Trees for FindMax : Query Versions

Giri Narasimhan

Programming Team

Fall 2024

Problem Solving

- **Correct** Solutions
- **Efficient** Solutions

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

General Approach for

- Preprocess input A
- Create data structure B
- For each input query:
  - Ask an appropriate query from B
  - Respond quickly with an answer to input query
- Complexity? Preprocessing & Query time/space
- Tradeoff
  - Preprocessing time and space vs Query time

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

- How many entries in B?
  - $O(n^2)$
- How to "Compute $B[p, q]$"?
  - Naïve: $O(q - p) = O(n)$
  - Naïve: $O(n^3)$
- Time for k queries:
  - $O(k + n^3)$
  - Fine if $n^3 = O(k)$
  - What if $n^2 = O(k)$, but $k < n^3$
  - Need better preprocessing

Preprocessing Algorithm

How to fill in 2D array B

For $p = 1$ to n do

  For $q = 1$ to n do

    Compute $B[p, q]$

Improved Preprocessing

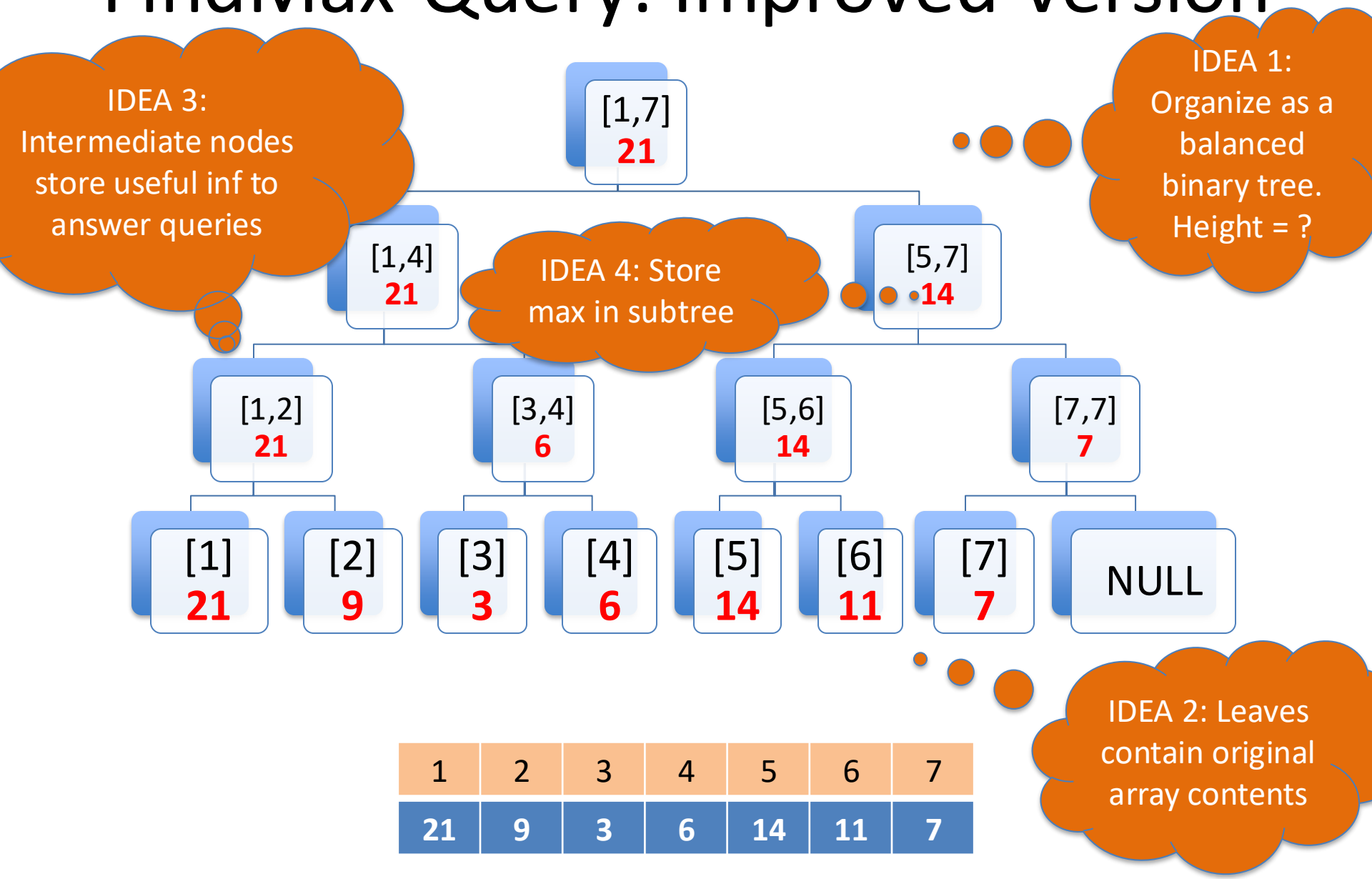
- How to "Compute $B[p, q]$"
  - Use: $B[p, q-1]$
  - $B[p, q] = \max\{B[p, q-1], A[q]\}$
- "Compute $B[p, q]$" $= O(1)$
- Time for k queries:
  - $O(k + n^2)$
  - Fine if $n^2 = O(k)$
  - What if $n = O(k)$, but $k < n^2$
  - Need better preprocessing

