Cornell Notes: Heap Sort Algorithm with Python

Cues / Questions

- What is Heap Sort?
- How does heapify work?
- What is time complexity?
- Is it in-place?
- Python code?

Notes

Heap Sort is a comparison-based, in-place sorting algorithm that uses a **binary heap** (usually a max heap).
Steps:

1. Convert array into a max heap.
2. Repeatedly remove the root (largest) and move to the end.
3. Heapify the reduced heap.

Time Complexity:

- Build Heap: $O(n)$
- Reheapify: $O(\log n)$ for each of n elements
- Total: $O(n \log n)$

Space Complexity: $O(1)$ (in-place)
Stability: Not stable

Python Code

```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)

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

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

arr = [4, 10, 3, 5, 1]
heap_sort(arr)
print("Sorted:", arr)
```

Listing 1: Heap Sort in Python

Summary

Heap Sort is an efficient, in-place sorting algorithm with a guaranteed time complexity of $O(n \log n)$. It is useful when memory is constrained and stable sorting is not required.