Table of Contents

Heap Sort Visualizer User Manual

Overview

Algorithms are a fundamental concept in computer science, used to solve problems and automate processes in a variety of fields. Sorting algorithms are an important concept in computer science and the Heap Sort algorithm is one of the simplest and most commonly taught sorting algorithms.
In this project, we aimed to visualize the HeapSort algorithm using HTML and JS. Our visualization provides an animated representation of the algorithm in action, highlighting the steps involved in the algorithm and the elements being compared and swapped.
Our project aimed to provide a visual representation of the HeapSort algorithm that would make it easier to understand and learn. We achieved this by creating an engaging and interactive visualization with easy to use user interface that demonstrates the workings of the HeapSort algorithm.
 To create our visualization, we used HTML as our graphical user interface and JS as ourprogramming language. We developed an algorithm that animates the HeapSort algorithm step by step, highlighting the elements being compared and swapped. Our project also includes a feature that allows users to interact with the visualization and see the changes in the sorting algorithm in real-time.
Furthermore, it also includes the use of different searching methods to understand the full potential of the sorting algorithm by seeing how it benefits and can be used in real time problems.
Our visualization of the HeapSort algorithm provides a unique and effective way to teach and learn about sorting algorithms. We believe that our project has the potential to help students and educators better understand the HeapSort algorithm and its applications. Thus this report, we will describe our ethodology for visualizing the HeapSort algorithm, present the results of our project and discuss the strengths and limitations of our visualization.

Keywords: Algorithms, Sorting algorithms, Heapsort, Searching, Linear search, Binary search, JS,HTML, Visualization, User interface.

Heap Sort Visualizer User Manual

Introduction

• Algorithms and Sorting algorithms
Algorithms are a fundamental concept in computer science, used to solve problems and automate processes in a variety of fields. At its simplest, an algorithm is a set of instructions that a computer
program follows to complete a task. These instructions can range from simple, straightforward steps
to complex decision-making processes that involve multiple variables and conditions.
The importance of algorithms cannot be overstated, as they are used in virtually every aspect of computer science and technology. From simple arithmetic operations to complex machine learning
algorithms, algorithms are essential for solving problems and automating tasks. They are used in a wide range of applications, including web search engines, image recognition software, financial modelling and more.
In addition to their practical applications, algorithms are also a fascinating and intellectually stimulating subject in their own right. Studying algorithms provides insights into the underlying
principles of computer science and how computers can be used to solve problems in innovative and
creative ways.
One of the most common type of algorithm used all the time is Sorting algorithm.
Sorting algorithms are used to arrange data and information in a logical and efficient manner. A sorting algorithm is a set of instructions that a computer program follows to rearrange a collection of data in a specific order.
There are many types of sorting algorithms, each with their own strengths and weaknesses, making them suitable for different use eases and applications.
Sorting algorithms are used in a wide range of applications, from simple tasks such as sorting a list of names alphabetically to more complex tasks such as sorting large datasets for analysis. They are also
used in a variety of fields, including computer science, finance and data analysis.
Sorting algorithms can be broadly classified into two categories: comparison-based sorting algorithms
and non-comparison-based sorting algorithms. Comparison-based sorting algorithms compare elements of the collection being sorted and determine their relative order, Examples of comparison-
based sorting algorithms include Heap Sort, Quick Sort and Merge Sort.
Non-comparison-based.

Heap Sort Visualizer User Manual

In this project we will be discussing about Heapsort algorithm, one of the Hardest sorting algorithms out there and take you through the process how we achieved its visualization through code implementations.

- **Heap Sort**

Heap sort is a comparison-based sorting algorithm that uses the concept of a heap data structure to sort elements. This user manual will guide you through using the heap sort visualizer, which demonstrates the step-by-step process of heap sort and provides a hands-on learning experience.
This is a comparison-based sorting algorithm that uses the concept of a binary heap data structure to efficiently sort elements. It operates by converting the input array into a binary heap, a complete binary tree with a special property called the heap property. The heap property states that for a max heap, the value of each parent node is greater than or equal to the values of its children nodes, while for a min heap, the value of each parent node is less than or equal to the values of its children nodes.

