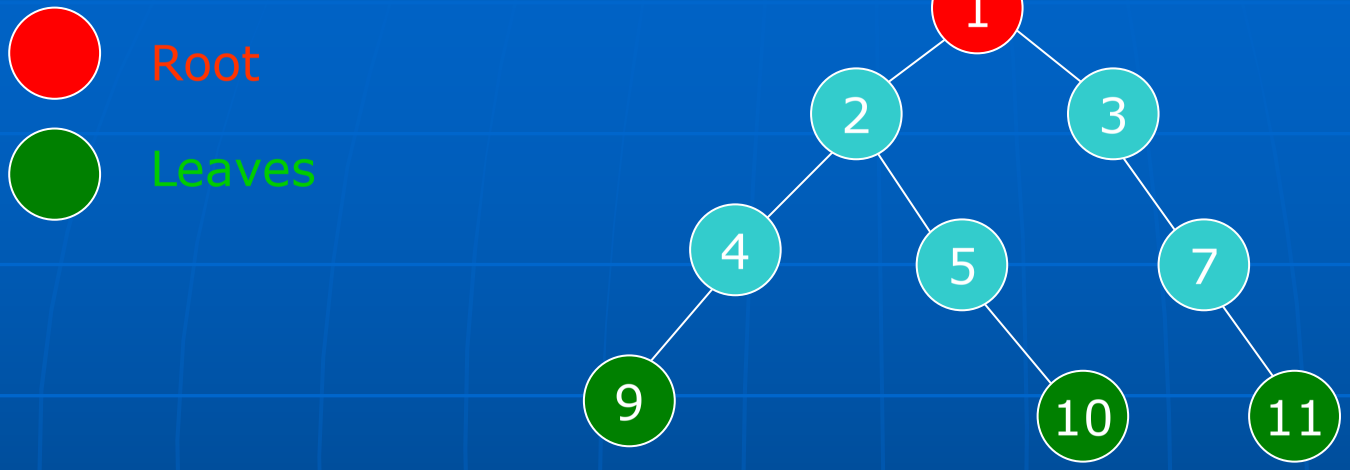
Trees

By Charl du Plessis

Contents

- Basic Terminology
- Binary Search Trees
- Interval Trees
- Binary Indexed Trees

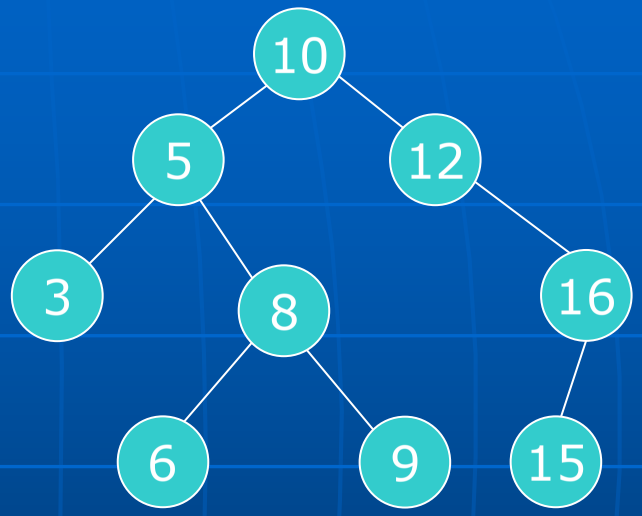
Basic Terminology



- Connected graph with no cycles
- Unique path from every vertex to every other vertex
- E = V-1 in any tree, where E and V are the number of edges and vertices respectively.

Binary Search Trees

- Structure?
- Operations: O(h)
  - query
  - delete
  - insert



Cons:

Degenerate trees make all operations Worst case O(n)

But can be “self-balanced” such as in

Red-Black Trees.

