In [1]:
```python
import random


# Bubble Sort (O(n^2))
def bubble_sort(arr):
    n = len(arr)
    for i in range(n):
        for j in range(0, n - i - 1):
            if arr[j] > arr[j + 1]:
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
    return arr
```

Bubble Sort: Time: O(n^2) Space: O(1)

In [2]:
```python
# Test different sizes
def test_sorting(algo, size):
    arr = [random.randint(0, 1000) for _ in range(size)]
    sorted_arr = algo(arr.copy())
    return sorted_arr == sorted(arr)
```

In [3]:
```python
#  Quick Sort
def quick_sort_random(arr):
    if len(arr) <= 1:
        return arr

    pivot = random.choice(arr)
    left = [x for x in arr if x < pivot]
    middle = [x for x in arr if x == pivot]
    right = [x for x in arr if x > pivot]

    return quick_sort_random(left) + middle + quick_sort_random(right)



def quick_sort_median(arr):
    if len(arr) <= 1:
        return arr

    first = arr[0]
    middle = arr[len(arr)//2]
    last = arr[-1]
    pivot = sorted([first, middle, last])[1]

    left = [x for x in arr if x < pivot]
    mid = [x for x in arr if x == pivot]
    right = [x for x in arr if x > pivot]

    return quick_sort_median(left) + mid + quick_sort_median(right)
```

Best/Average: T(n) = 2T(n/2) + O(n) Using Master Theorem: a = 2, b = 2 O(n log n)

In [4]:
```python
# Merge Sort
def merge_sort(arr):
    if len(arr) <= 1:
        return arr
```

```
    mid = len(arr)//2
    left = merge_sort(arr[:mid])
    right = merge_sort(arr[mid:])

    return merge(left, right)


def merge(left, right):
    result = []
    i = j = 0

    while i < len(left) and j < len(right):
        if left[i] < right[j]:
            result.append(left[i])
            i += 1
        else:
            result.append(right[j])
            j += 1

    result.extend(left[i:])
    result.extend(right[j:])
    return result
```

T(n) = 2T(n/2) + O(n) Using Master Theorem: O(n log n) Space: O(n)

In [5]:
```
# Heap Sort
def heapify(arr, n, i):
    largest = i
    left = 2*i + 1
    right = 2*i + 2

    if left < n and arr[left] > arr[largest]:
        largest = left

    if right < n and arr[right] > arr[largest]:
        largest = right

    if largest != i:
        arr[i], arr[largest] = arr[largest], arr[i]
        heapify(arr, n, largest)


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

    for i in range(n//2 - 1, -1, -1):
        heapify(arr, n, i)

    for i in range(n-1, 0, -1):
        arr[i], arr[0] = arr[0], arr[i]
        heapify(arr, i, 0)

    return arr
```

Time: O(n log n) Space: O(1)

In [6]:
```
# test
print("Bubble works:", test_sorting(bubble_sort, 20))
```

```
print("Quick random works:", test_sorting(quick_sort_random, 20))
print("Quick median works:", test_sorting(quick_sort_median, 20))
print("Merge works:", test_sorting(merge_sort, 20))
print("Heap works:", test_sorting(heap_sort, 20))
```

```
Bubble works: True
Quick random works: True
Quick median works: True
Merge works: True
Heap works: True
```