FindMax-Query

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

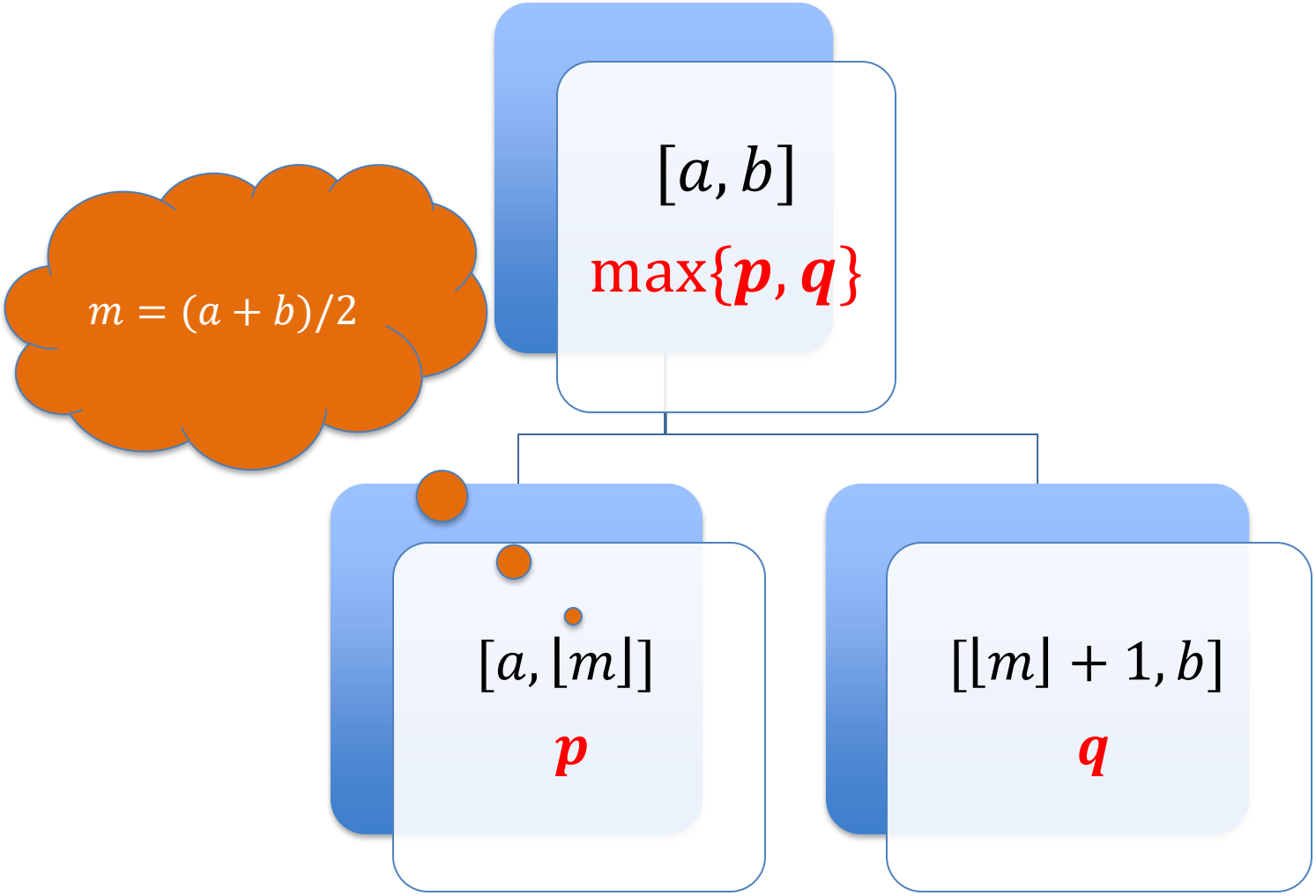
How about a better tradeoff between Preprocessing time and Query time?

