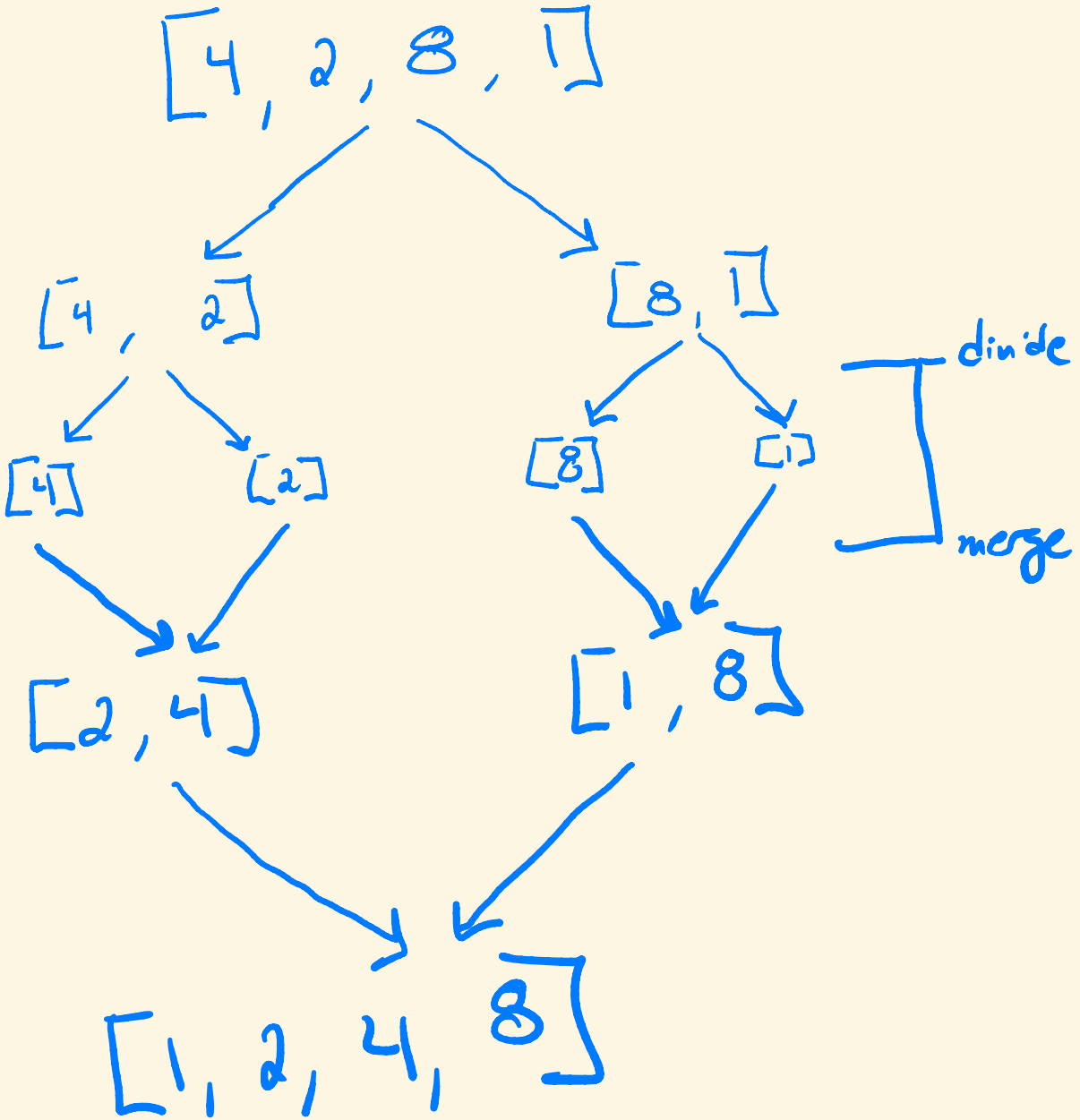
