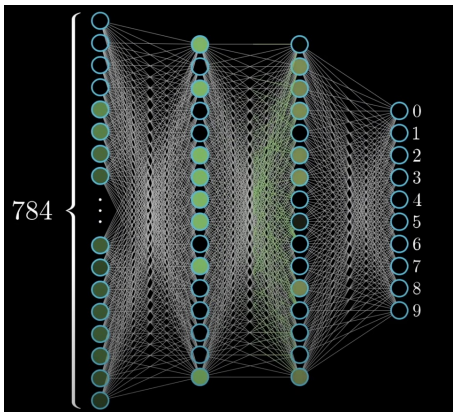
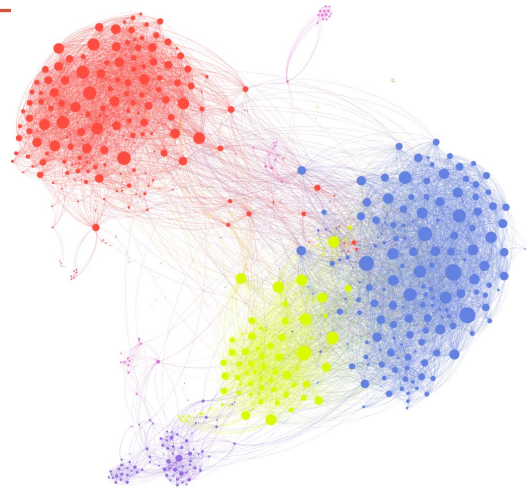
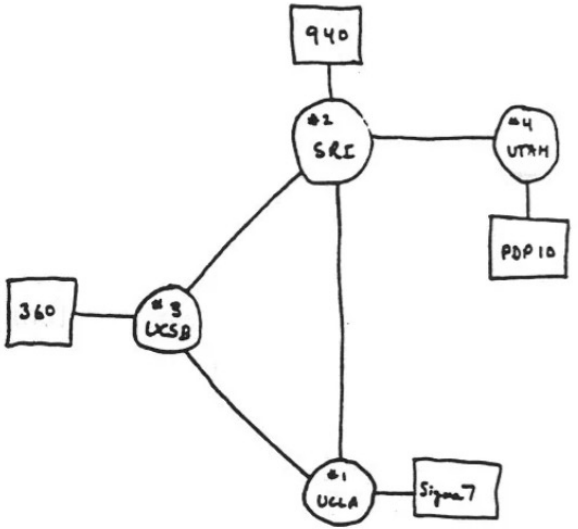
MERGE SORT


784
0
1
2
3
4
5
6
7
8
9




940
#2
SRI
#4
UTAH
PDP 10
360
#3
UCSB
#1
UCLA
Sigma 7
THE ARPA NETWORK
DEC 1969
4 NODES

Divide and Conquer Algorithms

Algorithm Approach:

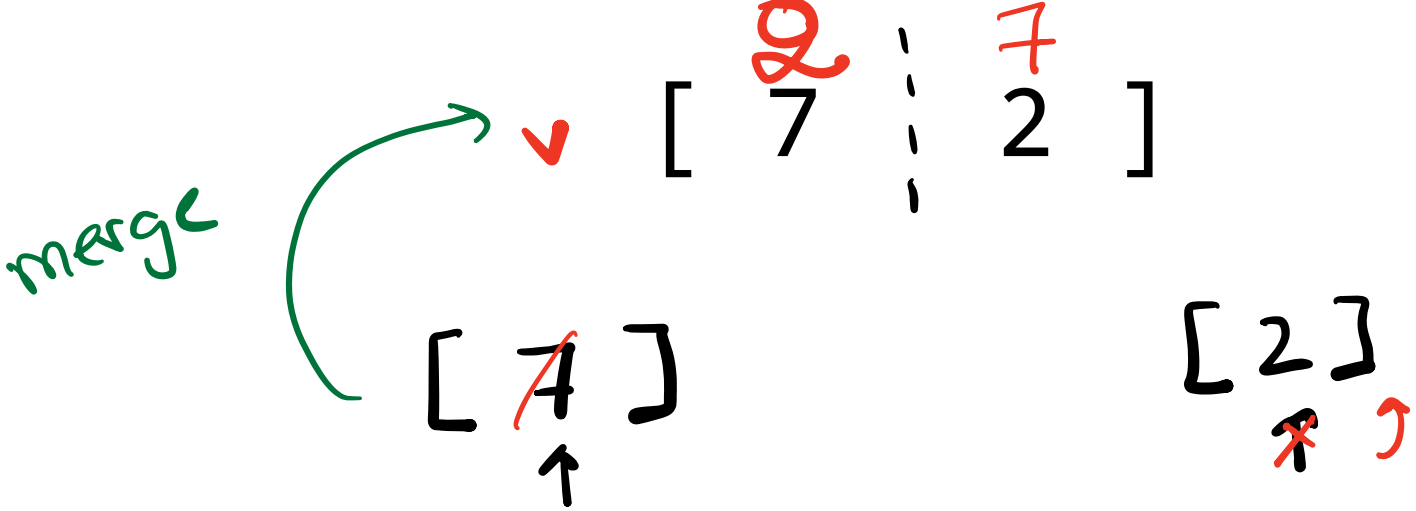
- **Divide** a large problem into sub-problems
- **Solve** each sub-problem
- **Combine** the solutions of sub-problems to obtain the solution for the original problem

