste of time and resources. Optimizing the bubble sort helps us to avoid unnecessary iterations and save time and resources.

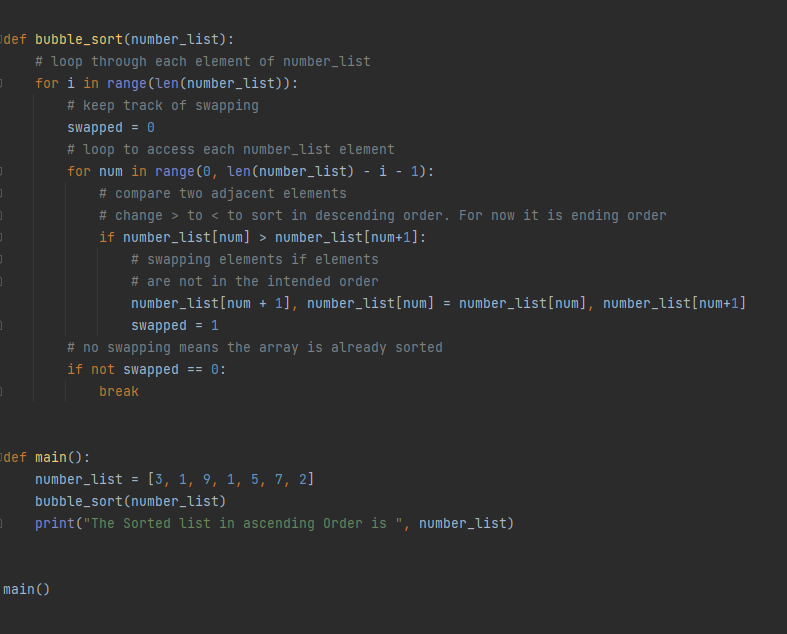
Optimization is done using the following steps:

Step 1: Create variables that monitors if any swapping has occurred or not

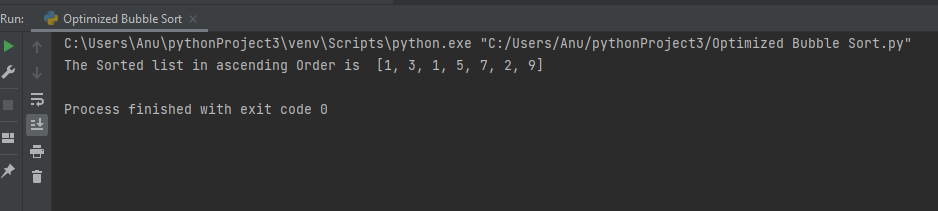
Step 2: If the values have swapped positions, continue to the next iteration

Step 3: If the benefits have not swapped positions, terminate the inner loop, and continue with the outer loop.

Here is the Python program for doing optimized bubble sort in ascending order:



Output:



An optimized bubble sort is more efficient as it only executes the necessary steps and skips those that are not required.

**Bubble sort advantages**

The following are some of the advantages of the bubble sort algorithm:

* Easy to understand
* It does not require extensive memory.
* It is easy to write the code for the algorithm.
* The space requirements are minimal compared to other sorting algorithms.

**Bubble sort Disadvantages**

The following are some of the disadvantages of the bubble sort algorithm:

* It does not perform well when sorting large lists.
* It takes too much time and resources.
* It’s mostly used for academic purposes and not the real-world application.
* The number of steps required to sort the list is of the order n2.

**HEAP SORT**

Heap Sort is a popular and efficient sorting algorithm in computer programming. Heap sort works by visualizing the elements of the array as a special kind of complete binary tree called a heap.

**Working of Bubble Sort:**

A complete binary tree has an interesting property that we can use to find the children and parents of any node.

0

0

1

5

20

6

8

9

1

2

9

7

5

6

**Min heap**

**Max Heap**

* Since the tree satisfies Max-Heap property, then the largest item is stored at the root node.
* Swap: Remove the root element and put at the end of the array (nth position) Put the last item of the tree (heap) at the vacant place.
* Remove: Reduce the size of the heap by 1.
* Heapify: Heapify the root element again so that we have the highest element at root.
* The process is repeated until all the items of the list are sorted.

**Algorithm**

Step1: Start

Step 2: Get the list number of items to be sorted and get the length.

Step 3: Create a max heap and pass list, length and index.

