Max item & Frequency Counts

Giri Narasimhan
Programming Team
Fall 2019

Dynamic Queries: FindMax

- FindMax(2,5) = 14
- FindMax(1,3) = 21
- FindMax(3,4) = 6
- FindMax(5,3) = undefined

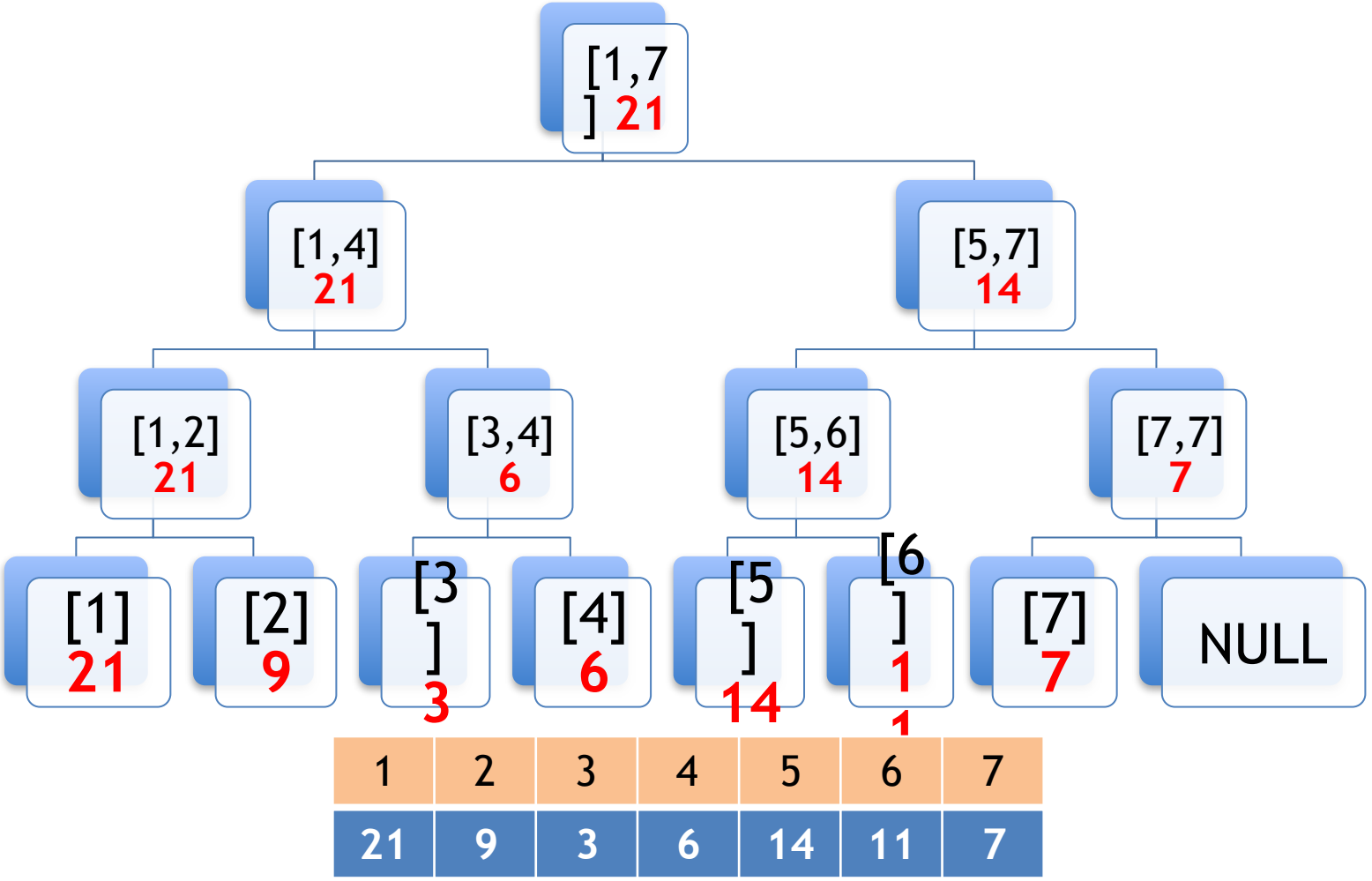
1	2	3	4	5	6	7
21	9	3	6	14	11	7

New Queries: FindMax

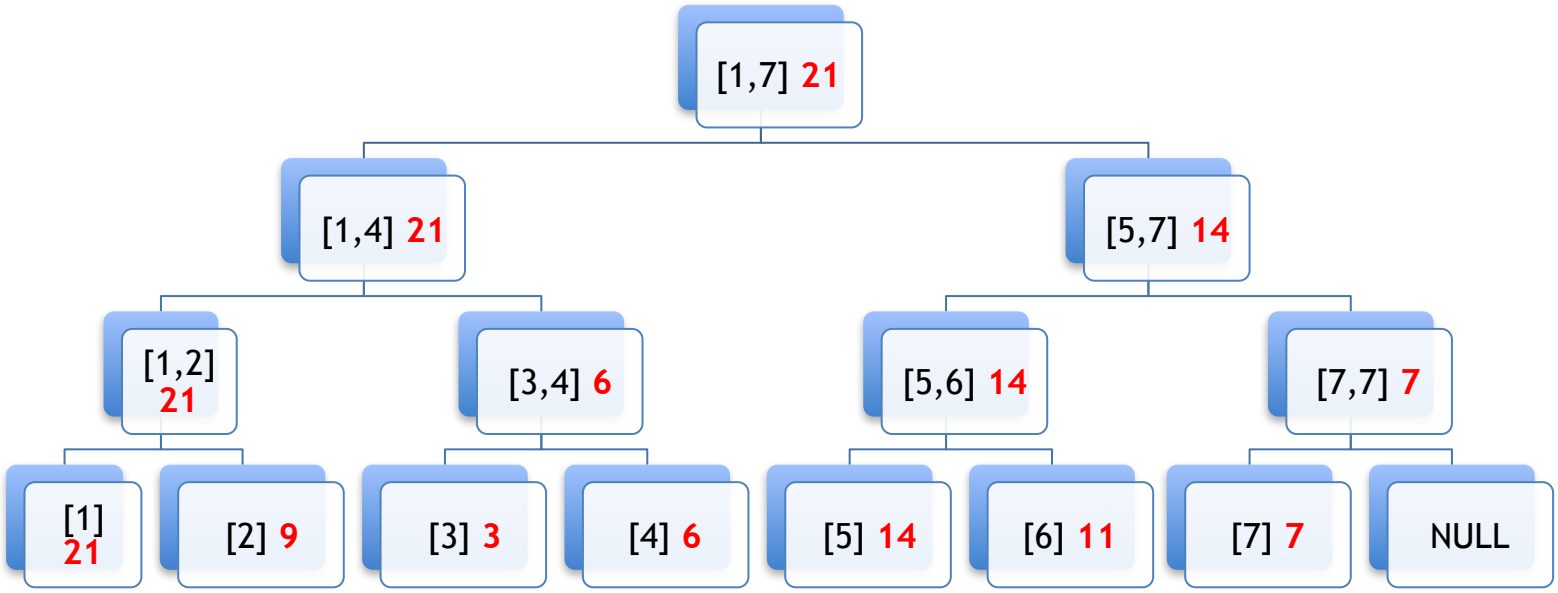
- Given dynamic list with standard operations:
  - Search, insert, delete
- Efficiently answer queries such as:
  - FindMax (StartIndex, EndIndex)
- For e.g.: FindMax(2,5) = 14; FindMax(1,3) = 21

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

Find Max in given range



Find Max in given range [i,j]



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

Harder Problem: Find Most Frequent item 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)

