DIVIDE AND CONQUER: MERGESORT

Link to practice handout

https://bit.ly/Divide-and-Conquer-Practice

Divide and Conquer Algorithms

Algorithm Approach:

- **Divide** a large problem into sub-problems
- **Solve** each sub-problem
- **Combine** the solutions of sub-problems to obtain the solution for the original problem

Merge Sort Algorithm

MergeSort(vector v)

- **Divide v into left half and right half**
- **Sort the left half, then sort the right half**
- **Combine (merge) the two sorted halves**

Example run of mergesort

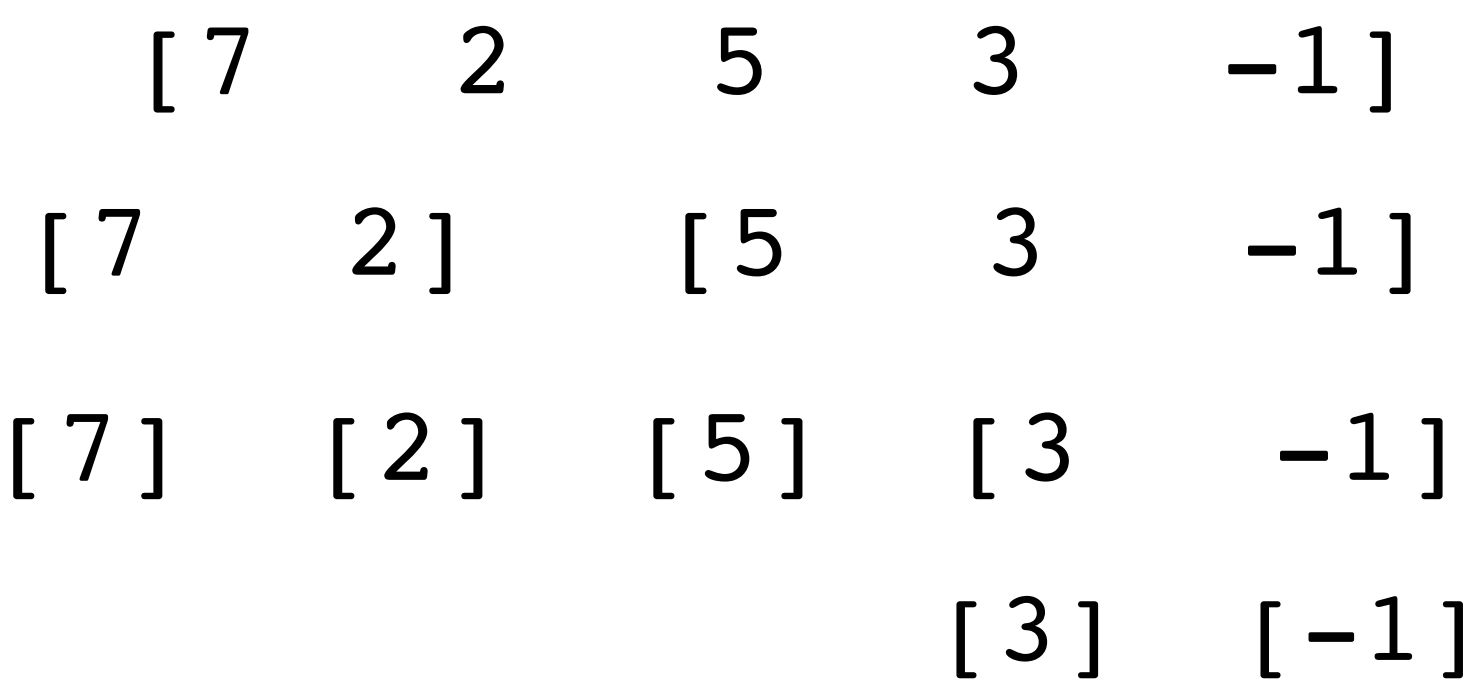
[7 2]

Example run of mergesort

[7 2 3 -1]

