# Heap Sort Cheatsheet

## Overview

- Heap sort is a comparison-based sorting algorithm that uses a binary heap data structure.
- It works by first creating a heap from the input array, then repeatedly extracting the maximum element and placing it at the end of the sorted array.
- It has a time complexity of O(n log n), making it more efficient than bubble sort, insertion sort, and selection sort.

## Algorithm

```
def heap_sort(arr):
    n = len(arr)

    # Build a max heap
    for i in range(n // 2 - 1, -1, -1):
        heapify(arr, n, i)

    # Extract elements from the heap one by one
    for i in range(n - 1, 0, -1):
        arr[i], arr[0] = arr[0], arr[i]  # swap
        heapify(arr, i, 0)

def heapify(arr, n, i):
    largest = i  # Initialize largest as root
    left = 2 * i + 1  # left = 2*i + 1
    right = 2 * i + 2  # right = 2*i + 2

    # Check if left child of root exists and is greater than root
    if left < n and arr[left] > arr[largest]:
        largest = left

    # Check if right child of root exists and is greater than root
    if right < n and arr[right] > arr[largest]:
        largest = right

    # Change root, if needed
    if largest != i:
        arr[i], arr[largest] = arr[largest], arr[i]  # swap
        heapify(arr, n, largest)
```

## Time Complexity

- Worst-case performance: O(n log n)
- Best-case performance: O(n log n)
- Average-case performance: O(n log n)

## Resources

- Heap Sort Wikipedia
- GeeksforGeeks: Heap Sort
- Visualgo: Heap Sort