C++ stl <set> uses self-balancing binary trees

Interval Trees: Problem

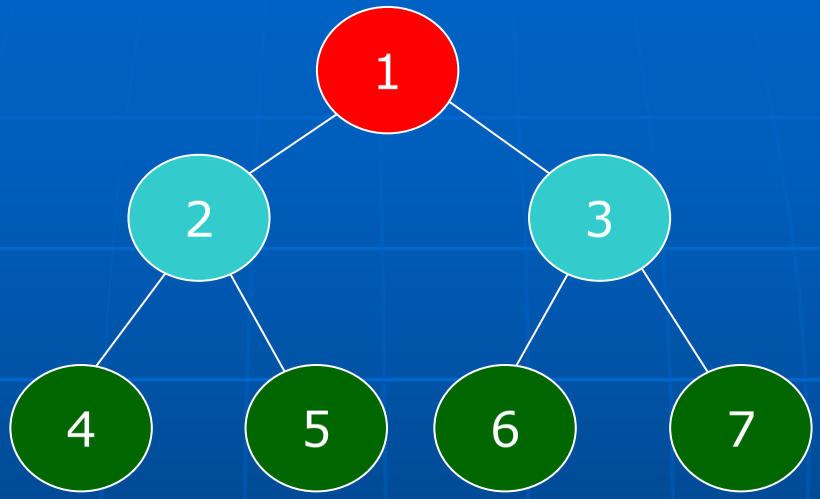
Consider the problem:

- You have a set of n items; 1,…, n
- There is some number of each item
- You want to increment the number of each item in range [x,y] by one. (for each i, x <= i <= y)
- And query how many of item i there are

Interval Trees: Structure

- Use a binary tree, where each node holds a count over range [A,B]
- Root holds count over [1,n]
- Node ranging over [A,B] will have children [A, (A+B)/2] and [(A+B)/2+1, B]
- The leaves of the tree will store count over a single item range