FindMax-Query: Improved Version

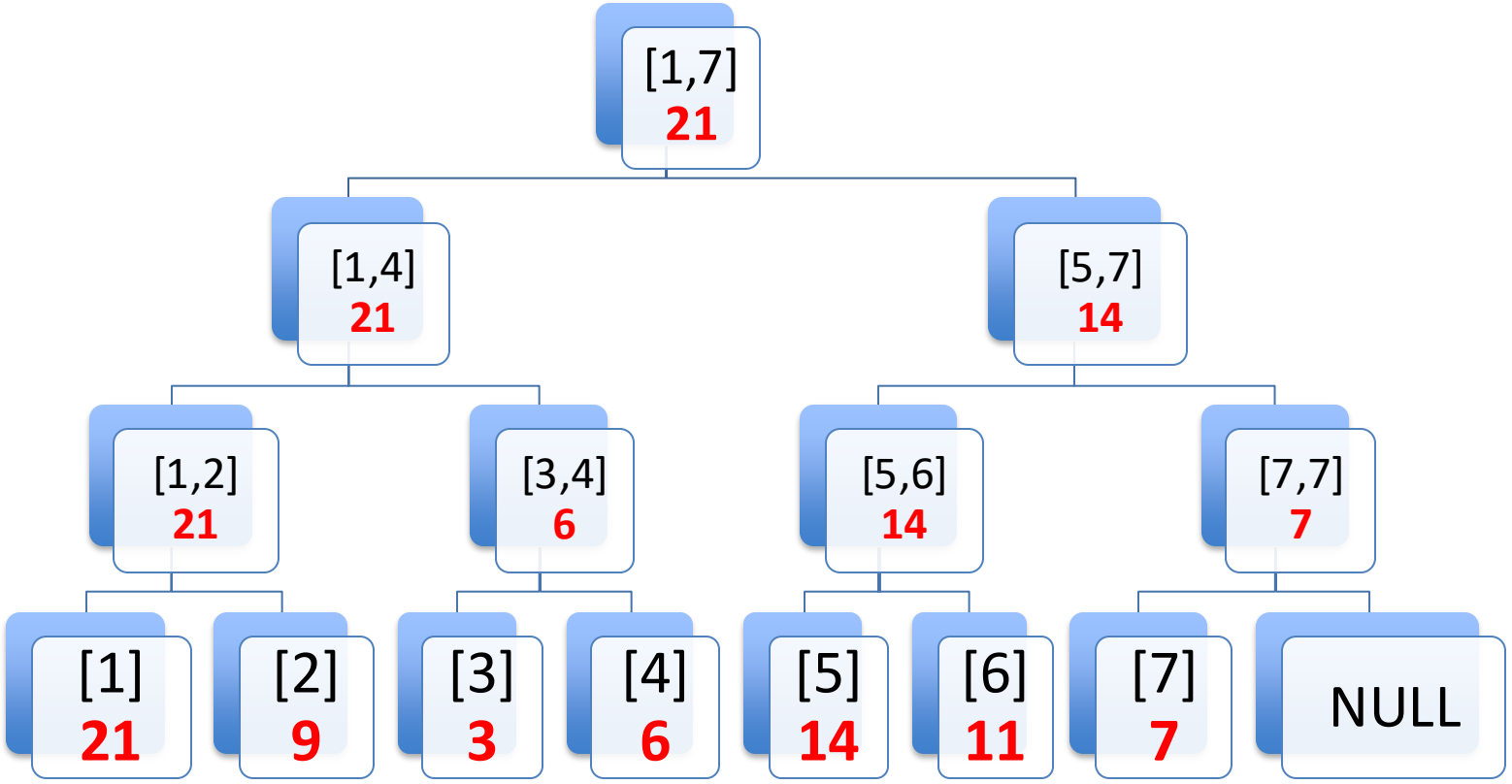


1	2	3	4	5	6	7
21	9	3	6	14	11	7

FindMax-Query: Node details



FindMax-Query: Improved Version



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

