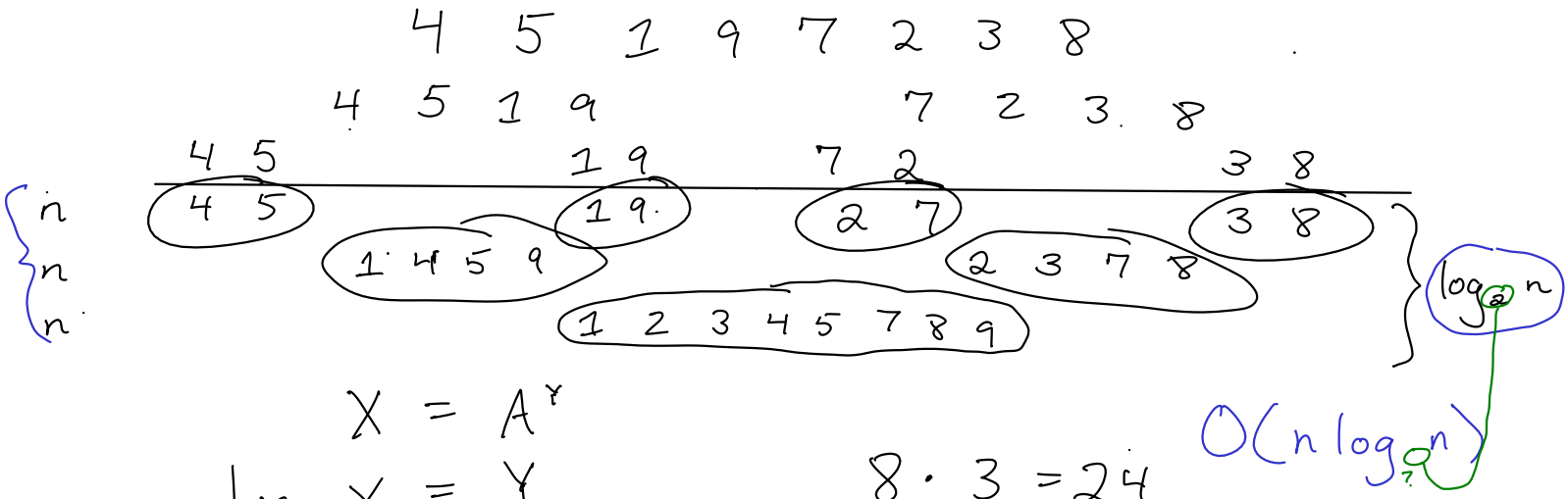
Merge Sort

Base case: array size ≤ 2

Recursive case: split the array, sort the parts, merge result



$X = A^Y$

$\log_A X = Y$

$8 \cdot 3 = 24$

$16 \cdot 4 = 64$

$O(\log_a n) = O(\log_b n)$

$1000\, n \log_2 n$

$1000 \cdot 1000000 \cdot \log_2 1000000 < 1000000000 \cdot 20$

$n^2 = 1000000^2 =$ a lot more

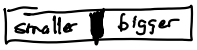
Merge Sort is $O(n \log n)$ time

needs $O(n)$ "heap" space

Bubble Sort
Sel
Ins

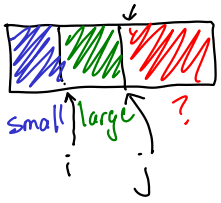
"in-place"

Fast sorting algorithm, in-place: Quick Sort



inclusive index

Function Partition(A, start, end)



1 3 4 | 7 8 6 || (5)

small | i large | j

Postcondition: array split into
small region (left)
large region (right)
pivot index between is returned

3 5 2 1 7 4

3 1 2 (4) 7 5

small | pivot | large

Today: Lomuto

Alt: Hoare

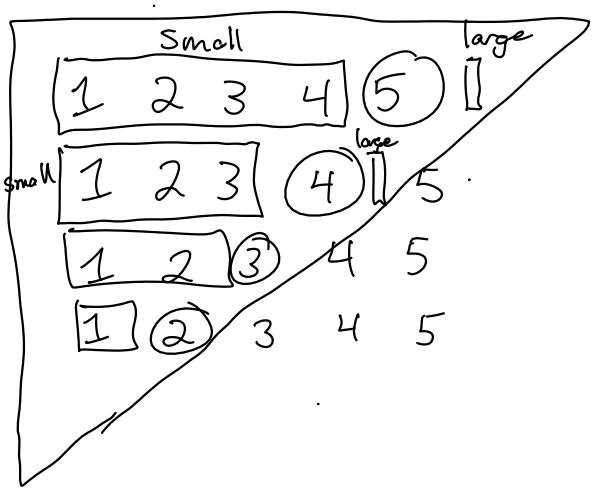
```
Function Partition(A, start, end)
    i ← start
    j ← start
    pivot ← A[end]
    While j < end:
        j ← j + 1
        If A[j-1] < pivot:
            Swap A[j-1] with A[i]
            i ← i + 1
        End If
    End While
    Swap A[i] with A[end]
    Return i
End Function
```

1 3 4 (5) 7 6 8

```
Function QuickSort(A, start, end)
    If start < end:
        pivotIndex ← Partition(A, start, end)
        QuickSort(A, start, pivotIndex - 1)
        QuickSort(A, pivotIndex + 1, end)
    End If
End Function
```

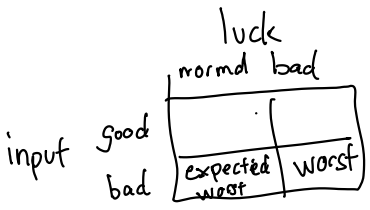
Quick Sort: Worst-case of $O(n^2)$

$O(n^2)$



Gambler's Sort:

Flip a coin
If heads: array is magically sorted
Else: recurse

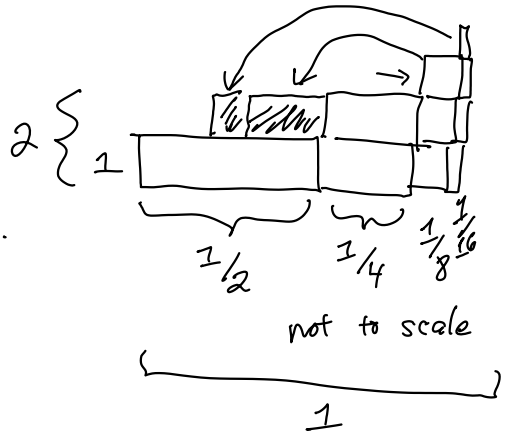


Worst-case: $O(\infty)$

Expected Worst-case: $O(1)$

$$\frac{1}{2} \cdot 1 + \frac{1}{2^2} \cdot 2 + \frac{1}{2^3} \cdot 3 + \cdots$$

$$\sum_{i=1}^{\infty} \frac{i}{2^i} = 1 \text{ or } 2 \text{ ?}$$



Randomized Quick Sort: WC $O(n^2)$

EWC $O(n \log n)$

Function Partition(A, start, end)

i ← start

j ← start

swap A[end] with A[randomly chosen]

pivot ← A[end]