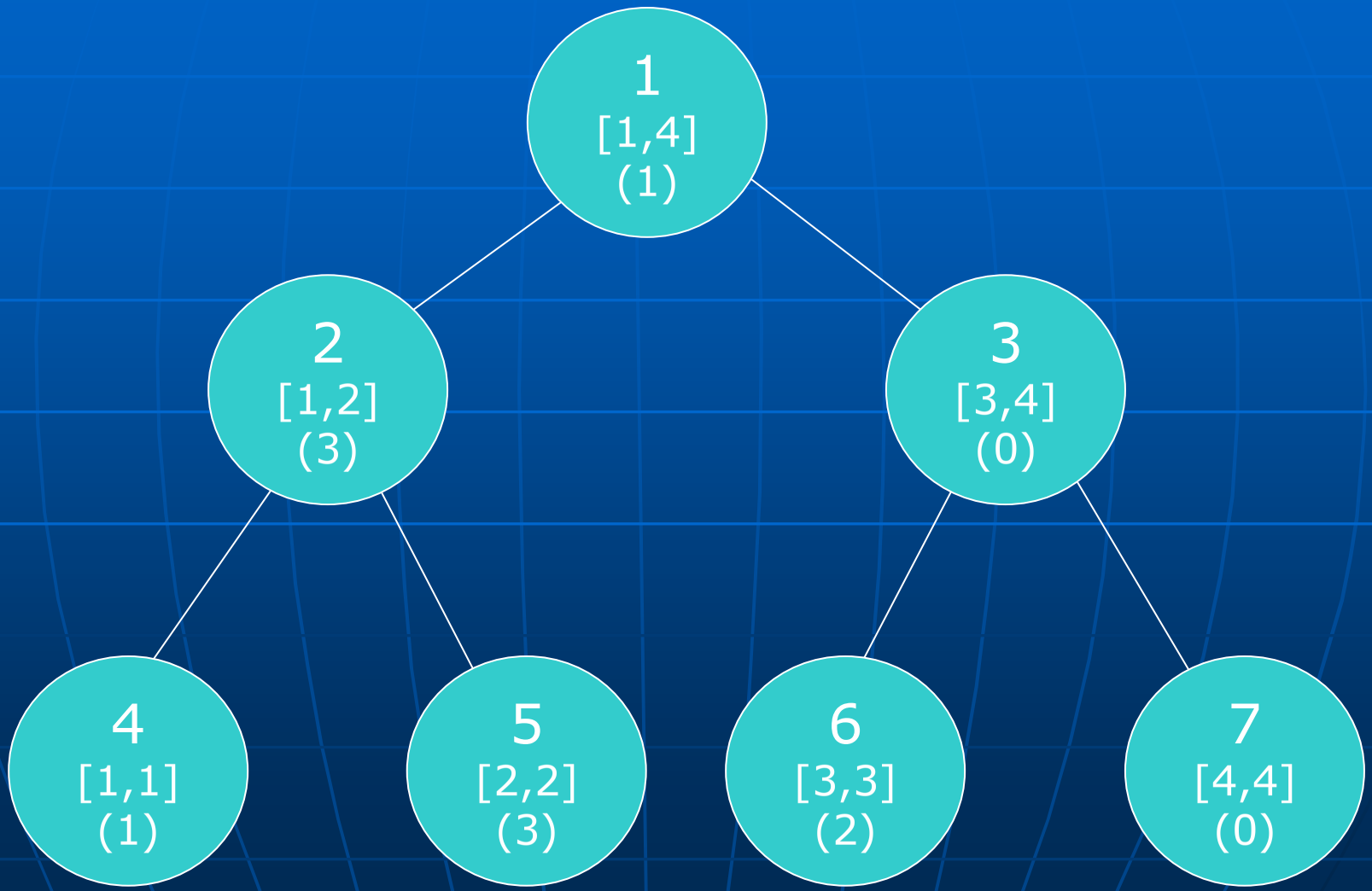
Interval Trees: Representation



Representation of tree:

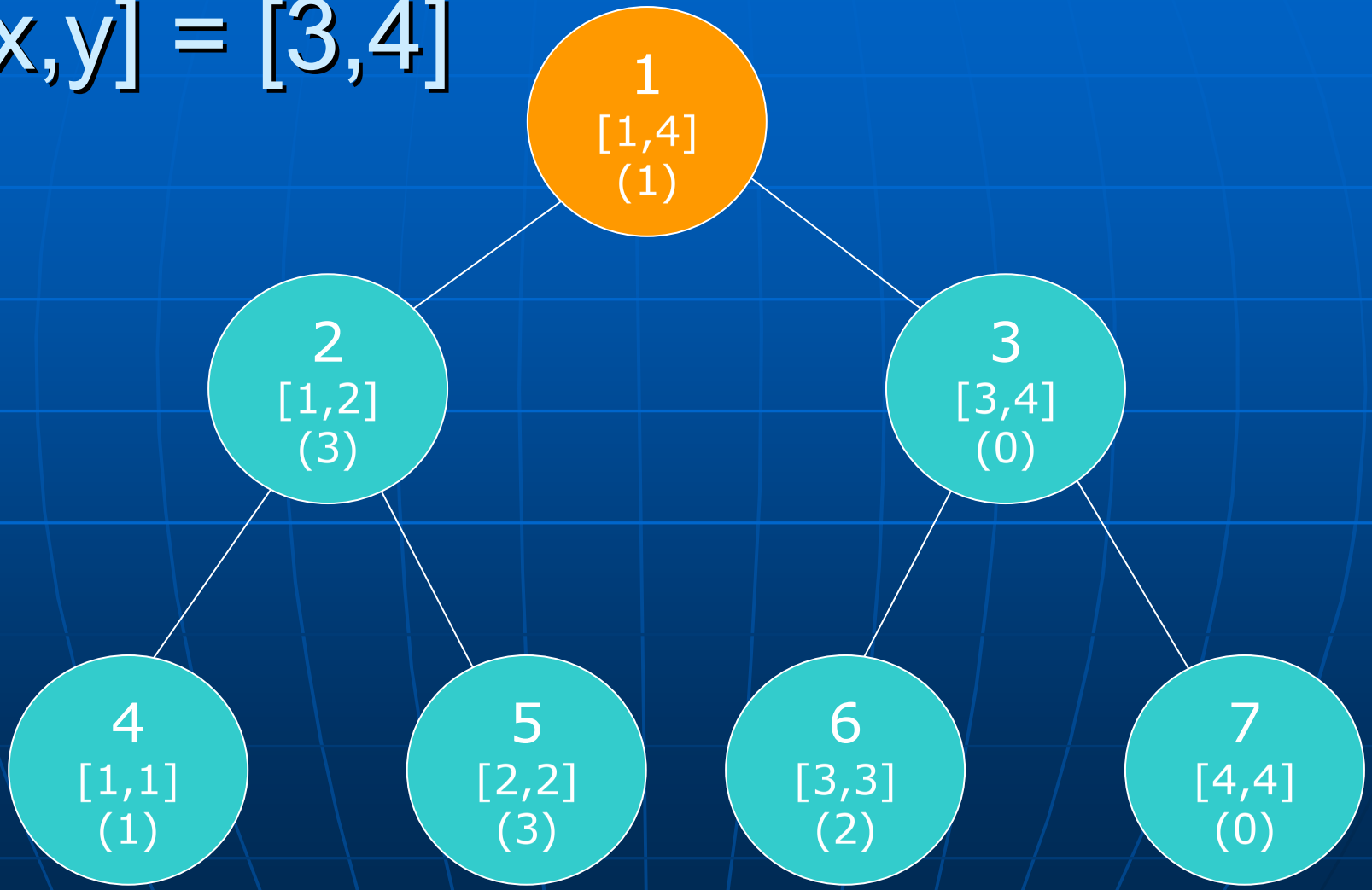
- Use an array
- Index nodes from 1
- Children of node i will be 2*i and 2*i +1