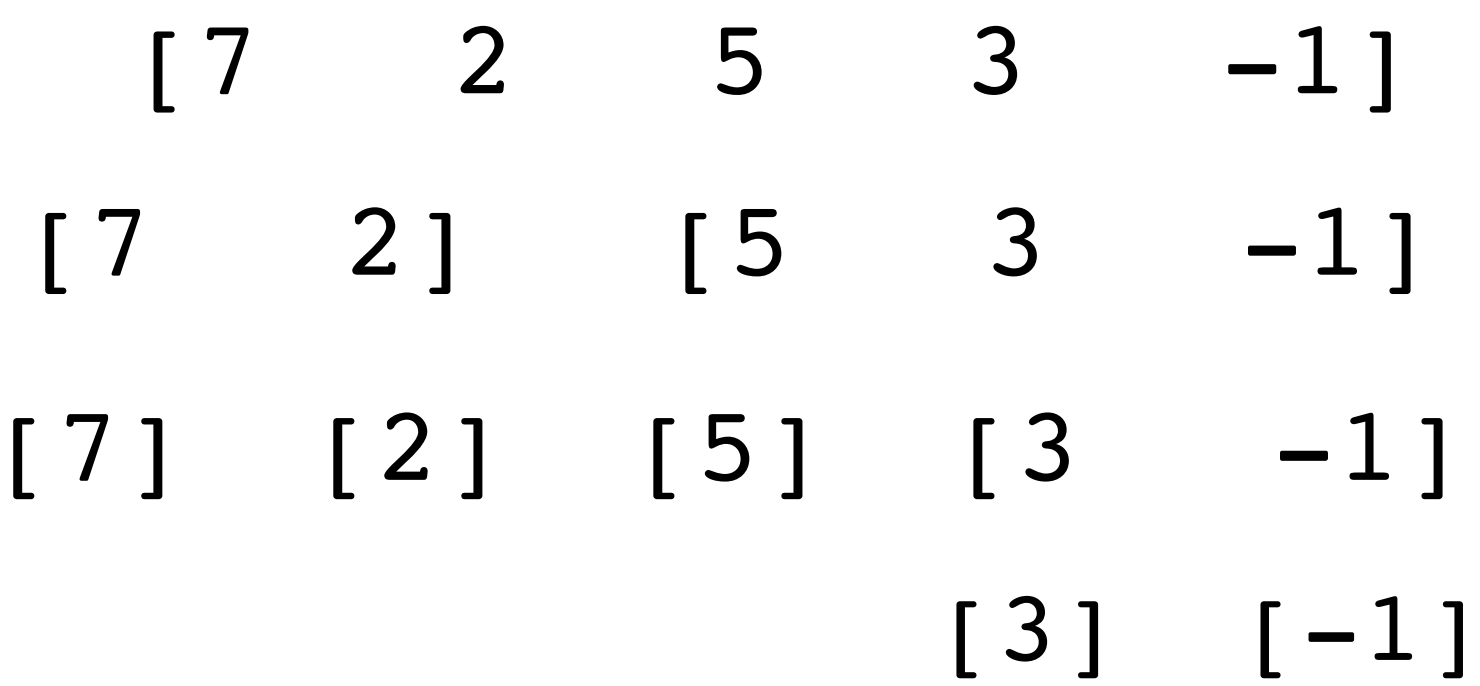
[7 2 5 3 -1]

Activity 1: Trace merge sort on the above example input with 5 elements
Draw the final binary tree representation of the trace.
What is the height of the resulting binary tree?



