Trees, FindMax & FindMode: Query Versions

Giri Narasimhan

Programming Team

Fall 2024

FindMax: Three Versions

FindMax-Basic

- **Input**: Array A of size n
- **Output**: Find largest in A
- Naïve Iterative solution
- **Time Complexity**?
  - Naïve solution: $O(n)$
  - It is **optimal**

FindMax-Query

Fixed

- **Input**: Array A of size n
- **Query**: $1 \leq i \leq j \leq n$
  - # queries, k, may be large
- **Output**: Find largest in $A[i..j]$
- **Time Complexity per query**?
  - Naïve solution: $O(n)$
  - Improved solution: $O(1)$

	FindMax-Basic	FindMax-Query-v1	FindMax-Query-v2
Preprocessing Time	-	$O(1)$	$O(n^2)$
Preprocessing Space	-	$O(1)$	$O(n^2)$
Query Time (per query)	-	$O(n)$	$O(1)$
Total Time	$O(n)$	$O(kn)$	$O(k + n^2)$

Example

Input array A

1	2	3	4	5	6	7	8	9
14	41	19	2	23	11	29	6	31

Preprocessed 2D array B

1	2	3	4	5	6	7	8	9
14	41	41	41	41	41	41	41	41
	41	41	41	41	41	41	41	41
		19	19	23	23	29	29	31
			2	23	23	29	29	31
				23	23	29	29	31
					11	29	29	31
						29	29	31
							6	31
								31

Preprocessing Algorithm

How to fill in 2D array B

For $p = 1$ to n do

  For $q = 1$ to n do

    Compute $B[p, q]$

Time Complexity

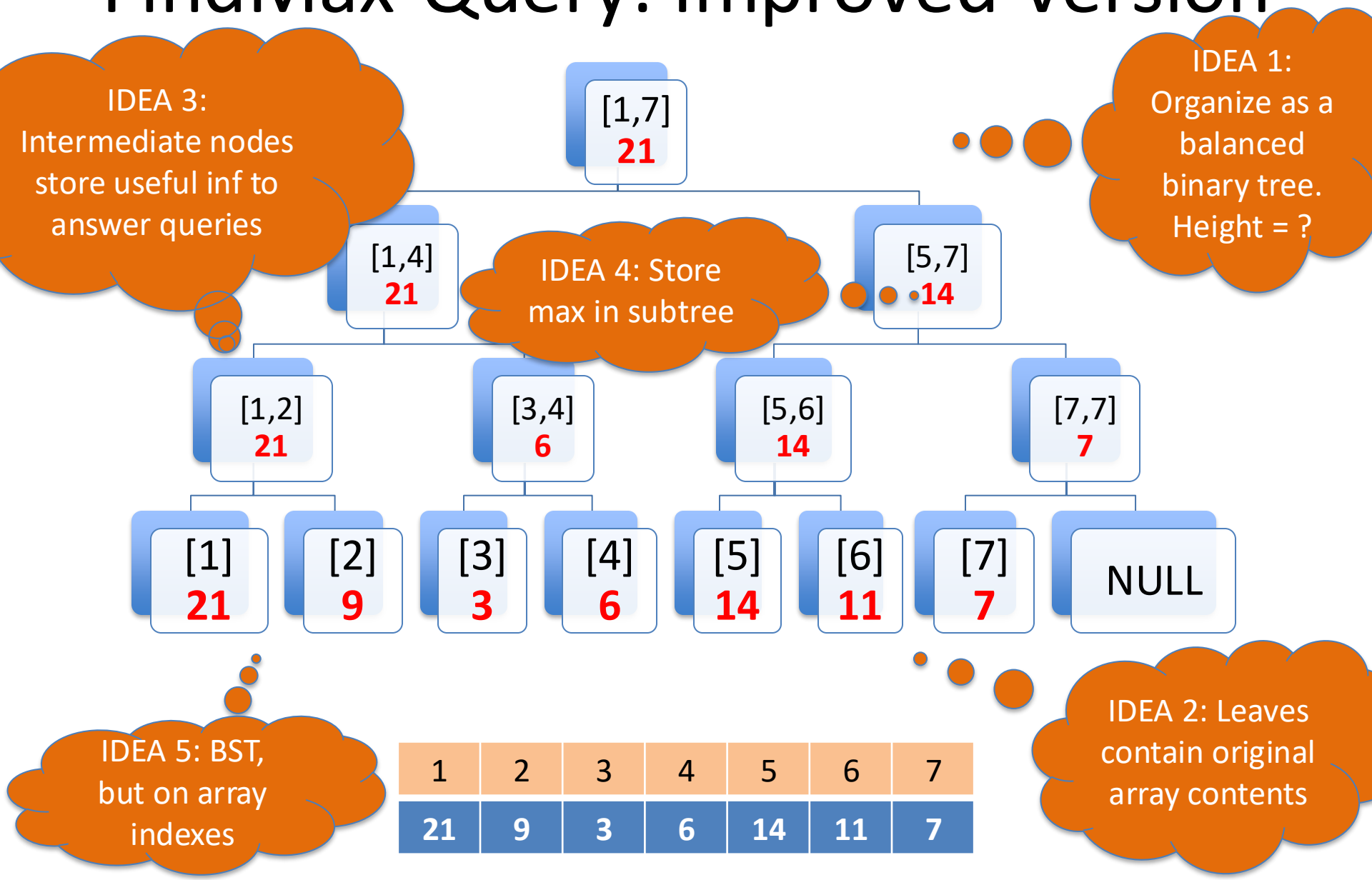
- Preprocessing Time to "Compute $B[p, q]$"?
  - Naïve: $O(n^3)$
- Time for k queries:
  - $O(k + n^3)$
- Improved preprocessing to "Compute $B[p, q]$" in $O(1)$ time
  - $B[p, q] = \max\{B[p, q-1], A[q]\}$
- Time for k queries:
  - $O(k + n^2)$