Heap sort consists of two main phases: building the heap and extracting elements.

1. Building the Heap:
   - Starting with the given input array, the algorithm builds a heap structure by iterating through the array from the last non-leaf node to the first node.
   - During each iteration, the algorithm compares the node with its children and swaps them if necessary to satisfy the heap property.
   - This process is called "heapify" and ensures that the largest (or smallest) element ends up at the root of the heap.

2. Extracting Elements:
   - Once the heap is built, the algorithm repeatedly extracts the root element and places it at the end of the array.
   - After removing the root, the heap property may be violated, so the algorithm performs heapify on the remaining elements to restore the heap property.
   - This process continues until all elements have been extracted, resulting in a sorted array.

There are two types of heaps commonly used in heap sort:

1. Max Heap:
- In a max heap, the value of each parent node is greater than or equal to the values of its children nodes.
- The maximum element is always stored at the root of the heap.
- Max heap is used to sort elements in ascending order using heap sort.

2. Min Heap:
- In a min heap, the value of each parent node is less than or equal to the values of its children nodes.
- The minimum element is always stored at the root of the heap.
- Min heap is used to sort elements in descending order using heap sort.

Both types of heaps, max heap and min heap, can be used interchangeably with the heap sort algorithm by adjusting the comparison condition during heapify and element extraction.

Overall, heap sort offers a time complexity of O(n log n), making it an efficient sorting algorithm for large datasets. Its use of the heap data structure allows for in-place sorting without requiring additional memory.

3. Getting Started

Implementation of Heap Data Structure: The project involves designing and implementing the Heap data structure, which will serve as the foundation for the sorting algorithm. The Heap structure should support the creation of both max heaps and min heaps.

Here, you will find instructions on how to set up and launch the heap sort visualizer on your computer. The requirements, installation steps, and any dependencies will be covered in this section.
We will need VS code to run the code.

4. Using the Visualizer

This section walks you through the different features and components of the heap sort visualizer. It explains the user interface and how to navigate through the visualizations.

• Initial setups

First, we needed to understand how the Heap Sort algorithm works. We researched and studied the
algorithm and identified the steps involved in it. We also learned about the complexities and limitations of the algorithm.

Then we had to start setting up our environment. We set up the environment by installing JS and
the library that includes HTML. These libraries provided the tools we needed to create the graphical user interface and handle the data manipulation and visualization. We researched from various
resources to understand how the initial program needs to be staned and what is need to be done accordingly.

