FindMaxSum

Giri Narasimhan

Programming Team

Fall 2024

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)

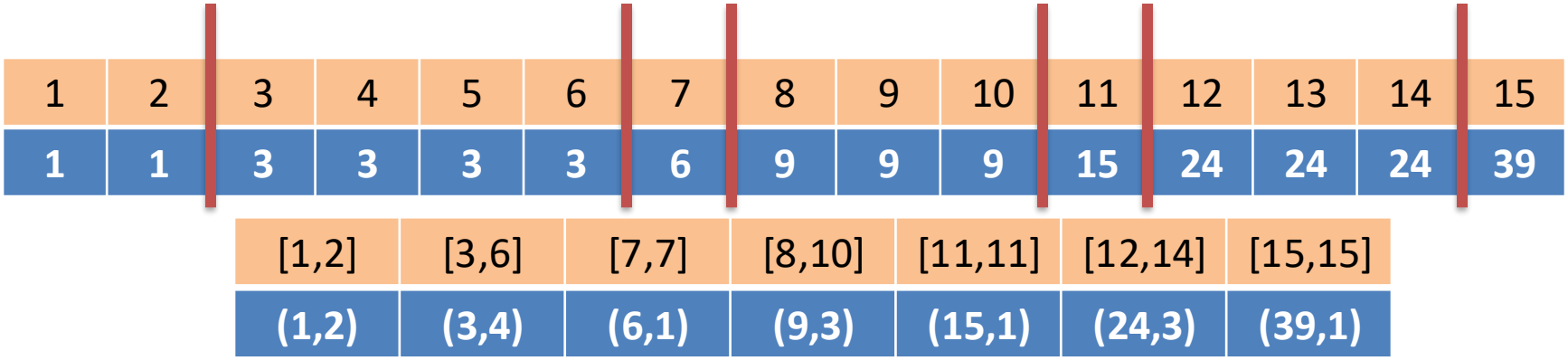
FindMode in range [i,j]

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 mode in range [i,j]

	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	$O(n)$	$O(kn)$	$O(k+n^2)$	

Idea 1: Store frequencies



FindMode: Find mode in range [i,j]

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

n: Size of input array; k: # of queries
m: # of unique items in input array

Idea 2: Use a tree

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**

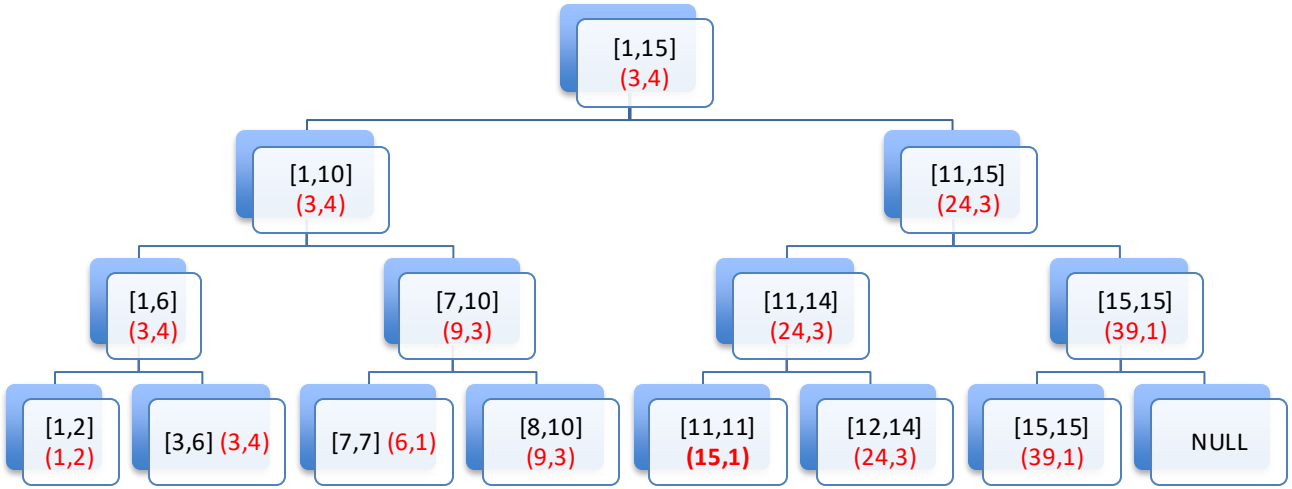
[1,2]	[3,6]	[7,7]	[8,10]	[11,11]	[12,14]	[15,15]
(1,2)	**(3,4)**	**(6,1)**	**(9,3)**	**(15,1)**	**(24,3)**	**(39,1)**