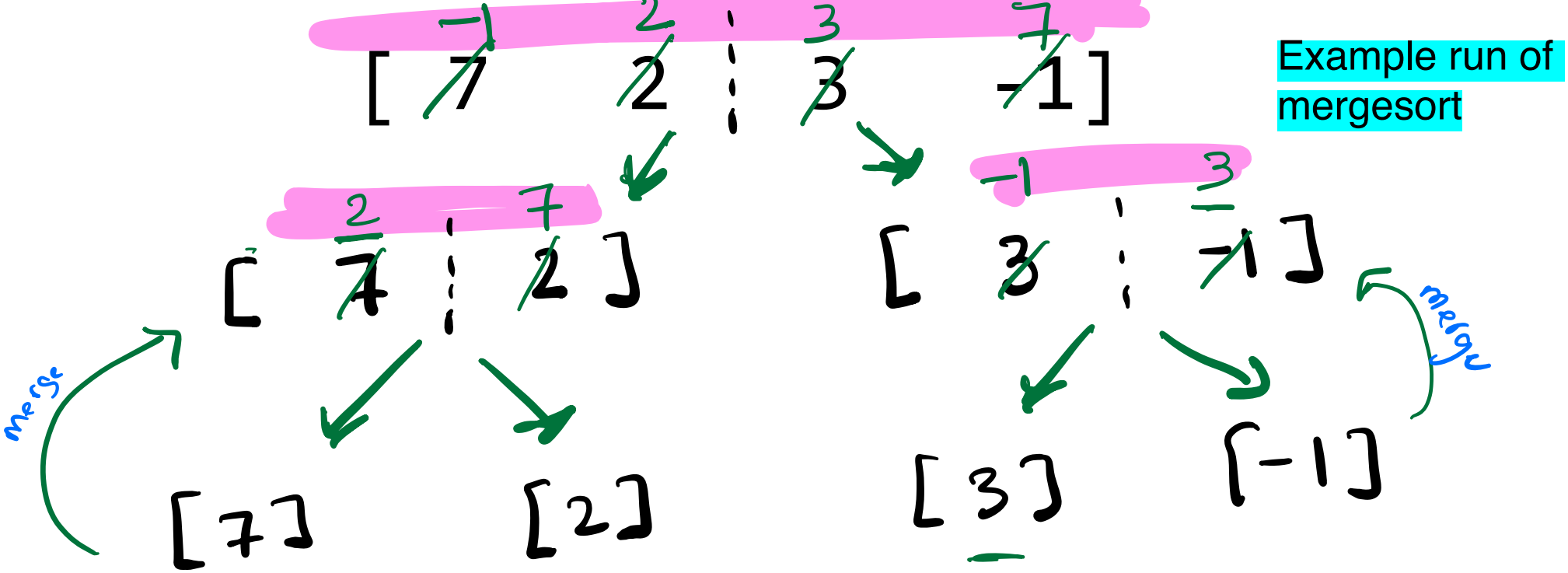
Merge Sort Algorithm

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



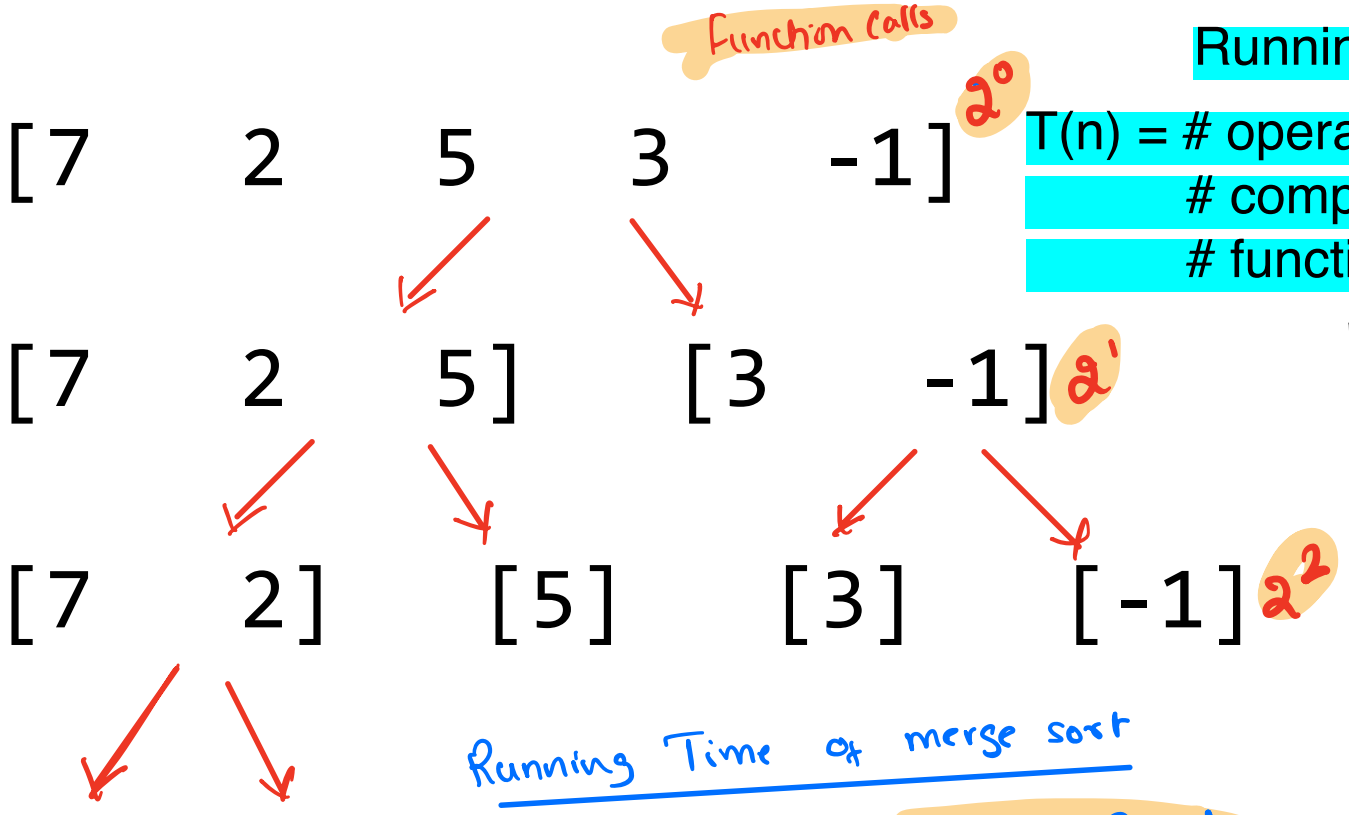

Example run of mergesort
merge
merge

Example run of mergesort

What is the maximum depth of the binary tree trace of mergeSort?

A. 1. B. 2 C. 3 D. 4 E. 5

Generalize the answer for an input vector of size n



Running Time Analysis

T(n) = # operations to split each list +
comparisions to merge lists +
function calls

At each level:
n elements copied
n comparisons to merge lists

# of function calls $\leq 2^{\log(n)+1} - 1$
$= 2n - 1$

Running Time of merge sort

$T(n) = \log(n) \cdot O(n) + 2n - 1$

- $\log(n)$: Max. number of levels
- $O(n)$: work done to copy & merge at each level
- $2n-1$: # of function calls

$T(n) = O(n \log n)$

n

[7 2 5 3 -1]

[7 2 5] [3 -1]

[7 2] [5] [3] [-1]

[7] [2]

Space Analysis

S(n) = $2n + \log n = O(n)$

Space used to perform copy until base case is reached

$n + \frac{n}{2} + \frac{n}{4} + \frac{n}{8} + \cdots \frac{n}{n}$

$= n\left(1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \cdots \frac{1}{n}\right) \leq 2n$

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

max depth of stack = max. space used for function calls

Path: a sequence of nodes in which each node is connected by an edge to the next.