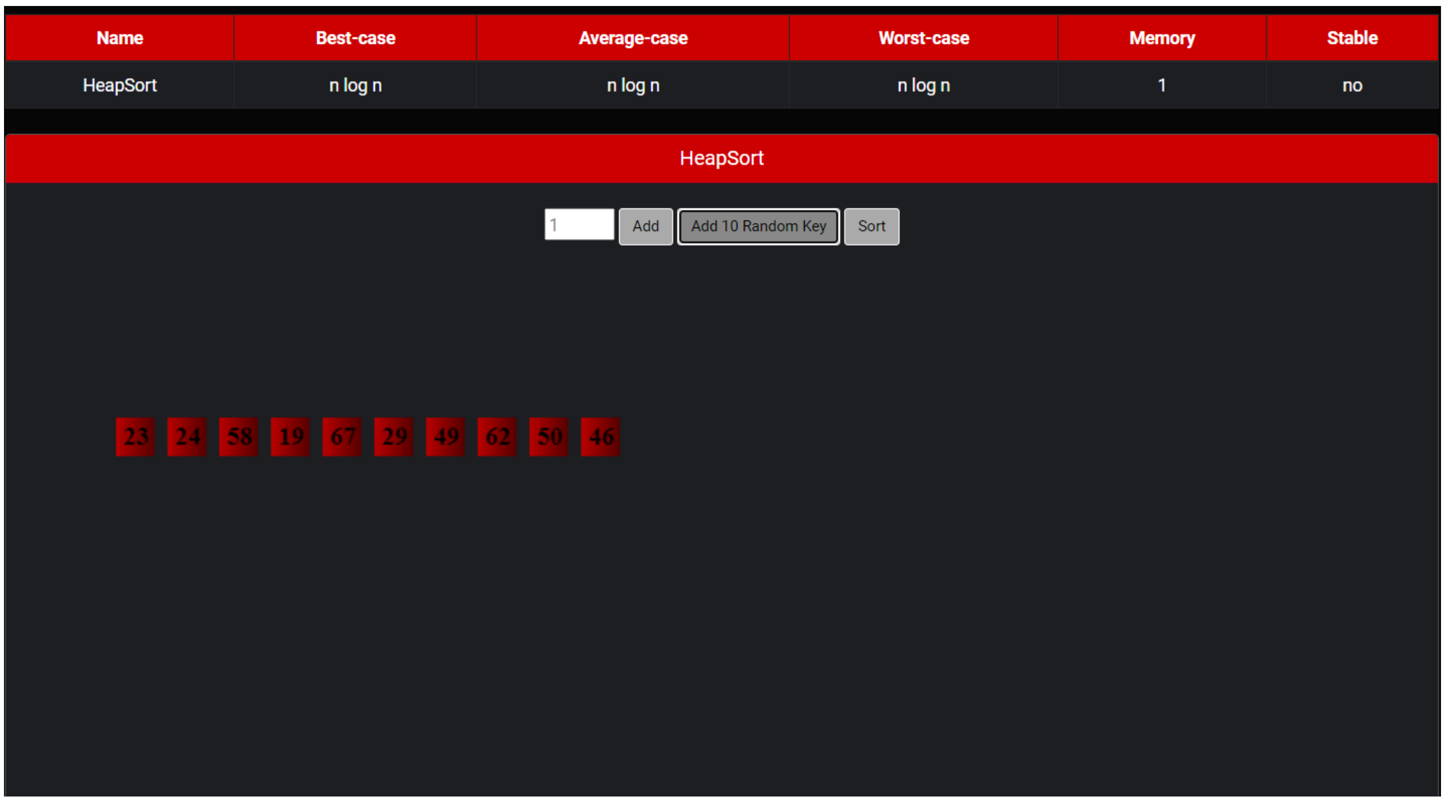
First, we started the project by simply creating a HTML and simple UI. We created a blank while
HTML and added some buttons to generate random data. Then we added the necessary code to draw the rectangles in the HTML. Creating HTML and buttons is very much easy in CSS as all we had to do was to call the pre-build fünctions with the necessary parameters and HTML almost build the widgets for us. Initially There was no input section and we didn't add the sorting algorithm yet.

5. Interacting with the Visualizations

In this section, will learn how to interact with the visualizations to gain a deeper understanding of heap sort. It covers how to control the animation, step through the sorting process, and adjust the speed of the visualization.
Visit:
https://github.com/cseku170202/Fall_21_CSE_2204_Bi_NWU/tree/main/Group%2011/Source%20Code

• **Input and Output**
Our project successfully visualizes the Heap Sort algorithm using HTML and JS. Here are
some screenshots or our visualization in action:

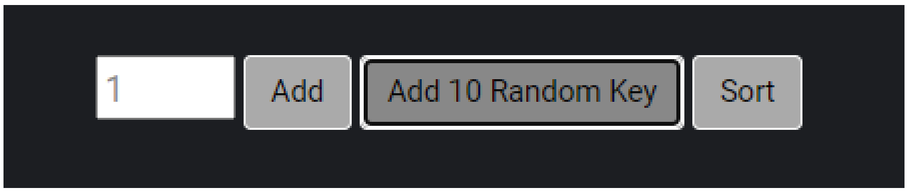
> Add some input array in the dialog box:



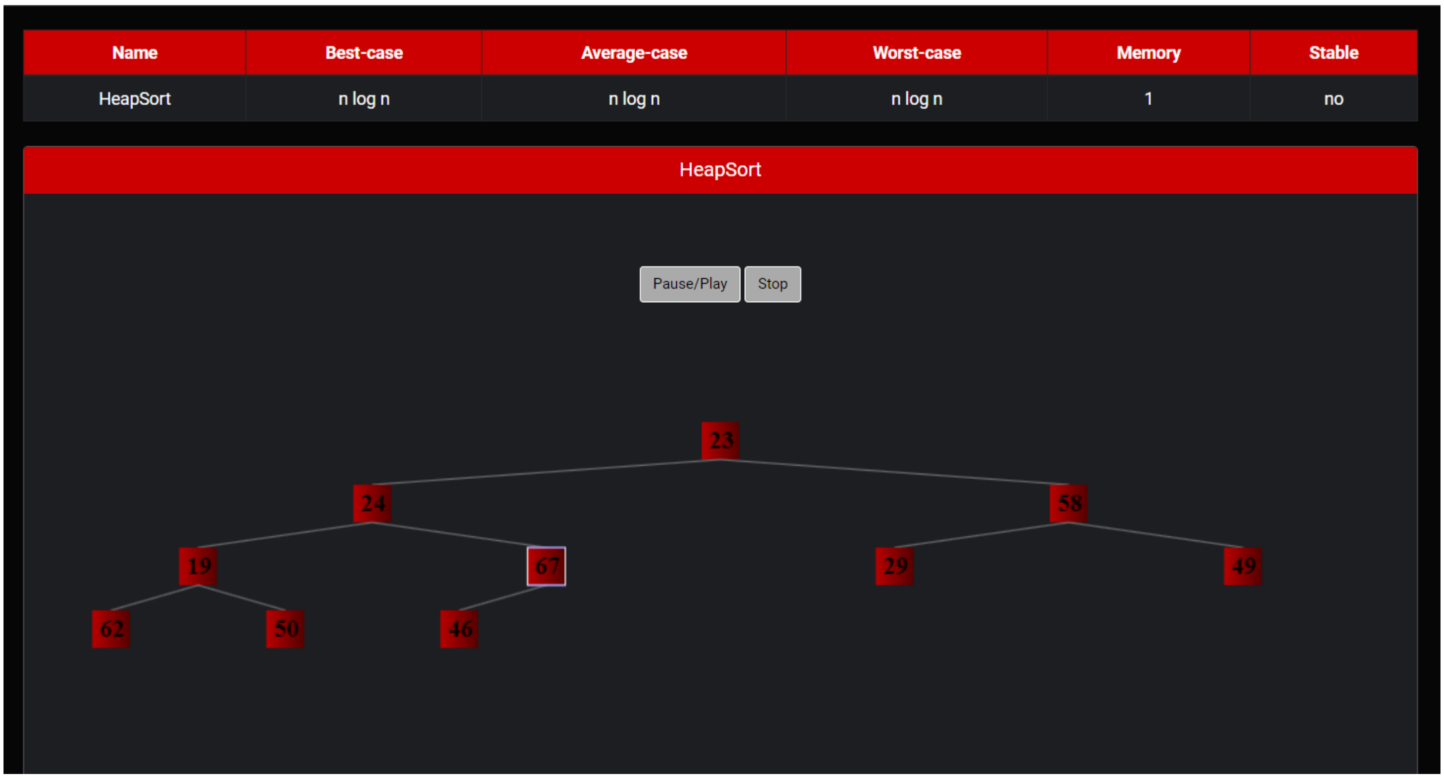
Name	Best-case	Average-case	Worst-case	Memory	Stable
HeapSort	n log n	n log n	n log n	1	no