Interval Trees: Updating a range



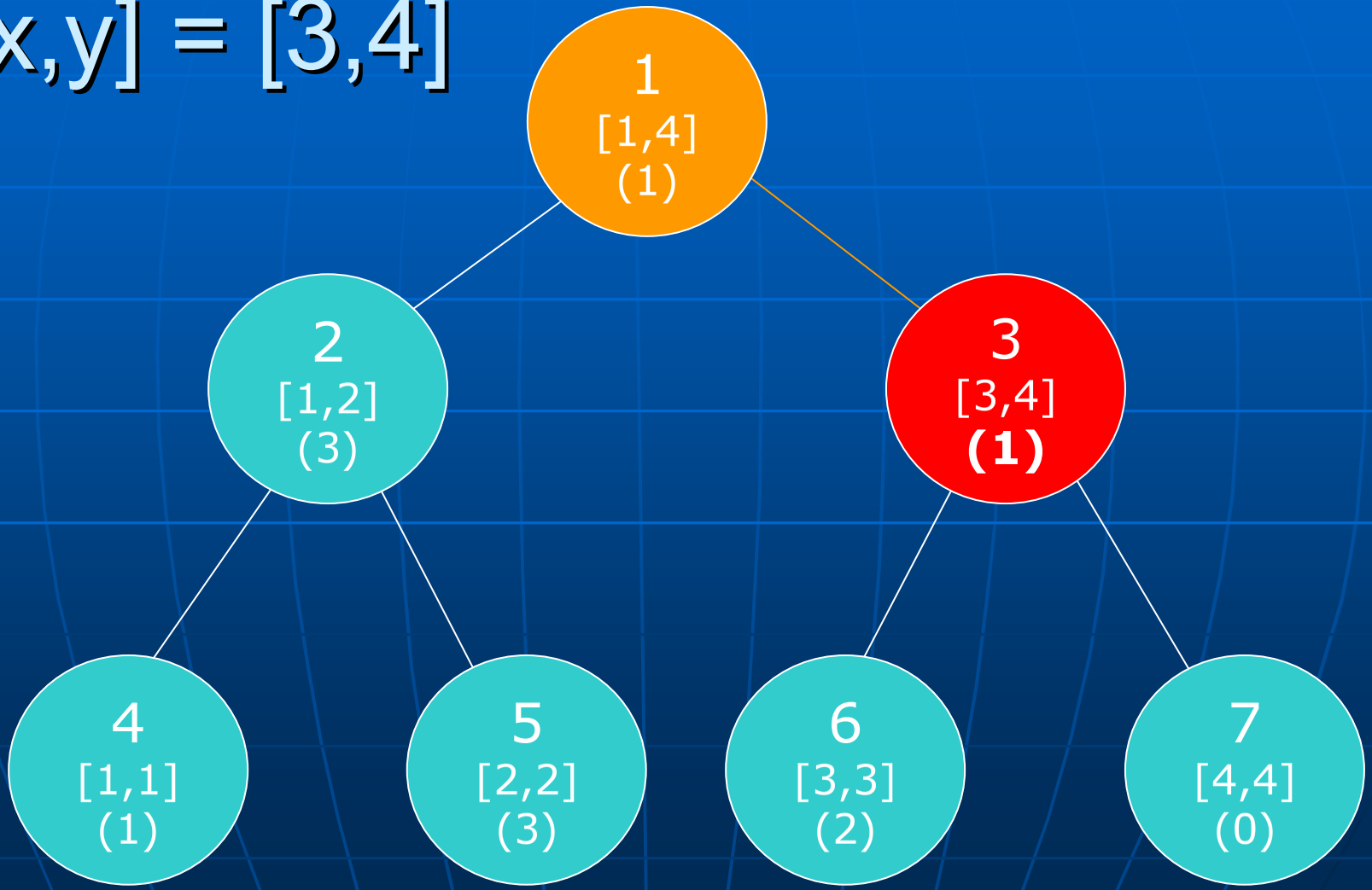
Interval Trees: Updating a range