FindMax-Query

	Basic	Query-v1	Query-v2	Query-v3 (Tree)
Preprocessing Time	-	$O(1)$	$O(n^2)$	
Preprocessing Space	-	$O(1)$	$O(n^2)$	
Query Time (per query)	-	$O(n)$	$O(1)$	
Total Time	$O(n)$	$O(kn)$	$O(k+n^2)$	

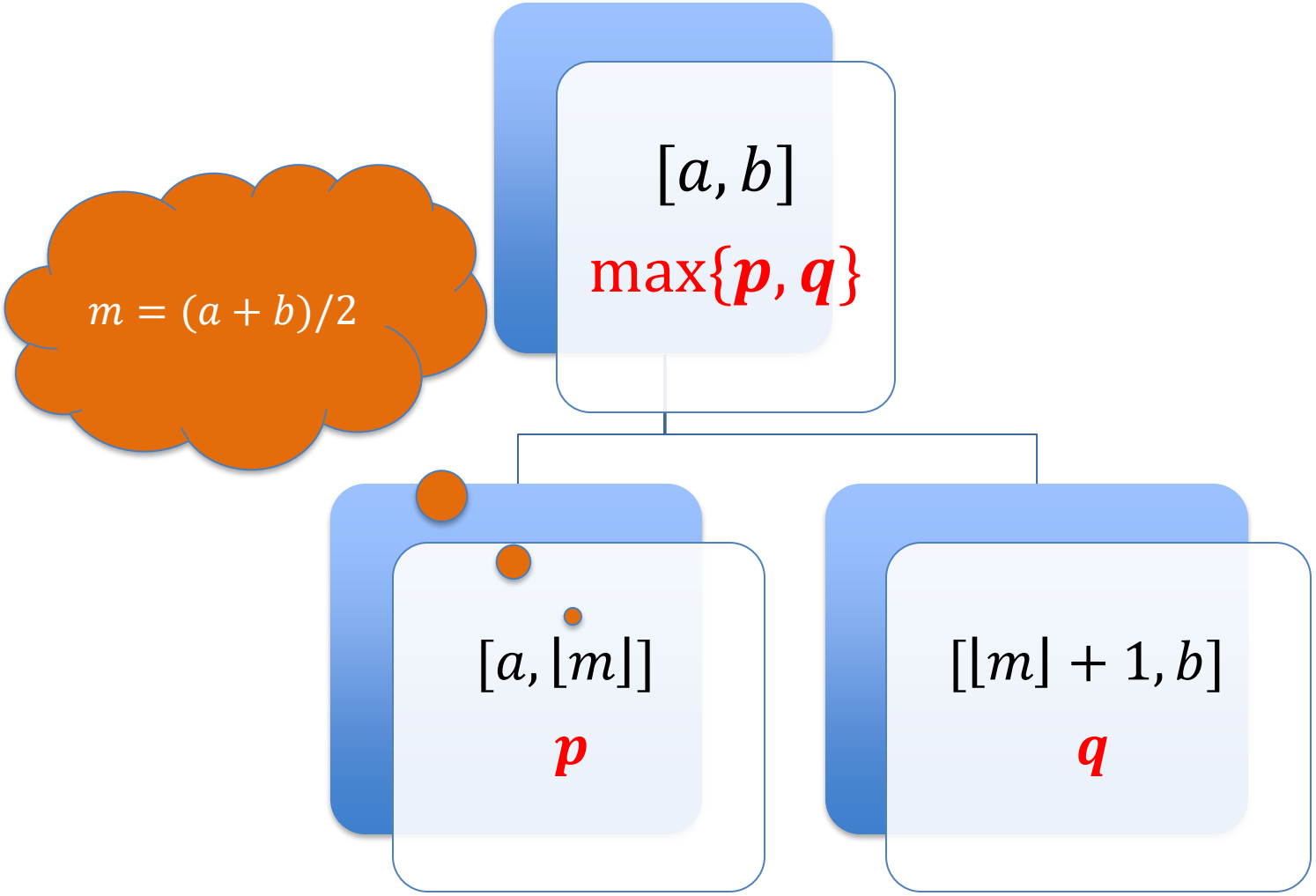
How about a better tradeoff between Preprocessing time and Query time?