HeapSort

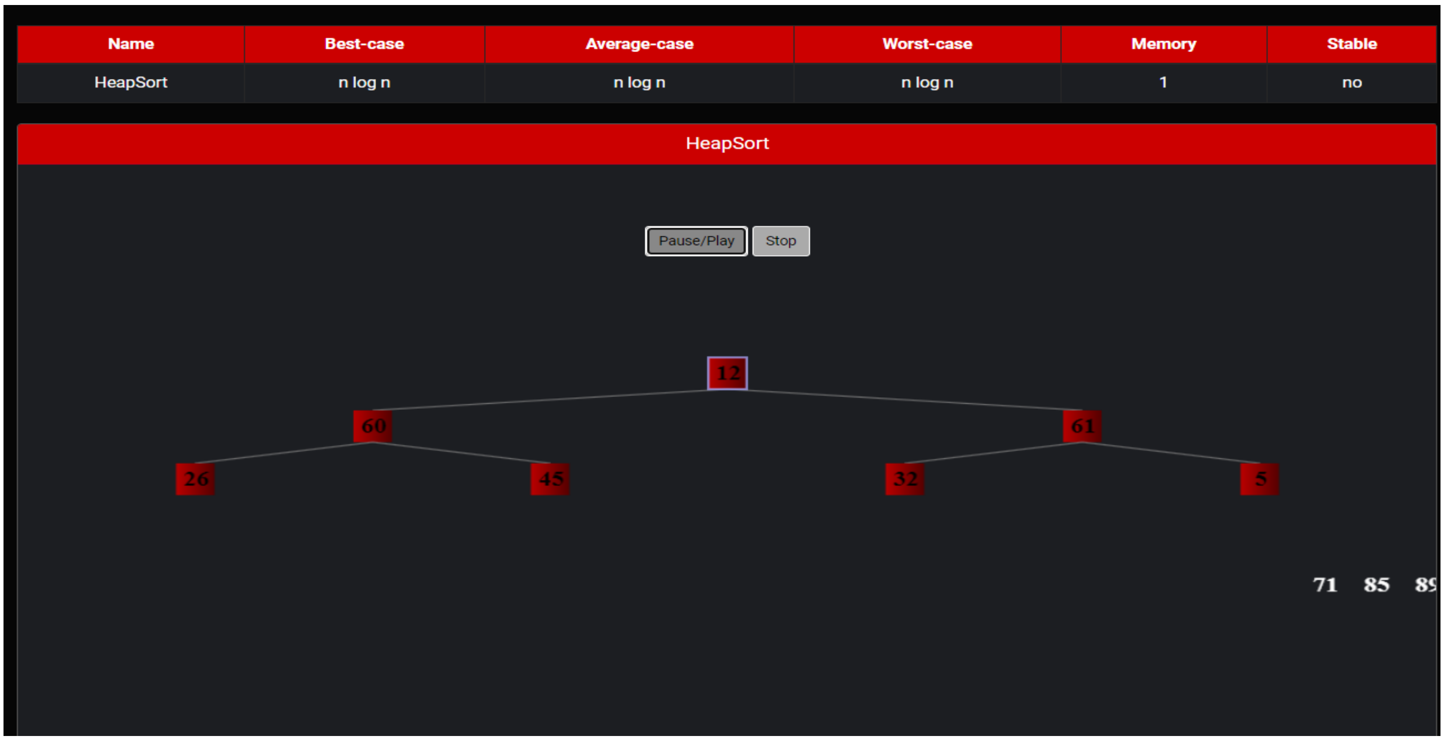
> Now from this 10 number of array We need to click Sort button to start the process:



>The process started:

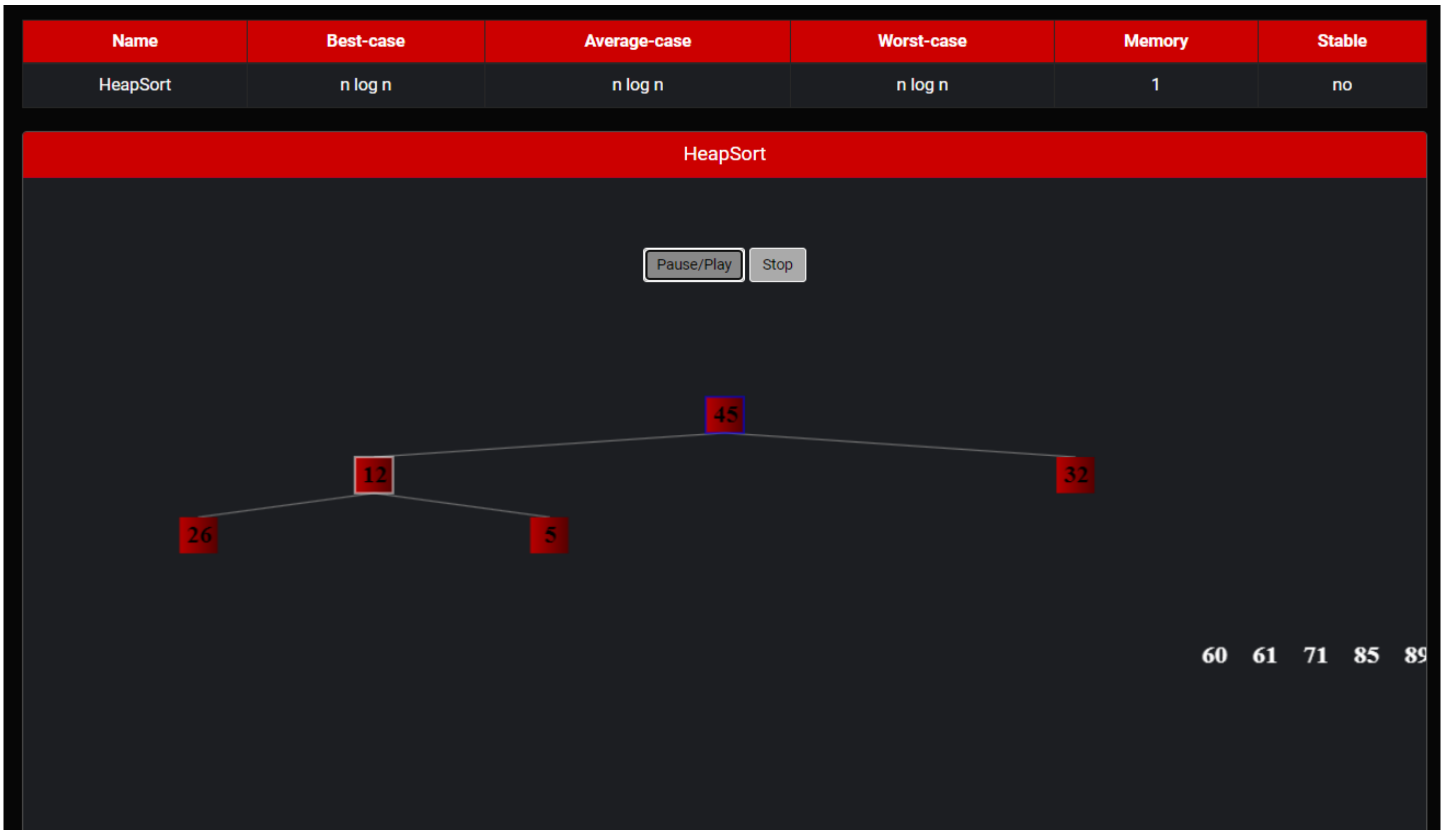


Name	Best-case	Average-case	Worst-case	Memory	Stable
HeapSort	n log n	n log n	n log n	1	no