[x,y] = [3,4]



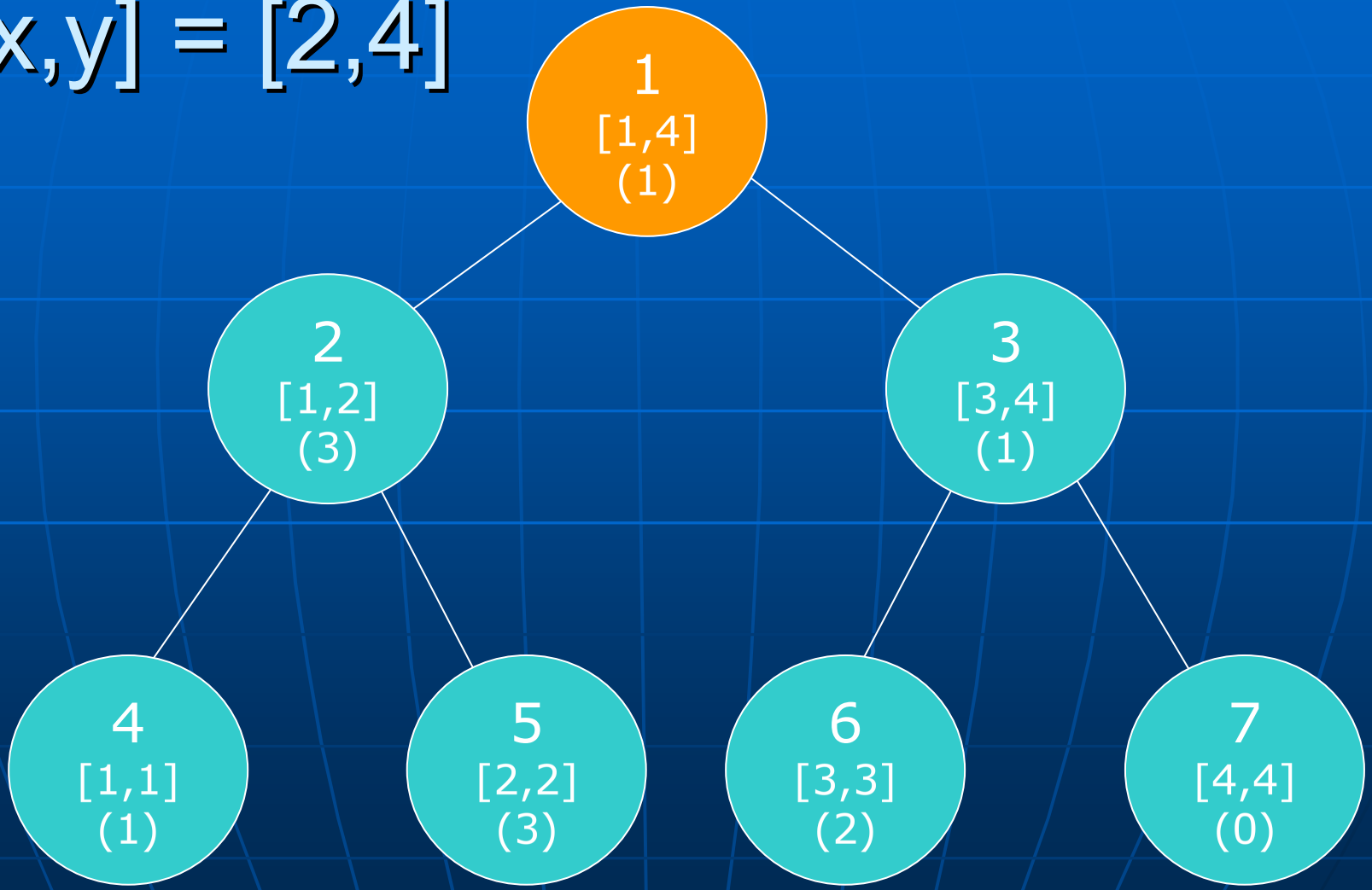
Interval Trees: Updating a range

[x,y] = [3,4]



