Design and Analysis of Algorithms

Lecture - 10

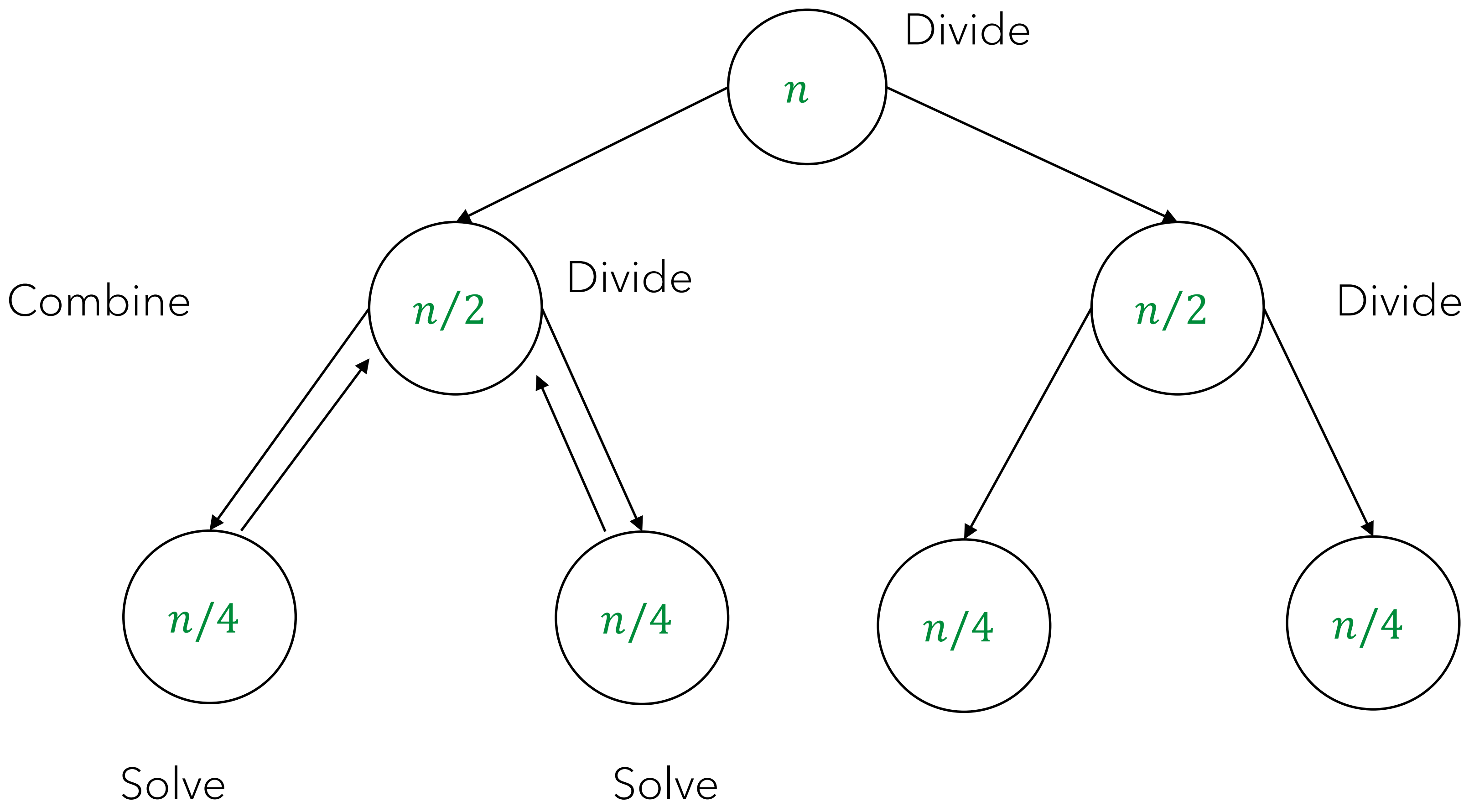
Success is always inevitable with Hard Work and Perseverance

N. Ravitha Rajalakshmi

Learning Objective

- Discuss D&C strategy for classical problems
  - Find Maximum and Minimum element in the array

D&C ~ Recursion



Code Structure

Function Recursive (n)

 # n – indicative of input size

```
if (test on input size(n)){
        Solve
}
else {
        Compute
        Divide
        Conquer(Compute)
}
```

Base Case

Recursive Case

Recursive call where problem instance is reduced

Find Max & Min in an Array

Input: An array A with n elements.

