# 🟦 Heapsort

## 1️⃣ Grundidee

Heapsort nutzt eine **Heap-Datenstruktur** (meist Max-Heap), um das größte Element effizient zu bestimmen. Der Algorithmus besteht aus zwei Phasen:

1. Aufbau eines Heaps aus der Liste
2. Wiederholtes Entfernen des Maximums und Einfügen ans Listenende

---

## 2️⃣ Voraussetzungen

- ❌ keine
- funktioniert auf **beliebigen vergleichbaren Elementen**

---

## 3️⃣ Laufzeiten & Eigenschaften

| Eigenschaft | Wert |
|---|---|
| Best Case | O(n log n) |
| Average Case | O(n log n) |
| Worst Case | O(n log n) |
| Speicherbedarf | O(1) |
| In-place | ja |
| Stabil | nein |

**Hinweis:** Die Laufzeit ist unabhängig von der Eingabereihenfolge.

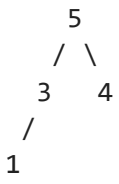
---

## 4️⃣ Schritt-für-Schritt-Beispiel

Ausgangsliste:

```
[5, 3, 4, 1]
```

### Heap-Aufbau (Max-Heap)




### Sortieren

- Tausche 5 mit letztem Element → [1, 3, 4, 5]
- Heapify Rest → [4, 3, 1, 5]
- Tausche 4 → [1, 3, 4, 5]
- Heapify → [3, 1, 4, 5]

Ergebnis:

```
[1, 3, 4, 5]
```

---

## 5️⃣ Besonderheiten / Prüfungsrelevante Hinweise

- Garantierte Laufzeit O(n log n)
- Kein zusätzlicher Speicher
- Schlechtere Cache-Lokalität als Quicksort

---

## 6️⃣ Vor- und Nachteile

### Vorteile

- in-place
- garantierte Laufzeit
- kein zusätzlicher Speicher

### Nachteile

- nicht stabil
- langsamer als Quicksort in der Praxis

---

## 🧠 Merksatz für die Prüfung

*Heapsort sortiert mithilfe eines Heaps in O(n log n), ist in-place, aber nicht stabil.*

---

## 7 Python-Implementierung

In [1]:
```python
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[0], arr[i] = arr[i], arr[0]
        heapify(arr, i, 0)

    return arr
```