FindMode: Find mode in range [i,j]

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

n: Size of input array; k: # of queries
m: # of unique items in input array

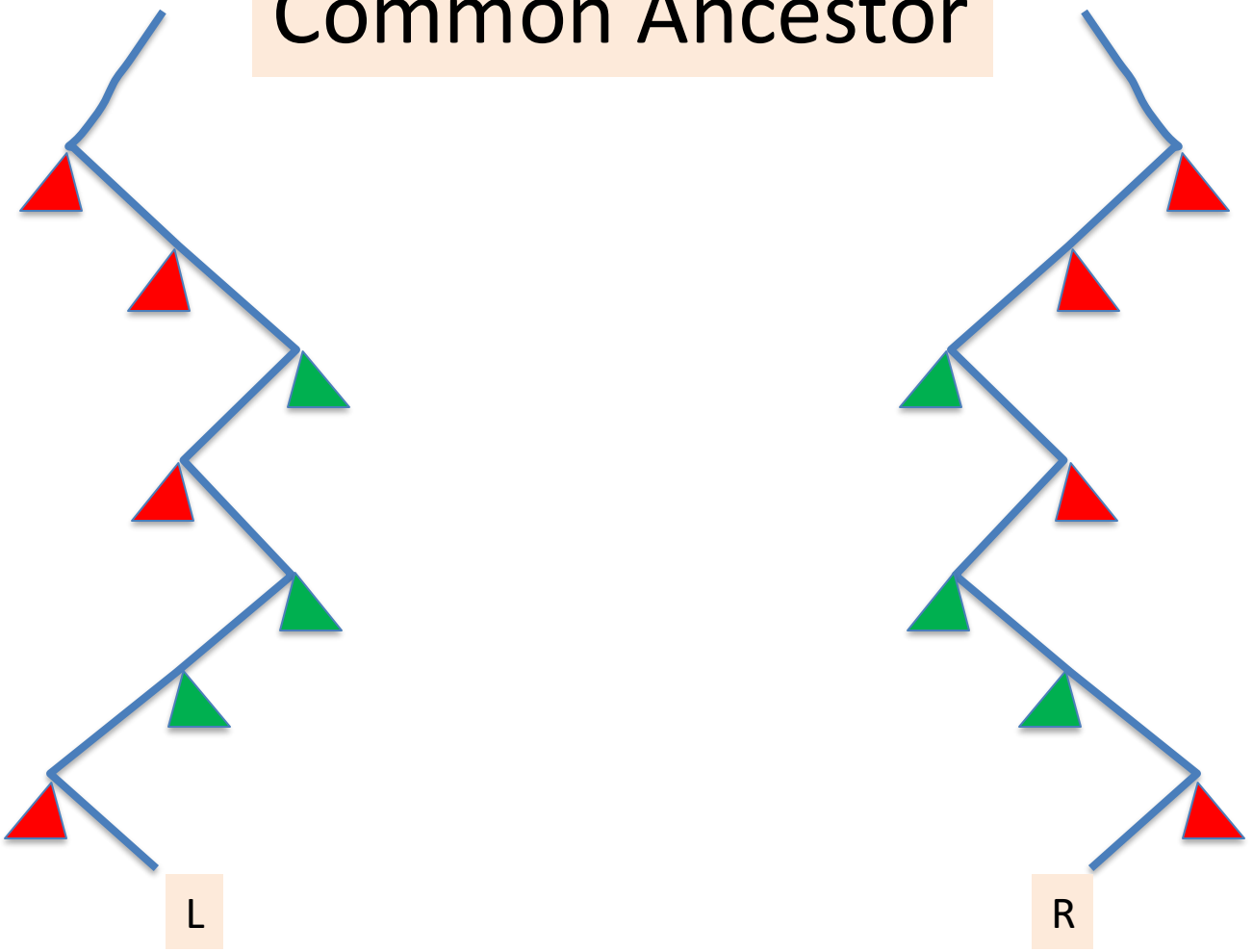
FindMode in range [i,j]



