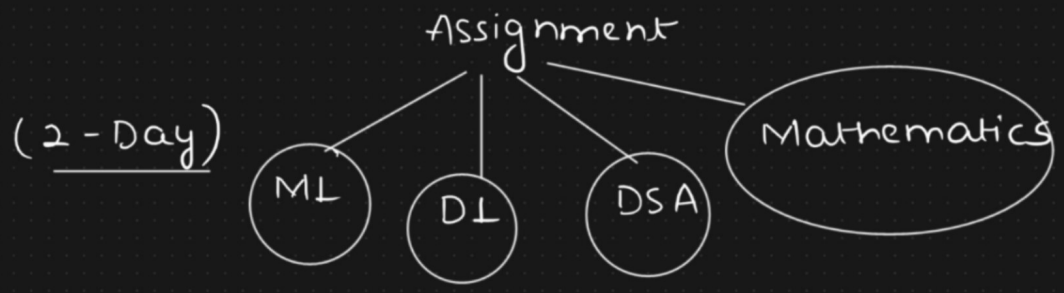
Divide & conquer



Sort the array

↳ n == 1

↳ Small problem

↳ return arr(i)

n > 1

↳ big Problem

↳ Divide & conquer approach

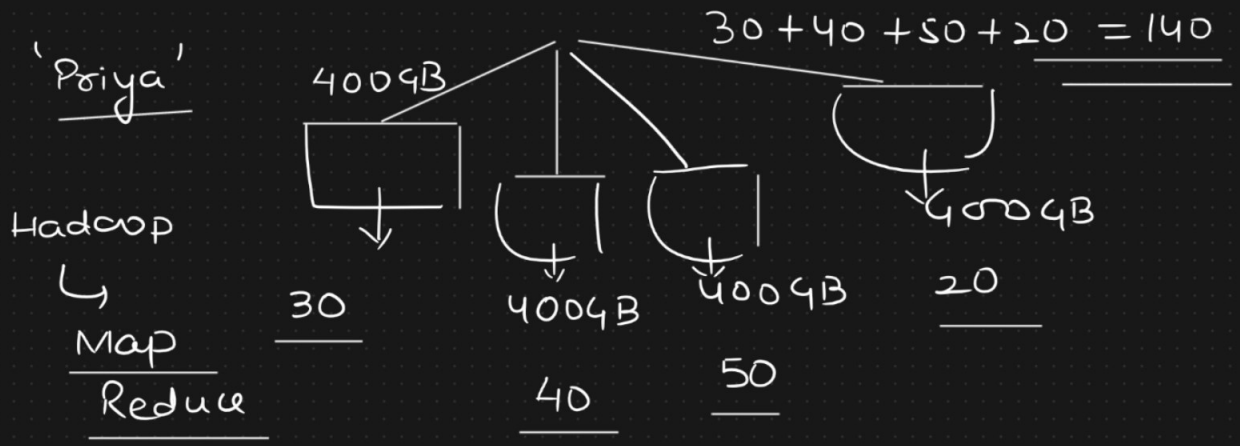
1) Divide the problem into various subproblems
2) conquer each subproblem
3) combine all the solutions (optional)

a) Data Science

↳ b) Data Engineering — **Big Data**

c) frontend developer

d) none of these



Psuedocode

```
divideAndconquer(arr, p, q):
    if (small(arr,p,q)):
        return solution
    else:                          ← two parts
Divide ——— m = Divide(arr,p,q)
                                   Recursion
Conquer —{ b = divideAndconquer(arr,p,m)
         { c = divideAndconquer(arr,m+1,
                                q)
Combine  return combine(b,c)
```

arr = [75, 45, 95, 50, 60, 67, 29, 32]

0 1 2 3 4 5 6 7

max = 95

min = 29

n == 2

↳ arr(i) < arr(j)

max = arr(i)

min = arr(i)

vice-versa

max = ~~75~~ 95

min = ~~75~~ ~~45~~ 29

↳ O(n)

n == 1

↳ max =

min =

arr(i)

⟼ complete Binary Tree

mid = i + (j-i)//2

Recursive tree

C1

arr, i=0, j=7, min=29, max=95

mid = 3

C2

arr, 0, 3, 45, 95

C6

arr, 4, 7, 29, 67

3-0+1 = 4

mid = 1

C3

arr, 0, 1, 45 (min1), 75 (max1)

C5

arr, 2, 3, 50 (min2), 95 (max2)

C7

arr, 4, 5, 60, 67

C8

arr, 6, 7, 29, 31

75, 45

↳ single comparison

95, 50

60, 67

29, 32

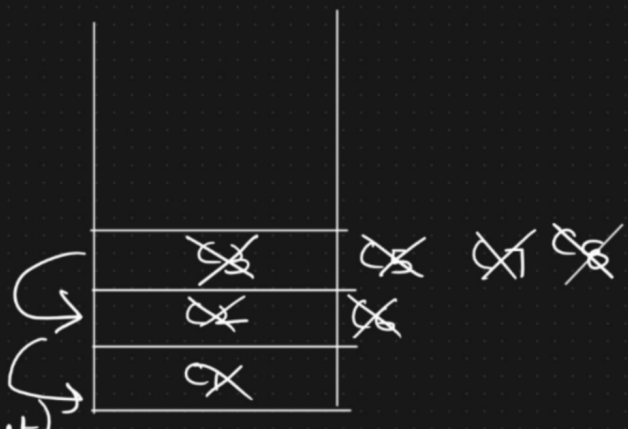
$K = \log_2(n+1)$

$O(\log n)$

↓ stack space

(Last In First Out)

LIFO



Stack

⇐ Data Structure

Psuedocode

$T(n)$

```
findMaxAndMin(arr, i, J):
    if i == J:                      # single element
        min = arr(i)
        max = arr(i)
    elif i == J-1:                  # two element
        if arr(i) < arr(J):
            max = arr(J)
            min = arr(i)
        else
            max = arr(i)
            min = arr(J)
    else:
        mid = i + (J-i)//2          # Divide  ← c
        min1, max1 = findmaxAndmin(arr, i, mid)      # ↳ T(n/2)
        min2, max2 = findmaxAndmin(arr, mid+1, J)    # ↳ T(n/2)
        if min1 < min2:             # combine ↳ c
            min = min1
        else:
            min = min2
```

i → starting index

J → ending index

i, J

20
0

0	1
20	40

i = 0

J = 1

Small Problem ↳ c

Conquer ↳ $2T(n/2)$

```
        if max1 < max2:
            max = max2
        else:
            max = max1
    return (max, min)
```

Recurrence Relation:

$$T(n) = \begin{cases} c & n \leq 2 \\ 2T(n/2) + c & n > 2 \end{cases}$$

$$T(n) = 2T(n/2) + c$$

$a = 2 \quad k = 0$

$b = 2 \quad p = 0$

$$\log_b a = \log_2^2 = 1$$

$$\log_b^a > k \longrightarrow O(n^{\log_b^a})$$

$$\rightarrow O(n)$$

Brute force approach

↳ O(n)

Divide & conquer approach

↳ O(n)

↳ Recursion

Stack space

= O(log n)