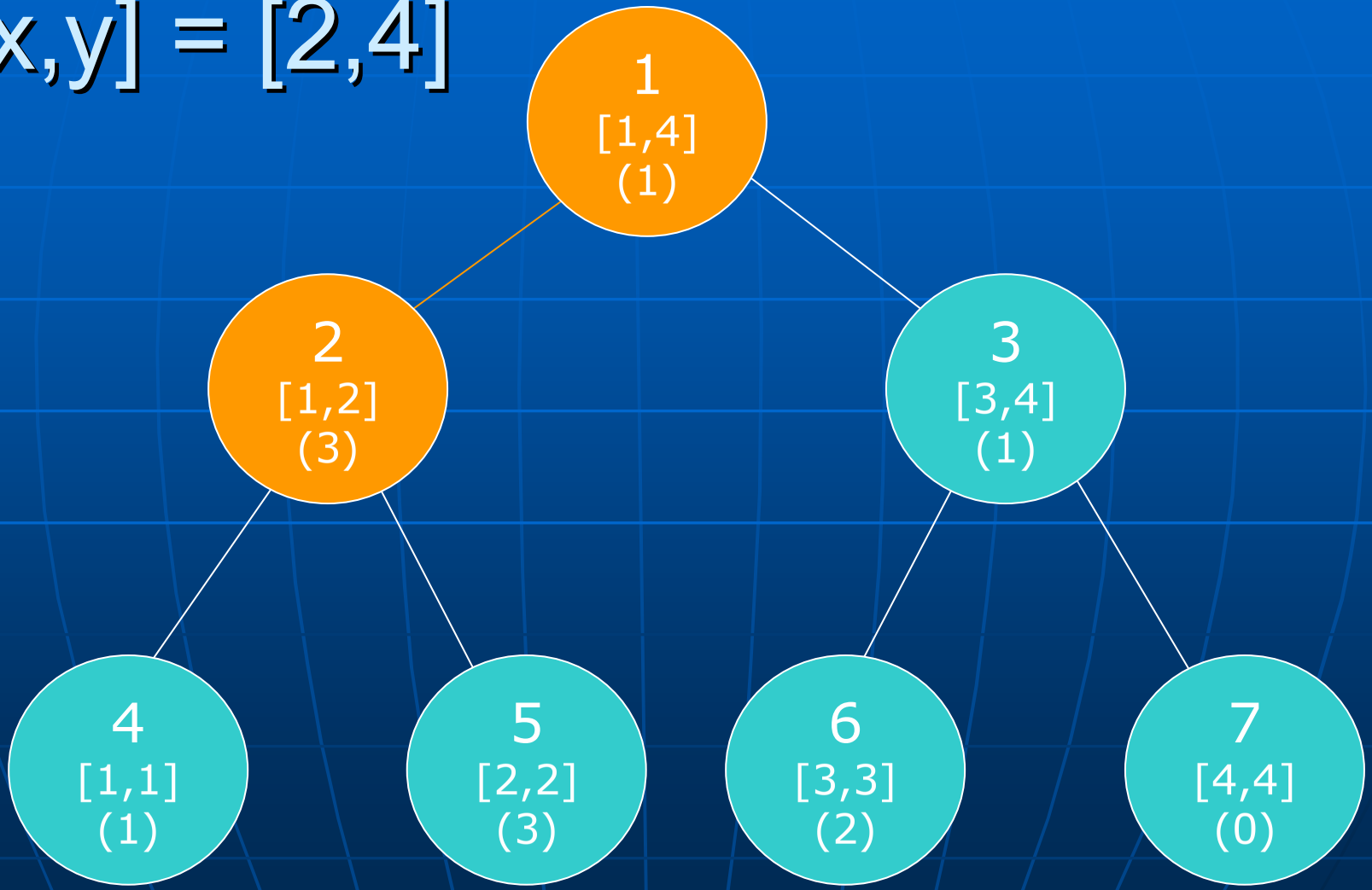
Interval Trees: Updating a range

[x,y] = [2,4]