Step4: Create pointers for parent(i) and child node(for left= 2\*i+1, for right= 2\*i+2).

Step5: Check if left and right child of root exist and greater than root. And if child is greater than parent set greater value to parent .

Step 6: If parent is change then do swapping between parent index and largest index. And call max heap

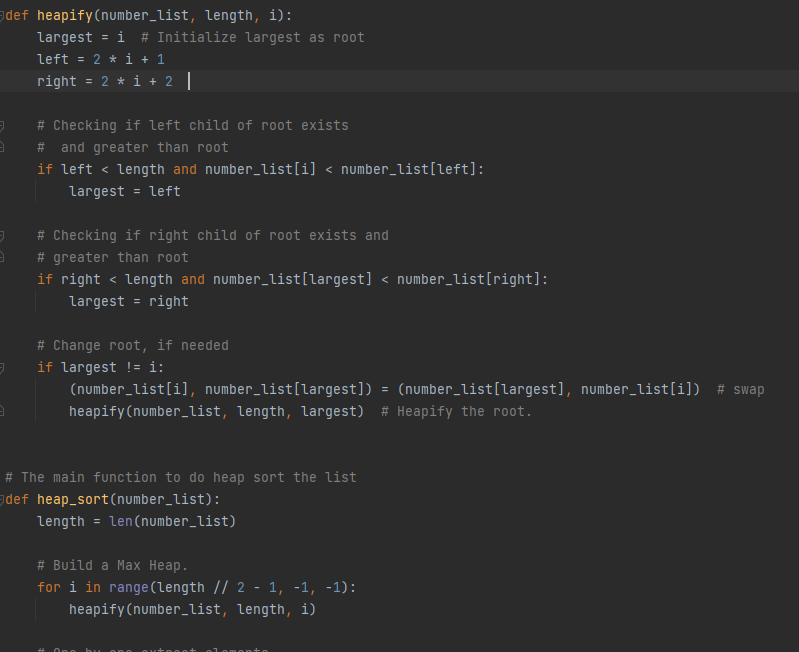
Step7: Swap last element and root and call max heap.

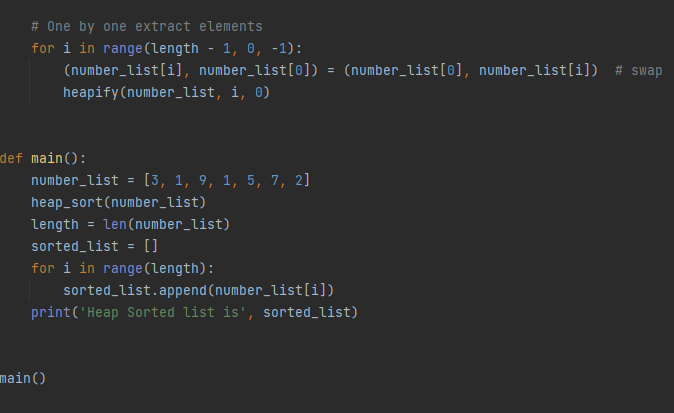
Step 8: Repeat step 4, 5, 6 and 7 until we get the sorted list

Step 9: Print result.

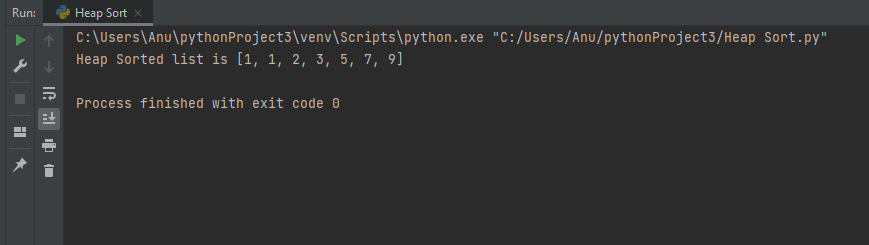
Step 10: Stop

Program for doing Max heap:





Output:



**Comparison between Bubble sort and Heap sort:**

|  |  |
| --- | --- |
| **Bubble Sort** | **Heap Sort** |
| Compare adjacent elements pairs and swaps if they are not in the intended order. | Proceed by heapify , swap and insert |
| Used to sort small data set. | Used to sort big data set |
| Not efficient | More efficient |
| Slower | Faster |
| Extremely efficient in terms of memory usage | Memory usage is minimal |
| Simple code | Complex than bubble sort |