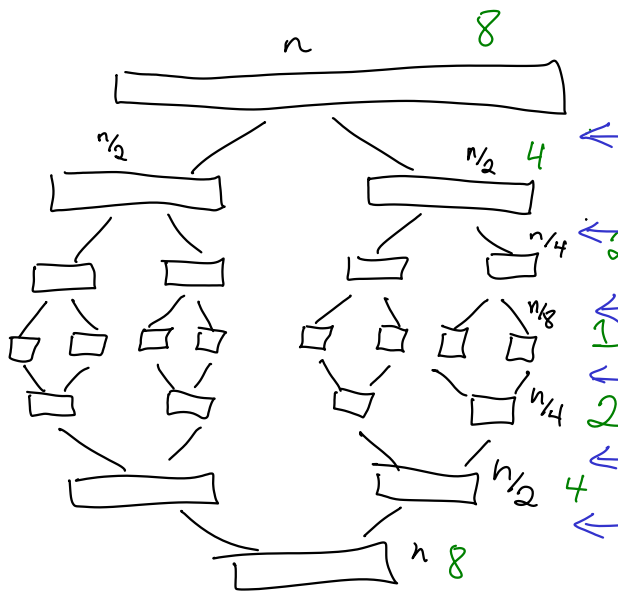
Merge Sort

- Split data into two piles
- Sort (recursively)
- Merge

Bubble sort
Insertion sort
Selection sort
$O(n^2)$

Recursive function

- Base case
- Inductive / recursive case (smaller problem)



$\leftarrow O(n)$
$\leftarrow O(n)$
$\leftarrow O(n)$
$\leftarrow O(n)$
$\leftarrow O(n)$
$\leftarrow O(n)$

$O(\log_2 n)$
$O(\log_2 n)$

$O(n)$ - Split data into two piles
- Sort (recursively)
$O(n)$ - Merge

# of levels = L

# of elements = n

$n \approx 2^L$

$\log_2 n \approx L$

$O(n \log n)$

Suppose $n = 1000000$.

$n^2 = 1000000000000$

$n \log_2 n := 20000000$

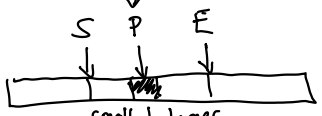
Quick Sort

- Recursive sort
  - Split array into two segments
    - s.t. all smaller things in 1st segment
      all bigger things in 2nd segment
  - recursively sort segments
- In-place (no extra memory*)

Lomuto

```
Partition(A, start, end)   <- returns "pivot index"   <- inclusive
    pivotIndex <- end
    pivot <- A[pivotIndex]
    fence <- start
    frontier <- start
    While frontier < end:
        If A[frontier] ≥ pivot:
            // (large element, so we're okay)
        Else:
            swap A[fence] w/ A[frontier]
            fence <- fence + 1
        End If
        frontier <- frontier + 1
    EndWhile
    Swap A[pivotIndex] w/ A[fence]
    Return fence
End Function
```

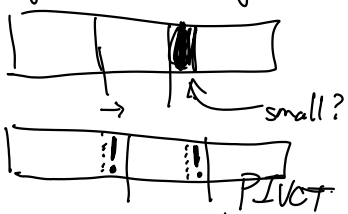
Goal



Dutch partition

ANY partition algorithm that does this will work

At each step, one more element is in the right place (on the correct side of pivot) starting at left and going to right.

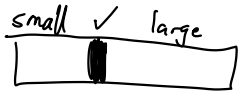


2 3 4 5 7 8

Hoare algorithm

Not discussing this one

```
Function QS(A, start, end):
    size <- end - start + 1
    If size < 2:
        Return
    pivotIndex <- partition(A, start, end)
    QS(A, start, pivotIndex - 1)
    QS(A, pivotIndex + 1, end)
End Function
```



QS complexity: worst case $O(n^2)$

Randomized QS

- At start of partition, pick a random index & swap w/ last index

	QS	RQS
worst case	$O(n^2)$	$O(n^2)$
expected worst case	$O(n^2)$	$O(n \log n)$

	bad user	bad luck
worst	✓	✓
expected worst	✓	~