FindMax-Query: Improved Version



1	2	3	4	5	6	7
21	9	3	6	14	11	7

Additional Node Info: MAX



How to answer FindMax-Query(i, j)

- Always start at the root and then …

FindMax-Query(i, j, u)

- If $(u.\text{Interval} == [i, j])$ then

  Report $u.\text{MaxValue}$

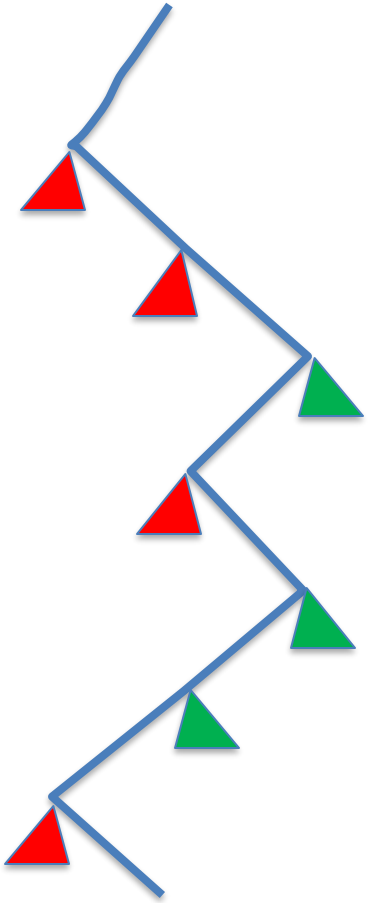
- Else if $(\text{u.Interval.Rt} == [i, k])\&(k < j)$ then

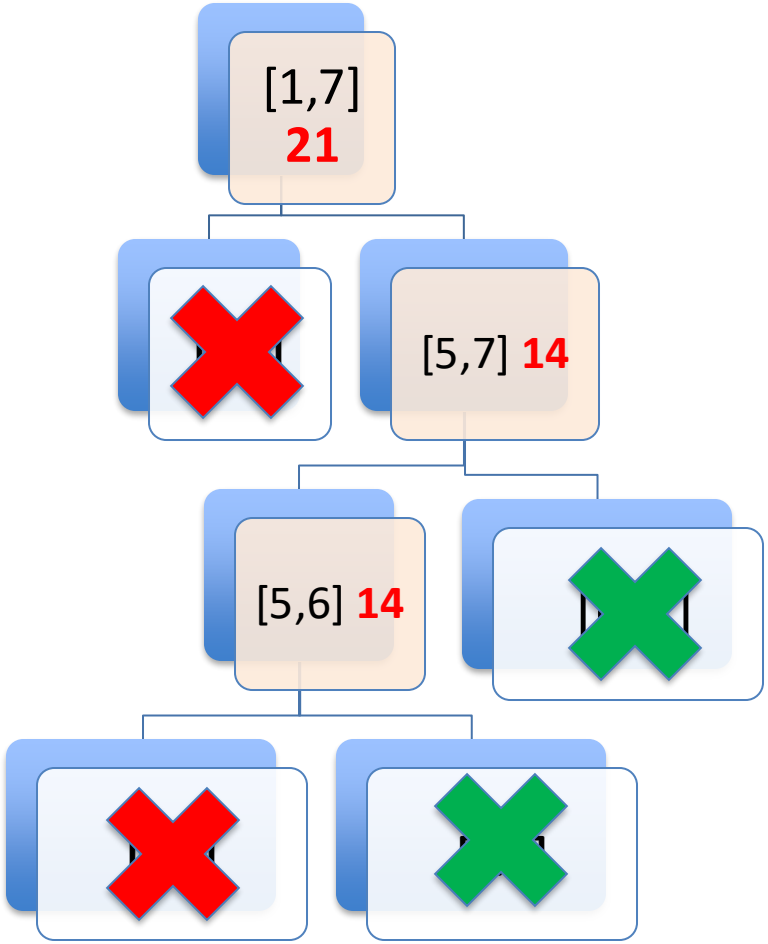
  FindMax-Query$(i, j, u.\text{LeftChild})$