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 $u.\text{MaxValue}$

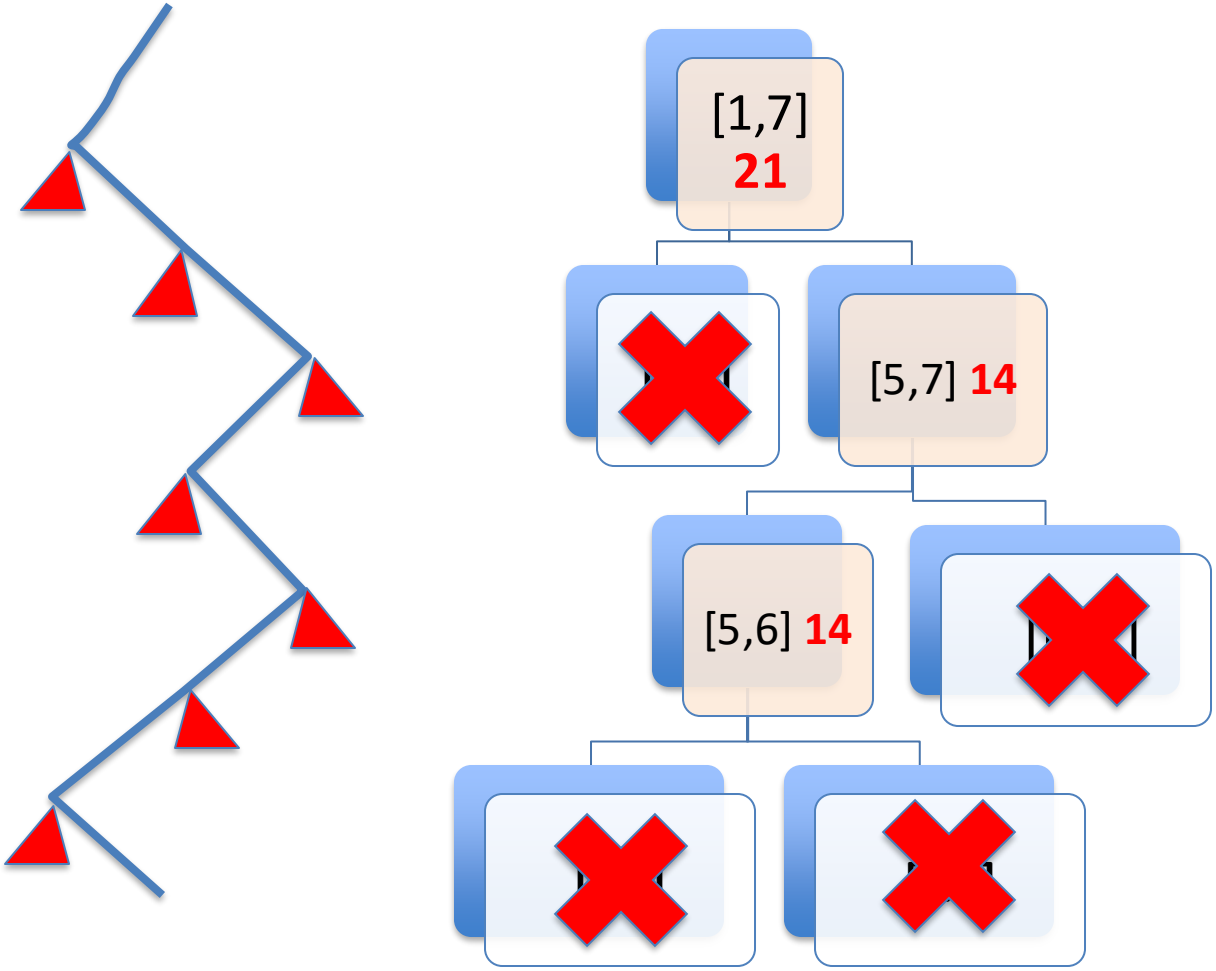
- Else if $(\text{u.Interval.Rt} == [i, k])\&(k < j)$ then