p: 3 → 5 → 6

Ancestor(u): any node that is on a path ending in u

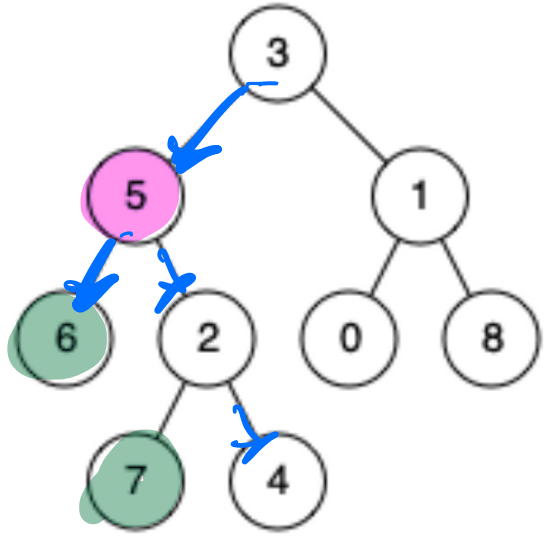
3 → 5 → 6 (Ancestors of 6)

Descendant(v): any node that is on a path starting from v

3, 5, 2, 4 (Descedents of 5)

Common ancestor(u, v): any node that is the ancestor of both u and v

Lowest Common ancestor(u, v): deepest node in the tree that is a common ancestor of u and v



https://leetcode.com/problems/lowest-common-ancestor-of-a-binary-tree/

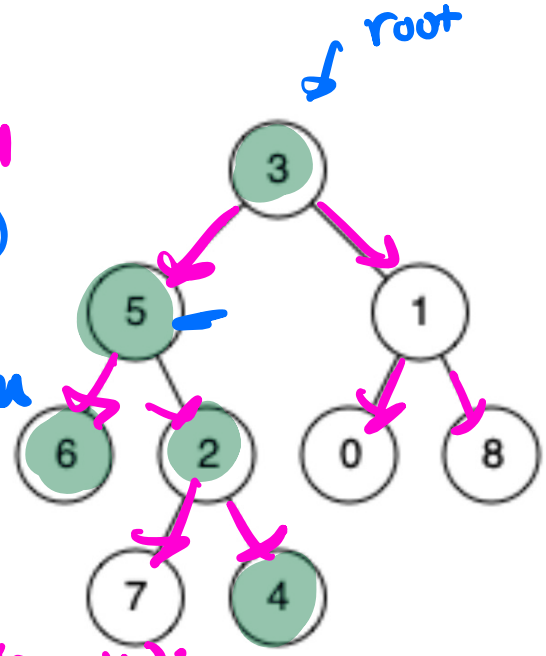
Approach 1: Turn definitions into an algorithm

(u, v)

① Find all ancestors of u path to u: 3 → 5 → 6

② Find all ancestors of v path to v: 3 → 5 → 2 → 4

③ Find lowest common ancestor by combining ① & ②

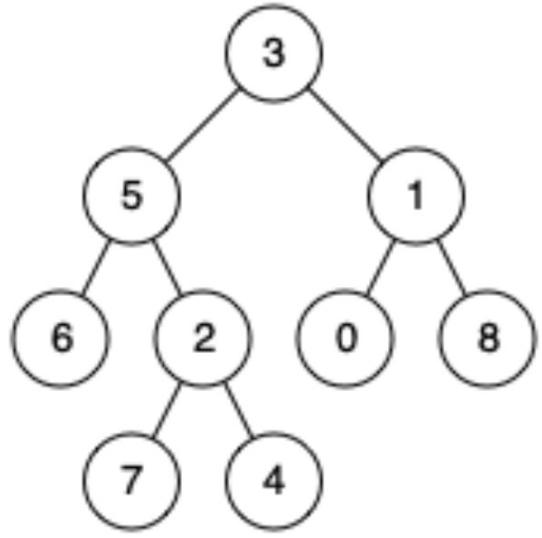



```
Algo findPath( Node *r, vector path, int u):
    if(!r) return
    path.push_back(r->val);
    if(r->val == u)
        return;
    if(r->left)
        findPath(r->left, path, u);
    if(r->right)
        findPath(r->right, path, u);
```

Path : 3, 5, 6, 2, 7, 4 (incorrect need to fix code see next lecture)

https://leetcode.com/problems/lowest-common-ancestor-of-a-binary-tree/

Approach 2: Divide and Conquer



https://leetcode.com/problems/lowest-common-ancestor-of-a-binary-tree/