- Else if $(\text{u.Interval.Lt} == [k, j])\&(k > j)$ then

  FindMax-Query $(i, j, u.\text{RightChild})$

Search Path in Tree





Search for range [5,6]

FindMax-Query

	Basic	Query-v1	Query-v2	Query-v3 (Tree)
Preprocessing Time	-	$O(1)$	$O(n^2)$	$O(n)$
Preprocessing Space	-	$O(1)$	$O(n^2)$	$O(n)$
Query Time (per query)	-	$O(n)$	$O(1)$	$O(\log n)$
Total Time	$O(n)$	$O(kn)$	$O(k+n^2)$	$O(k \log n + n)$

How about a better tradeoff between Preprocessing time and Query time?

Sorted array with repeats

FindMode: Find most frequent item (mode)

Time Complexity: O(n)

FindMode in range [i,j]

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	**1**	**3**	**3**	**3**	**3**	**6**	**9**	**9**	**9**	**15**	**24**	**24**	**24**	**39**

FindMode: Find most frequent item (mode)

	Basic	Query-v1	Query-v2	
Preprocessing Time	-	$O(1)$	$O(n^2)$	
Preprocessing Space	-	$O(1)$	$O(n^2)$	
Query Time (per query)	-	$O(n)$	$O(1)$	
Total Time	$O(n)$	$O(kn)$	$O(k+n^2)$	

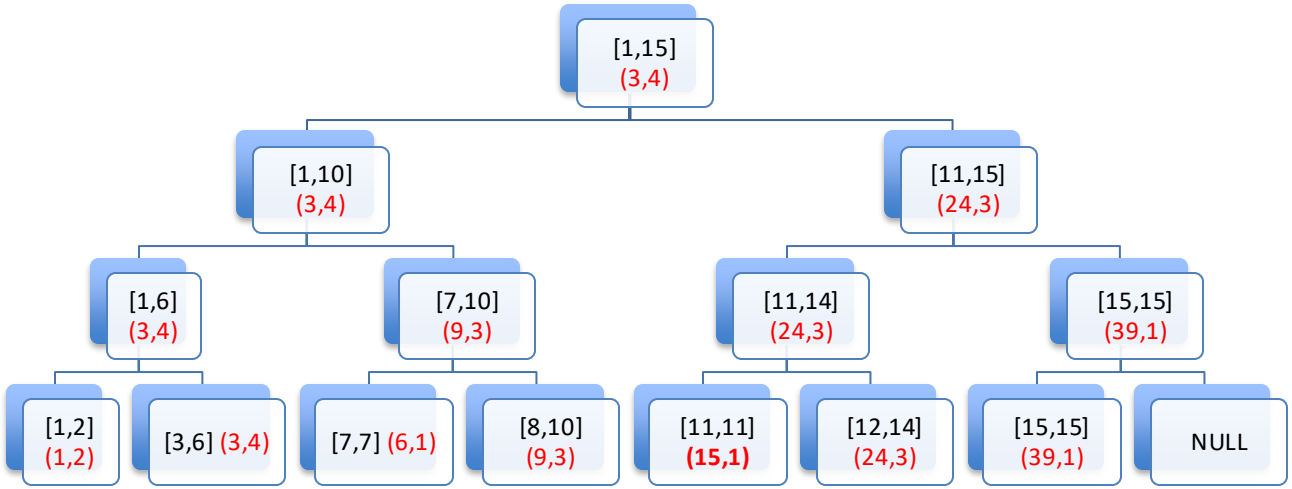
FindMode in range [i,j]

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	**1**	**3**	**3**	**3**	**3**	**6**	**9**	**9**	**9**	**15**	**24**	**24**	**24**	**39**