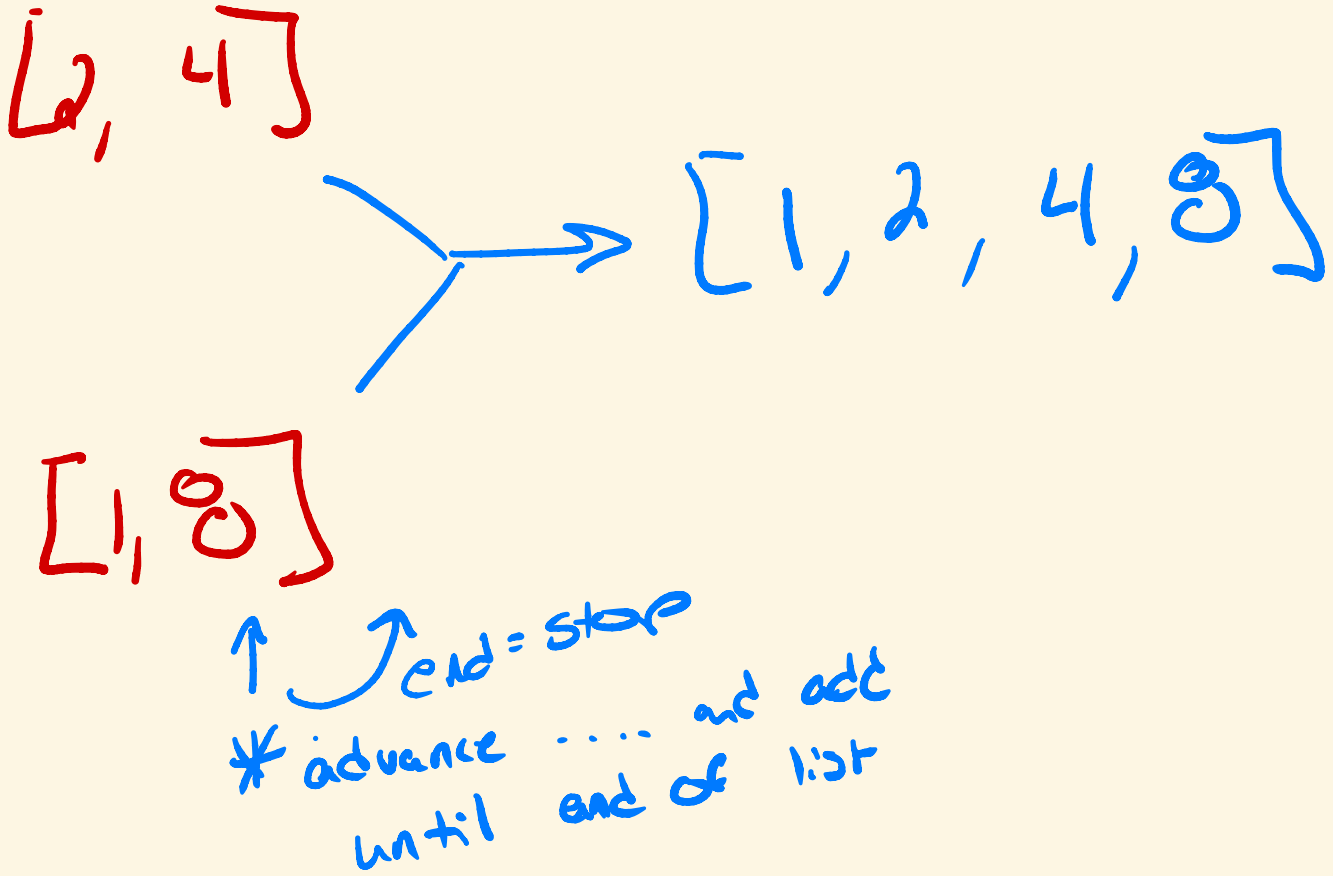
SORTING

