Heap Sort is a comparison-based sorting algorithm that uses a binary heap data structure to sort elements.

It involves two main steps: building a heap from the input list and repeatedly extracting the maximum (or minimum) element from the heap to obtain the sorted list.

Here's how the Heap Sort algorithm works:

1. Build a max-heap from the input list:

- Starting with the second-to-last level and moving upwards, heapify each subtree by comparing the parent node with its children and swapping if necessary.
- Continue this process until the entire list is transformed into a max-heap, where each parent node is greater than or equal to its children (in case of a max-heap).

2. Repeatedly extract the maximum element from the heap and place it at the end of the list:

- The maximum element is always located at the root of the max-heap. Swap it with the last element in the heap (which is the smallest remaining element in the unsorted part of the list).
- Reduce the heap size by one to exclude the extracted element from future consideration.
- Restore the heap property by heapifying the new root (the previously last element of the heap) to maintain the max-heap structure.

3. Repeat step 2 until all elements have been extracted from the heap:

- Each iteration will extract the maximum element and place it at the corresponding position in the sorted part of the list.

Example:

Consider the following unsorted list of integers: [5, 3, 8, 2, 1].

Step 1: Building the max-heap

- Convert the list into a max-heap: [8, 5, 3, 2, 1].

Step 2: Extracting elements from the max-heap

- Extract the maximum element (8) and swap it with the last element (1). Reduce the heap size.
- The list becomes [1, 5, 3, 2, 8].
- Heapify the new root (1) to restore the max-heap structure: [5, 2, 3, 1, 8].
- Extract the maximum element (5) and swap it with the last element (1). Reduce the heap size.
- The list becomes [1, 2, 3, 5, 8].
- Heapify the new root (1) to restore the max-heap structure: [3, 2, 1, 5, 8].
- Extract the maximum element (3) and swap it with the last element (1). Reduce the heap size.
- The list becomes [1, 2, 1, 5, 3].
- Heapify the new root (1) to restore the max-heap structure: [2, 1, 1, 5, 3].
- Extract the maximum element (2) and swap it with the last element (1). Reduce the heap size.
- The list becomes [1, 1, 2, 5, 3].
- Heapify the new root (1) to restore the max-heap structure: [1, 1, 2, 5, 3].

- Extract the maximum element (1) and swap it with the last element (3). Reduce the heap size.
- The list becomes [1, 1, 2, 3, 5].
- Heapify the new root (1) to restore the max-heap structure: [1, 1, 2, 3, 5].

Step 3: All elements have been extracted

- The final sorted list is [1, 1, 2, 3, 5, 8].

Heap Sort has a time complexity of O(n log n) in all cases, where ‘n’ is the number of elements in the list.

It is an in-place sorting algorithm and is not affected by the initial order of the elements. However, Heap Sort requires additional space for the heap structure.