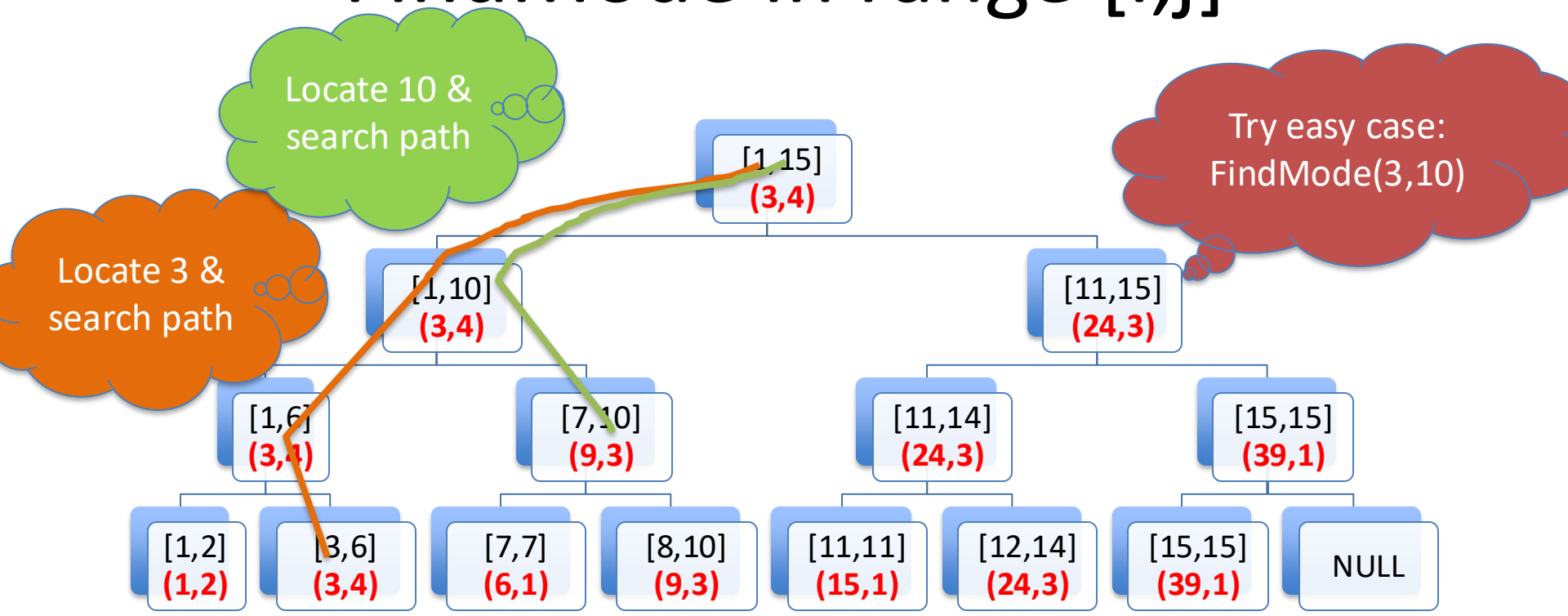
[1,2]	[3,6]	[7,7]	[8,10]	[11,11]	[12,14]	[15,15]
(1,2)	**(3,4)**	**(6,1)**	**(9,3)**	**(15,1)**	**(24,3)**	**(39,1)**

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**

Tree Search for Range [L,R]



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

FindModeQuery: Tree-based Solution

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

n: Size of input array; k: # of queries
m: # of unique items in input array

FindMaxSum

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

FindMaxSum: Find index range with highest sum.

- **Naïve Solution**

For $i = 1$ to n do

    For $j = 1$ to n do

        ComputeSum(i, j); Update MaxSum & range

Report MaxSum & range

Observation

- Max(i) = index range of form $[i, x]$ with highest sum
- Observations about Max(i) and Max(j)