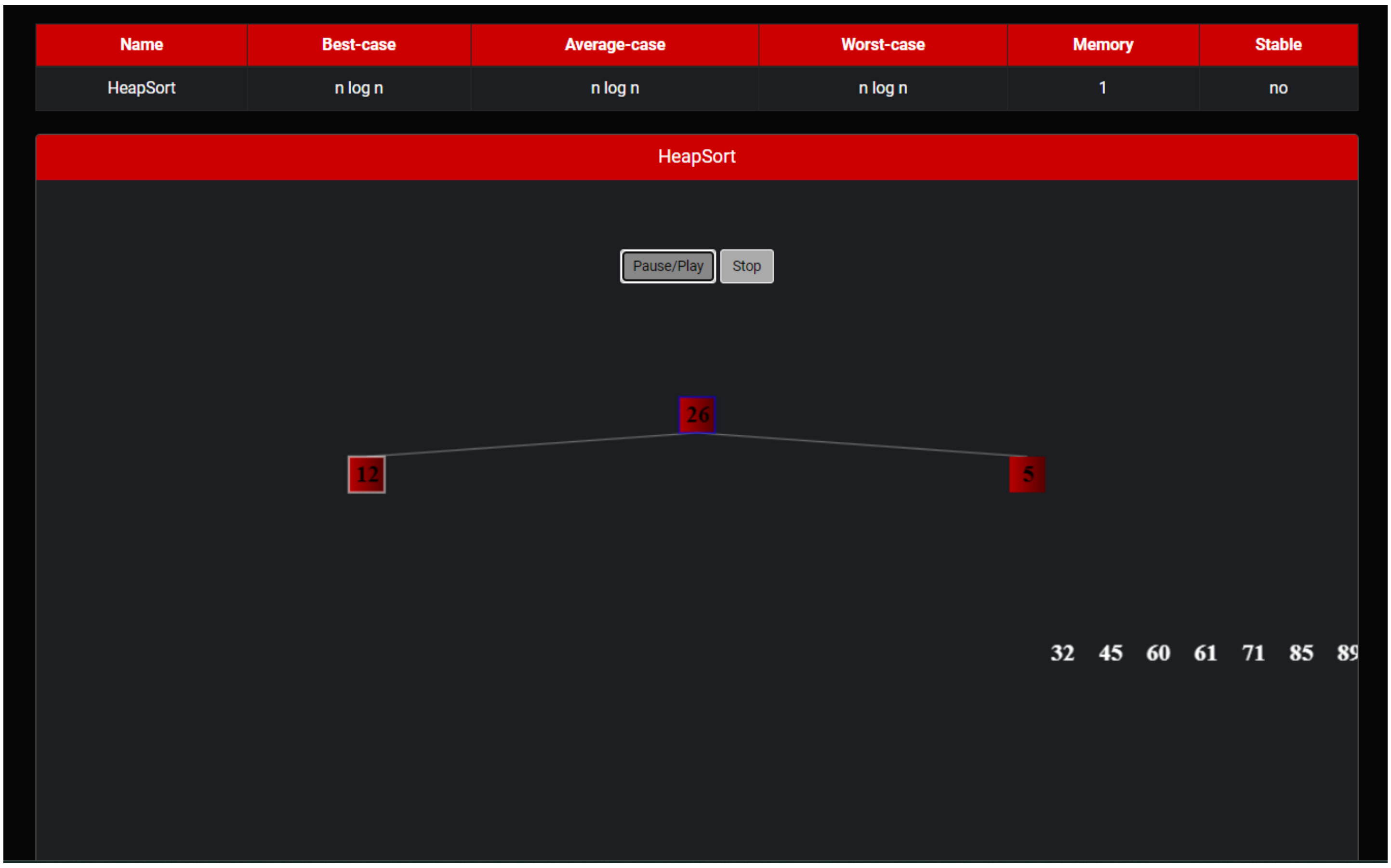


Name	Best-case	Average-case	Worst-case	Memory	Stable
HeapSort	n log n	n log n	n log n	1	no

Heap Sort Visualizer User Manual



Name	Best-case	Average-case	Worst-case	Memory	Stable
HeapSort	n log n	n log n	n log n	1	no



Name	Best-case	Average-case	Worst-case	Memory	Stable
HeapSort	n log n	n log n	n log n	1	no

And finally, we can see the sorted elements at the bottom of the tree. For this case it is sorted as Max heap.

Conclusion

Heap sort is a powerful sorting algorithm, and by using the heap sort visualizer and following this user manual, you will gain a solid understanding of its inner workings. Start exploring the manual and enjoy learning heap sort through interactive visualization.

Developed By:

Mohammad Ziyauddin, Arnob Mistry, Iqbal Mahmud Eshan.
Id: 20212036010, 20212042010, 20212043010

Department Of CSE,
North Western University Khulna, Bangladesh