Output: Find minimum and maximum element in array

Index	0	1	2	3	4	5
Array Elements	130	10	40	8	20	200

Naïve Algorithm

- Iterate through every element in the array and track the maximum and minimum value

Index	0	1	2	3	4	5
Array Elements	130	10	40	8	20	200

Function MinMax(A, n)

```
# n – indicative of  array size
Min = A[0]
Max = A[0]
for(i=1;i<n;i++){
        if(A[i]> Max){
                Max = A[i]
        }
        else{
                if(A[i]<Min)
                        Min = A[i]
        }

return (Min, Max)
```

Time Complexity

Input Size : n

Basic Operation : Comparison

Does Complexity changes with specifics of input ? No

2(n-1)
(Two comparisons are made in the else case)

D&C Strategy

- Can we split the array and do comparisons in parallel?

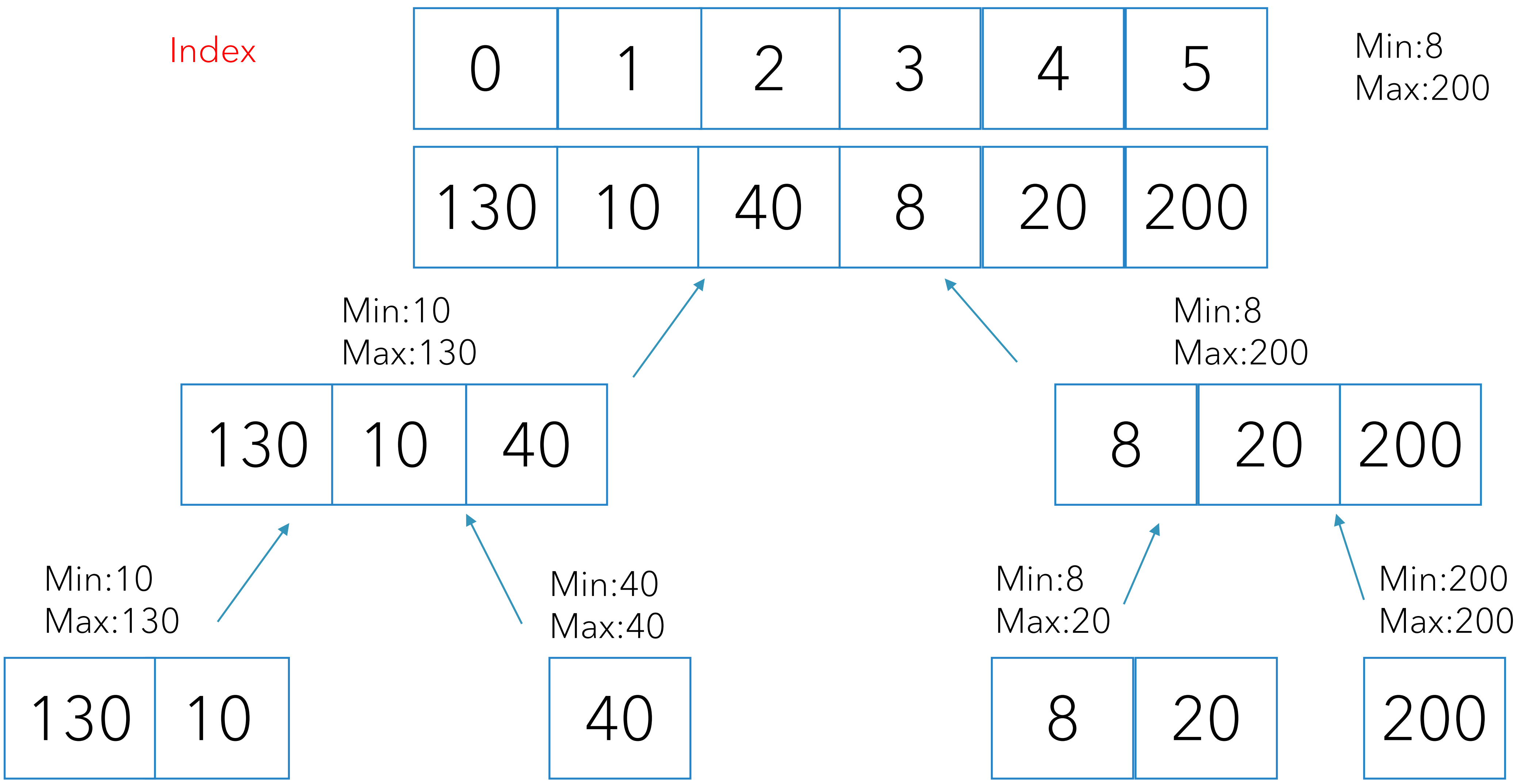
  Yes

- Will it affect the final answer ?

  No

- Is there a mechanism for combining the independent solutions?

  Yes


Index
0
1
2
3
4
5
Min:8
Max:200
130
10
40
8
20
200
Min:10
Max:130
Min:8
Max:200
130
10
40
8
20
200
Min:10
Max:130
Min:40
Max:40
Min:8
Max:20
Min:200
Max:200
130
10
40
8
20
200

Function MinMaxD&C(A, low, high)

```
# n – indicative of  array size
        # Only one element in array
        if (low==high){
                min, max <-  A[low] ,  A[low]
        }
        # Two elements in array
        if(low – high==1){

                min <-  smaller(A[low], A[high])
                max <-  larger(A[low], A[high])
        }
```

```
# more than two elements
else{
        #Dividing the Problem
        mid = (low + high) /2