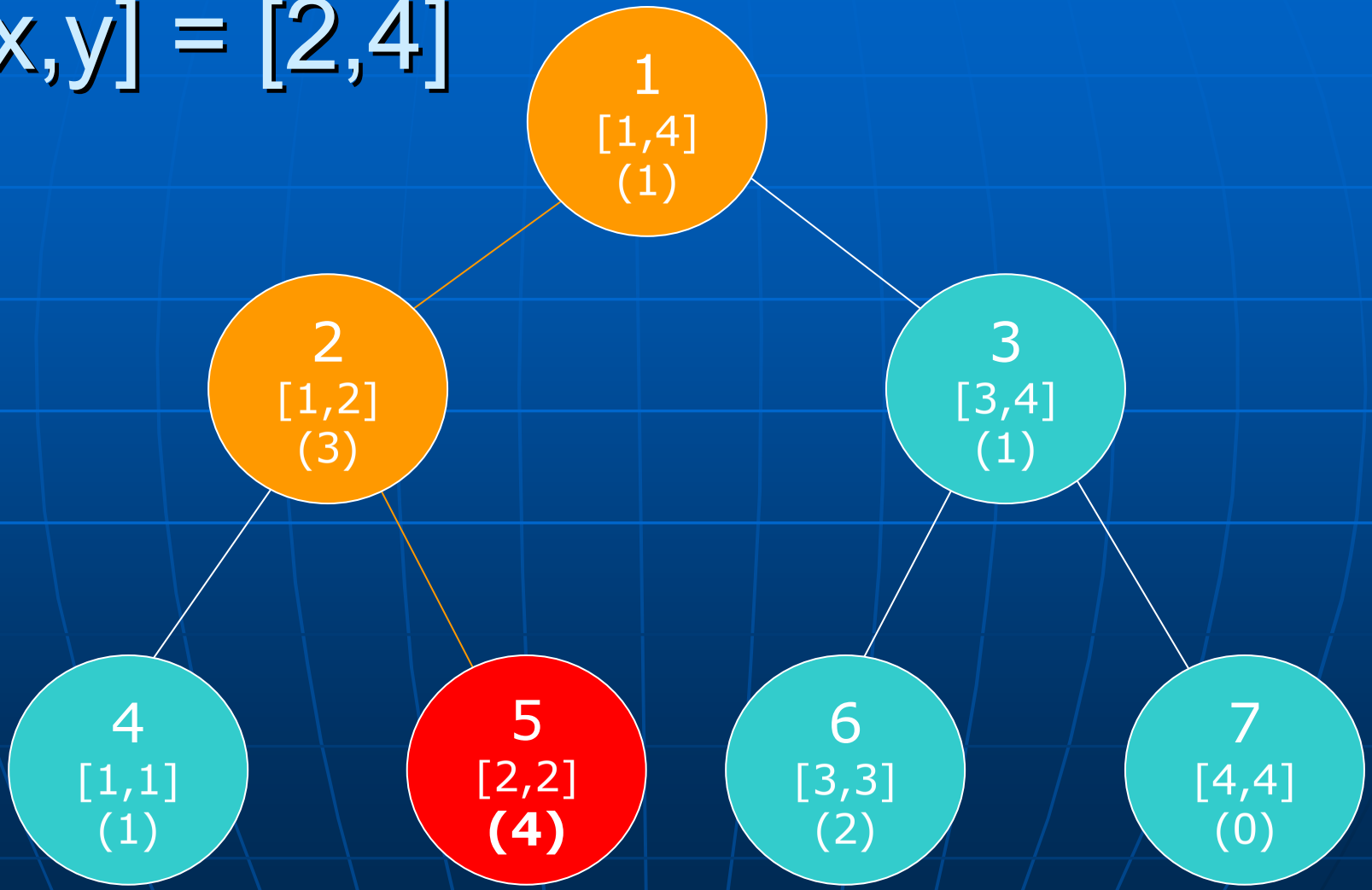
Interval Trees: Updating a range

[x,y] = [2,4]