FindMode: Find most frequent item (mode)

	Basic	Query-v1	Query-v2	Query-v3 (Tree)
Preprocessing Time	-	$O(1)$	$O(n^2)$	$O(n)$
Preprocessing Space	-	$O(1)$	$O(n^2)$	$O(n)$
Query Time (per query)	-	$O(n)$	$O(1)$	$O(\log n)$
Total Time	$O(n)$	$O(kn)$	$O(k+n^2)$	$O(k \log n + n)$

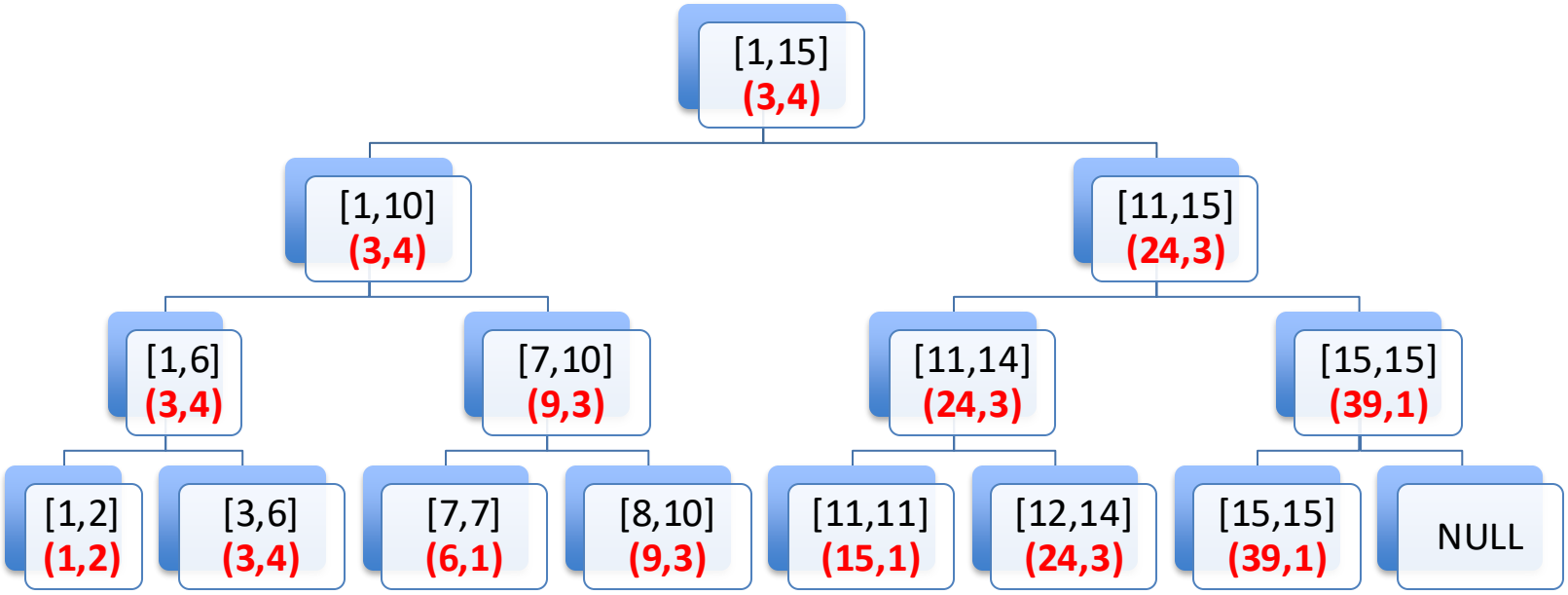
FindMode in range [i,j]



[1,2]	[3,6]	[7,7]	[8,10]	[11,11]	[12,14]	[15,15]
(1,2)	**(3,4)**	**(6,1)**	**(9,3)**	**(15,1)**	**(24,3)**	**(39,1)**

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	**1**	**3**	**3**	**3**	**3**	**6**	**9**	**9**	**9**	**15**	**24**	**24**	**24**	**39**

FindMode in range [i,j]



[1,2]	[3,6]	[7,7]	[8,10]	[11,11]	[12,14]	[15,15]
(1,2)	(3,4)	(6,1)	(9,3)	(15,1)	(24,3)	(39,1)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	3	3	3	3	6	9	9	9	15	24	24	24	39

FindMaxSum

17	4	-13	-3	9	7	6	-9	9	-19	15	24	2	-20	39

FindMaxSum: Find index range with highest sum