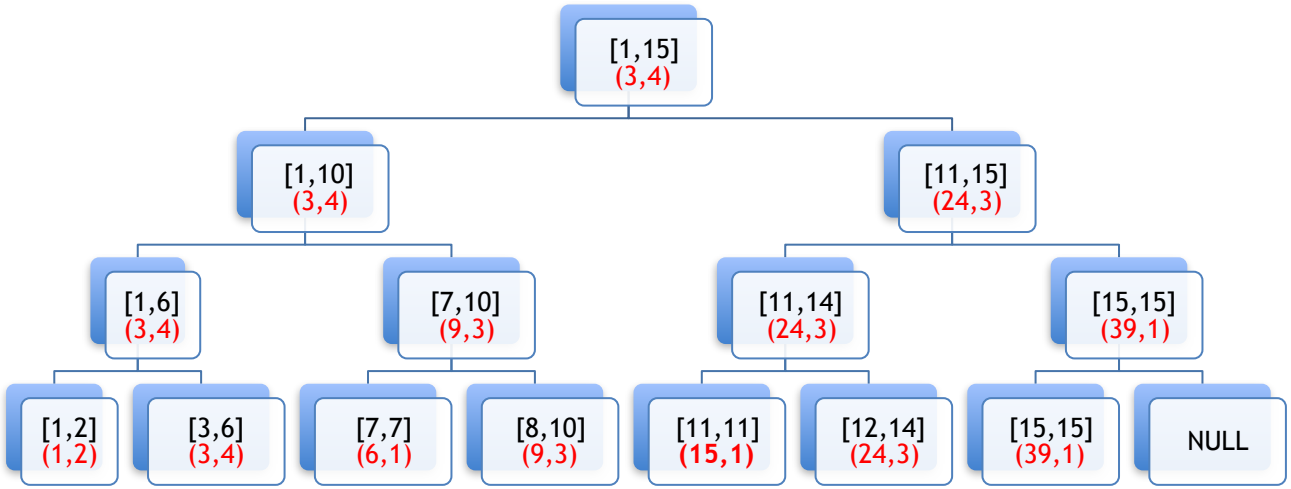
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	3	3	3	3	6	9	9	9	15	24	24	24	39

Find Most Frequent item in range [i,j]

- Given a sorted array with repeats, answer FindMF queries, which reports the most frequent item in a given range of items
  - FindMF(1,5) = 3
  - FindMF(5,10) = 9

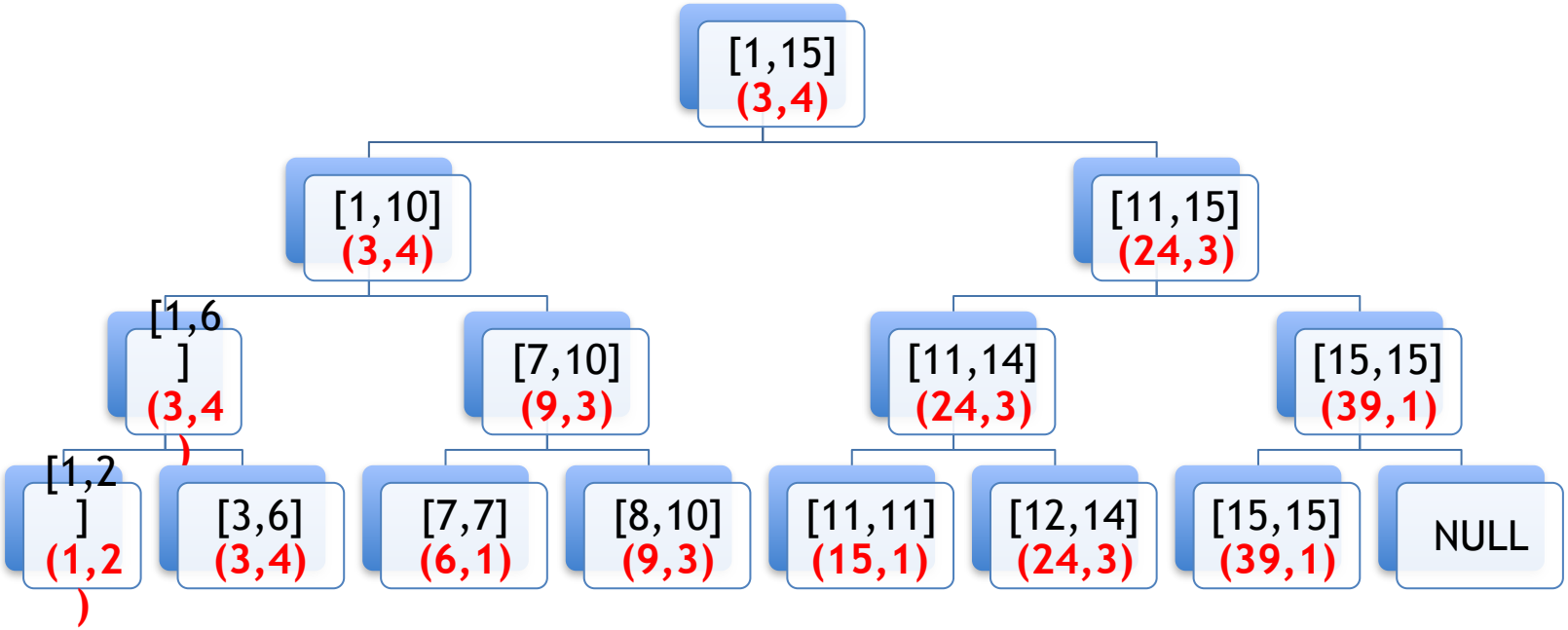
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	3	3	3	3	6	9	9	9	15	24	24	24	39