Running Time Analysis

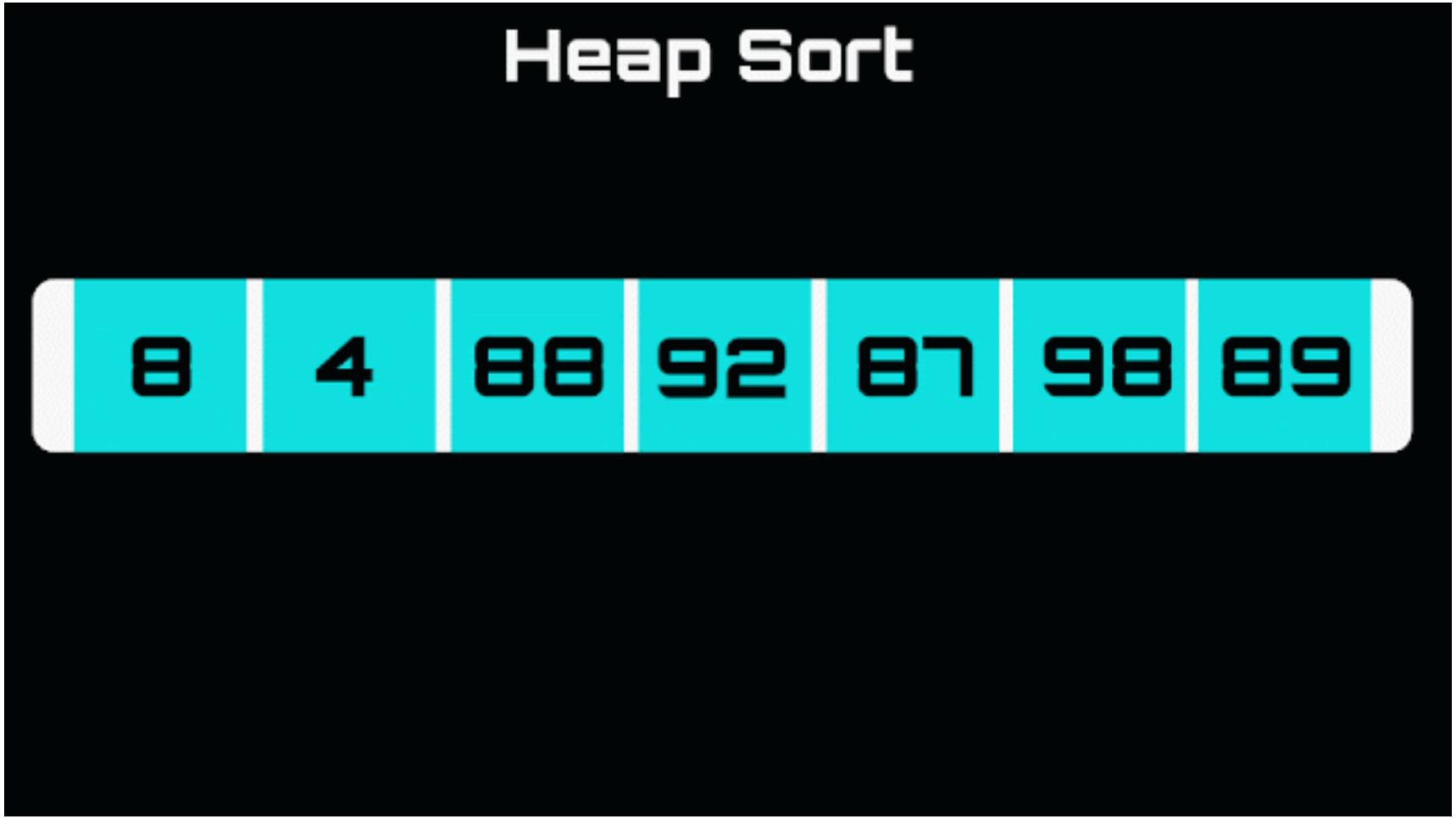
T(n) = # copy operations to split lists +
comparisons to merge lists +
function calls



Space Complexity Analysis

How much additional space is used by the time mergsort reaches the base case?

A. n* log(n)
B. n + n/2 + n/4 + n/8 +….+ 1
C. n + n/2 + n/4 + n/8 +….+ 1 + log(n)
D. Something else



Source: https://cosmictechie.hashnode.dev/heap-sort

HeapSort Algorithm

```
HeapSort(vector v)
    for i from n/2-1 down to 0:    // heapify: build max heap
      bubble_down(v, n, i)

    for i from n-1 down to 1:      // sort invariant:
      swap(v[0], v[i])                heap: v[0..i-1]
      bubble_down(v, i, 0)            sorted: v[i..n-1]
```

Heapsort: https://visualgo.net/en/heap

Why Mergesort over Heapsort?

- Same O(n log n) time, but heapsort has poor cache performance - why?
- Mergesort is **stable**: equal elements keep their original relative order
- Mergesort parallelizes naturally