        min1, max1 <-  MinMaxD&C(A, low, mid)
        min2, max2 <-  MinMaxD&C(A,mid+1, high)

        #Combine Solutions
        min <-  Smaller(min1, min2)
        max <-  larger(max1, max2)
}

return (min, max)
}
```

Analyzing Time Complexity

- Recurrence Relation
- Input size : n
- Basic Operation : Comparison
- Time Complexity : $T(n)$

$$T(1) = 0 \qquad\qquad T(2) = 1$$

$$T(\mathrm{n}) = 2T\left(\frac{n}{2}\right) + 2$$

- Using Master's theorem , $T(n) = O(n)$

Iteration Method

$$T(n) = 2T\left({}^{n}/_{2}\right) + 2$$

$$= \ 2\left[2T\left({}^{n}/_{4}\right) + 2\right] + 2$$

$$= 4T\left({}^{n}/_{4}\right) + 4 + 2$$

$$= 4\left[2T\left({}^{n}/_{8}\right) + 2\right] + 4 + 2$$

$$= 8T\left({}^{n}/_{8}\right) + 8 + 4 + 2$$

$$T(\mathrm{n}) = 2T\left(\frac{n}{2}\right) + 2$$

$$T(2) = 1$$

Characteristic Equation :

$$2^k T\left(\frac{n}{2^k}\right) + 2^k + (2)^{k-1} + \cdots 2$$

Solve for base case $\frac{n}{2^k} = 2$

$$n = (2)^k \cdot 2$$

$$n = 2^{k+1}$$

$$\log n = k + 1$$

$$k = \log n - 1$$

Substitute value of k in characteristic equation

$$2^k T\left(\frac{n}{2^k}\right) + 2^k + (2)^{k-1} + \cdots 2$$

$$2^{\log n - 1} + (2)^{\log n - 1} + (2)^{\log n - 2} \cdots 2$$

$$(2)^{\log n}(2)^{-1} + [2 + 4 + 8 \cdots (2)^{log\, n\, -1}]$$

$$\frac{n}{2} + 2\left[\frac{(2)^{\log n - 1} - 1}{2 - 1}\right]$$

$$\frac{n}{2} + 2\left[\frac{n}{2} - 1\right] = \frac{3n}{2} - 2$$

Summary

- How D&C is applied for finding min and max efficiently

Thank You
Happy Learning

Success is always inevitable with Hard Work and Perseverance