Harder Problem: Find Most Frequent item 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



[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

Organizing Data into Structures

- Data items stored and organized into Data Structures for efficient querying
- Data item
  - Primary Key
  - Secondary Key & Additional information

Basic Data Structure Operations

- Search
- Insert
- Delete
- …

Unsorted Arrays vs Sorted Arrays

- Unsorted Arrays
  - Easier to insert
  - Harder to search and delete
- Sorted Arrays
  - Easier to search
  - Harder to insert and delete

Sorted Arrays vs BSTs

- Sorted Arrays
  - Easier to search
  - Harder to insert and delete
- BSTs
  - Easier (average) to search, insert and delete
- Balanced BSTs
  - Easier (worst-case) to search, insert and delete

Advanced Queries

- Range Queries
  - How many students between 19 and 21 years old
- Queries on secondary keys
  - Highest GPA of student between 19 and 21 yrs
- Complex Range Queries
  - How many students between 19 and 21 yrs with GPA between 3.25 and 3.75

Need **Augmented Data Structures**

Operations on Dynamic RB Trees

- **K-Selection**
  - **Select** an item with a specified rank
  - **"Efficient" solution not possible without preprocessing**
  - **Preprocessing - store additional information at nodes**
- **Inverse of K-Selection**
  - Find **rank** of an item in the tree
- **What information should be stored?**
  - **Rank**
  - **??**

OS-Rank

```
OS-RANK(x,y)
// Returns rank of x in subtree rooted at y
1. r = size[left[y]] + 1
2. if x = y then return r
3. else if ( key[x] < key[y] ) then
4.     return OS-RANK(x,left[y])
5. else return r + OS-RANK(x,right[y] )
```

Time Complexity O(log n)

OS-Select

OS-SELECT(x,i) //page 304

// Select the node with rank i in subtree rooted at x

```
1. r = size[left[x]]+1
2. if i = r then
3.             return x
4. elseif  i < r then
5.             return OS-SELECT (left[x], i)
6. else      return OS-SELECT (right[x], i-r)
```

Time Complexity O(log n)

RB-Tree Augmentation

- Augment x with **Size(x)**, where
  - Size(x) = size of subtree rooted at x
  - Size(NIL) = 0

How to augment data structures

1. choose an underlying data structure
2. determine additional information to be maintained in the underlying data structure,
3. develop new operations,
4. verify that the additional information can be maintained for the modifying operations on the underlying data structure.

Augmentations for RB-Trees

- Parent
- Height
- Any associative function on all previous values or all succeeding values.
- Next
- Previous