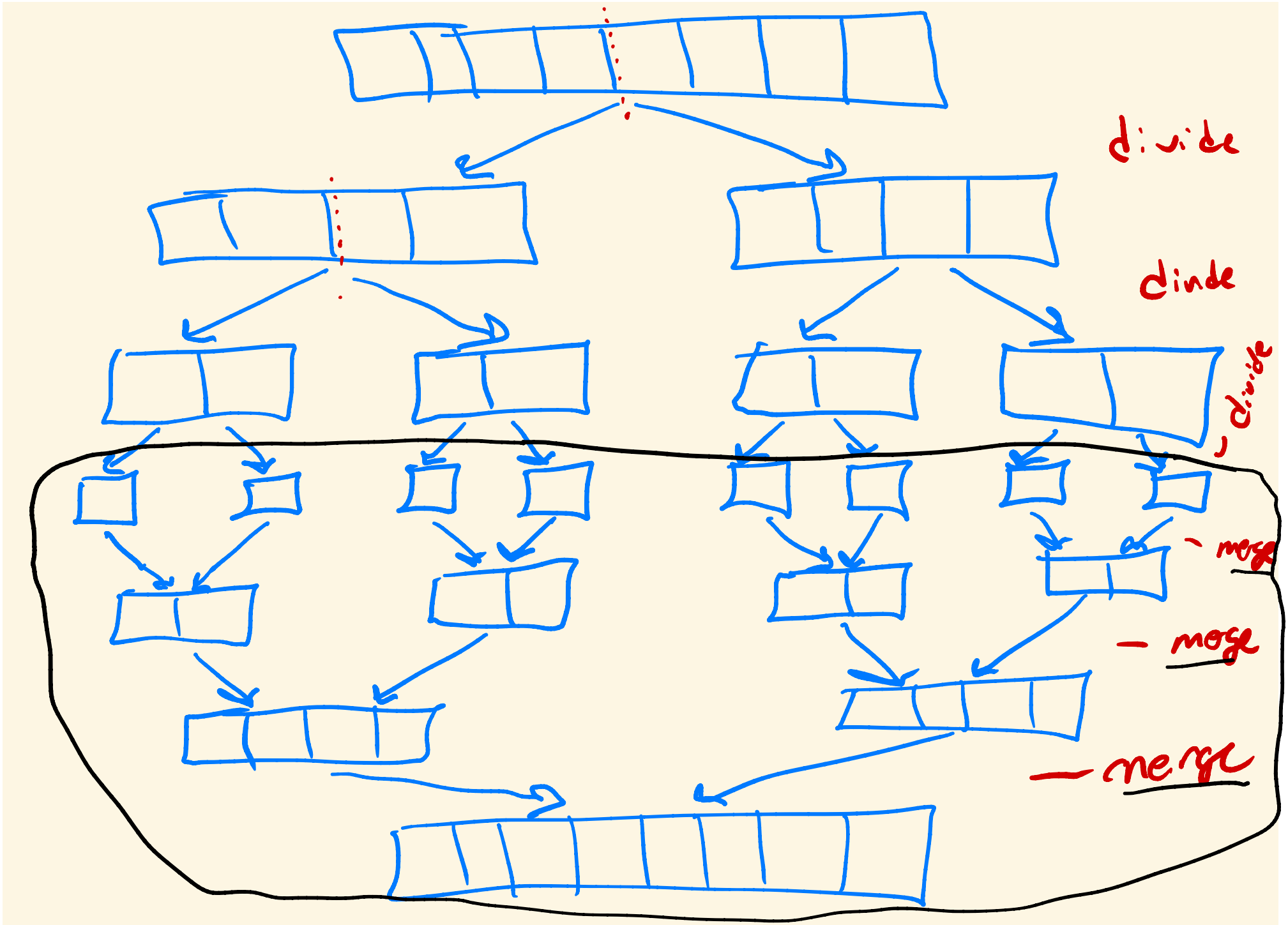
MERGE SORT

- divide and conquer algorithm
- recursive
- process:
  - divide array into 2 halves
  - **recursively** sort each half (by calling mergesort on each half)
  - merge sorted halves (take 2 sorted lists and combine into one sorted list)


[4, 2, 8, 1]
[4, 2]
[8, 1]
[4]
[2]
[8]
[1]
[2, 4]
[1, 8]
[1, 2, 4, 8]
divide
merge

How to merge sorted lists




divide
dinde
divide
merge
merge
merge

MERGE SORT - COMPLEXITY

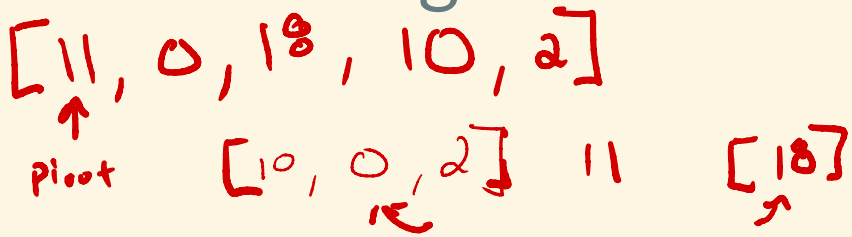
n = 8
3 merge steps
$\log_2(8) = 3$

- Tree like, recursive halving/combining $\Rightarrow \log(n)$
- How much work at each step of tree?
  - ↳ about n work