  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



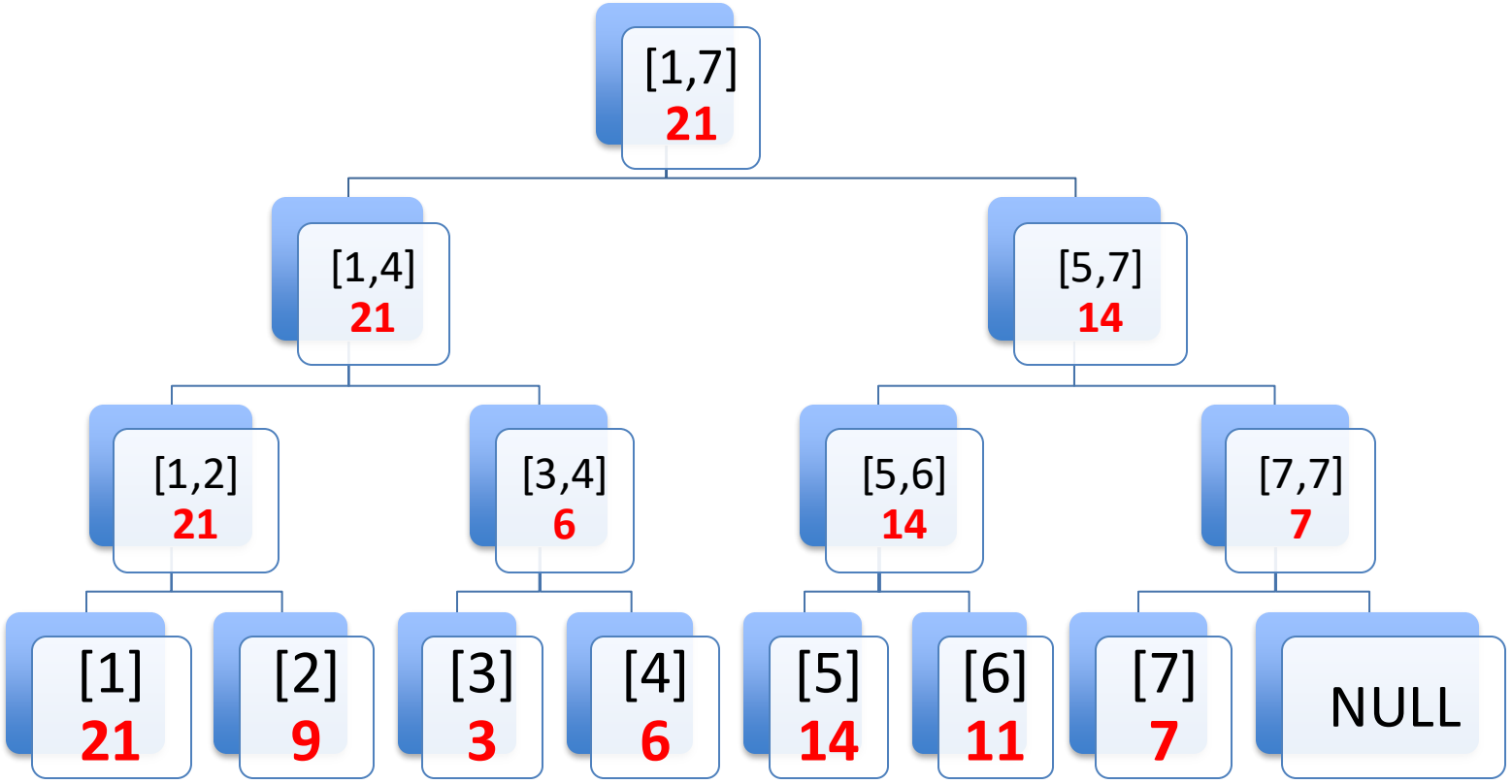
Search for range [5,6]

FindMax-Query

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

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