complexity $\Rightarrow n \log(n)$

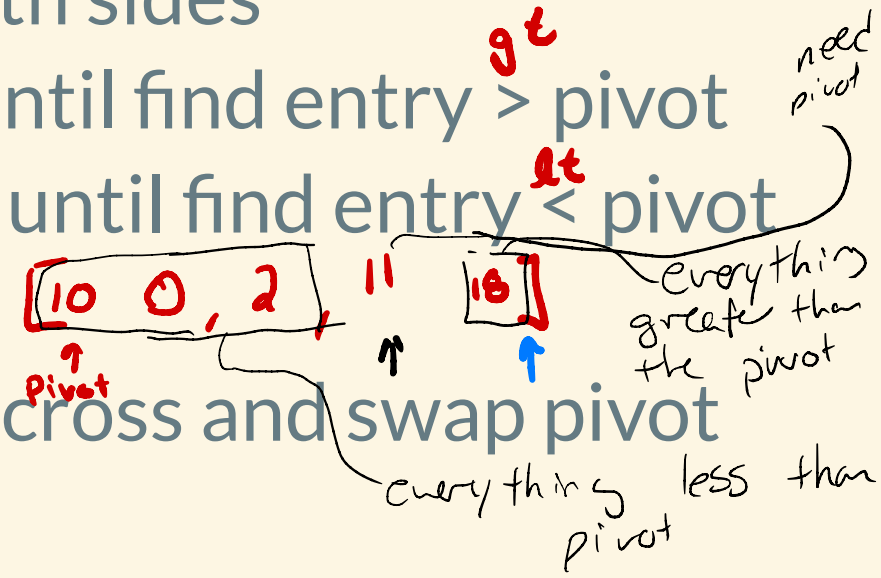
QUICK SORT

- divide and conquer algorithm
- recursive
- process:
  - choose pivot (often leftmost or rightmost element)
  - place:
    - everything less than pivot to it's left
    - everything greater than pivot to it's right
    - pivot in between
  - repeat process for left chunk and right chunk (use quicksort to sort each chunk)



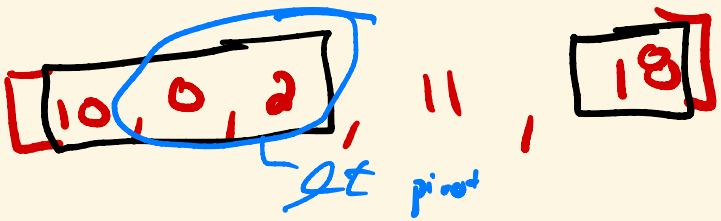
QUICK SORT

- How to move elements to correct side of pivot?
- Start scanning from both sides
  - Scan from left -> until find entry > pivot
  - Scan from <- right until find entry < pivot
  - Swap entries
- Stop once scan indices cross and swap pivot element into place
  - leftmost pivot, swap pivot with rightmost in left subarray
  - rightmost pivot, swap pivot with leftmost in right subarray



original [11, 0, 18, 10, 2]

after step 1

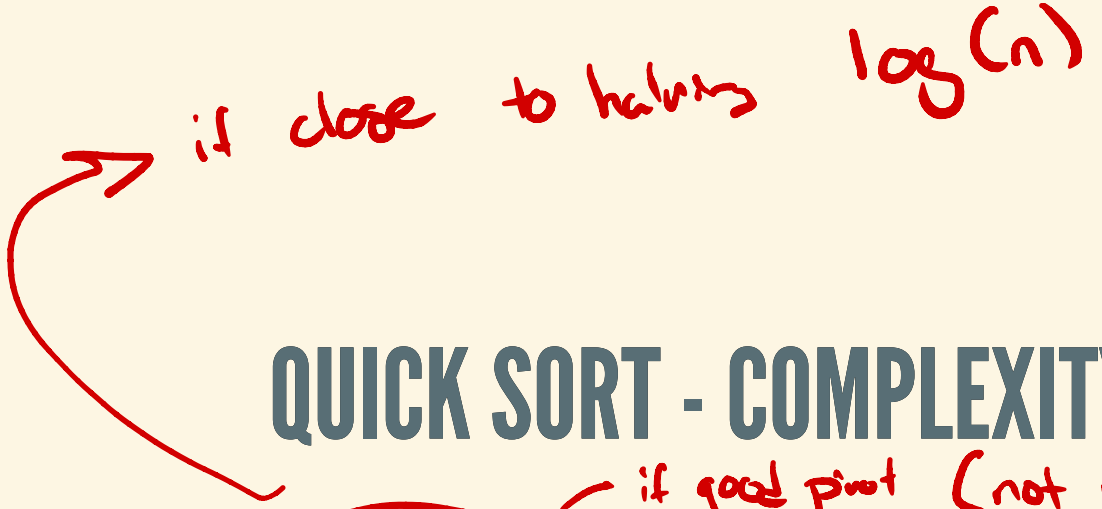


after step 2 [2, 0, 10, 11, 18]

pivot = 10

after step 3 [0, 2, 10, 11, 18]

QUICK SORT - COMPLEXITY



- Recursive halving of problems - how many levels?
- How much work at each level? ≈ n