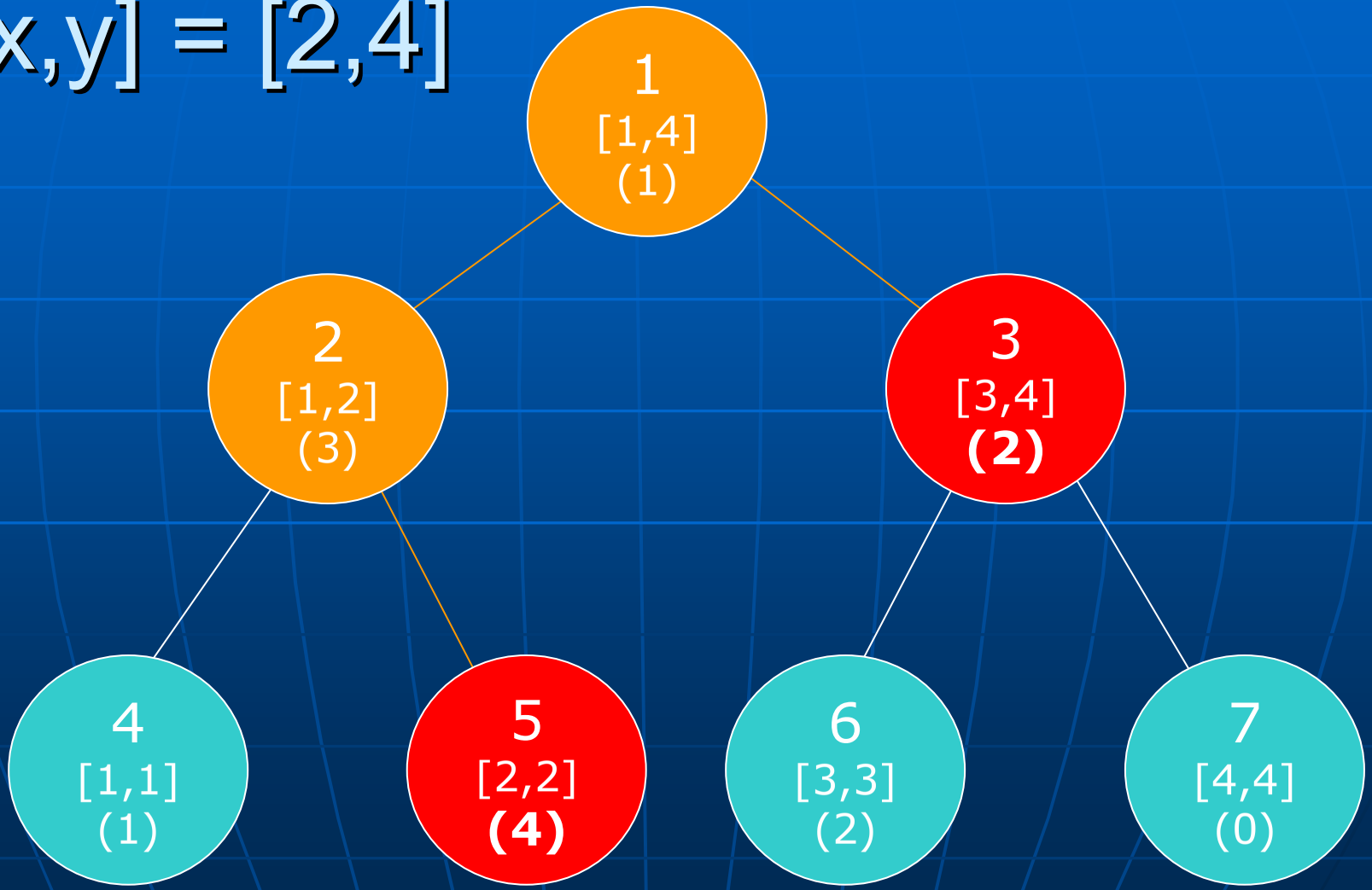
Interval Trees: Updating a range

[x,y] = [2,4]



Interval Trees: Updating a range

[x,y] = [2,4]



Interval Trees: Updating Code

Updating a range [x,y]:

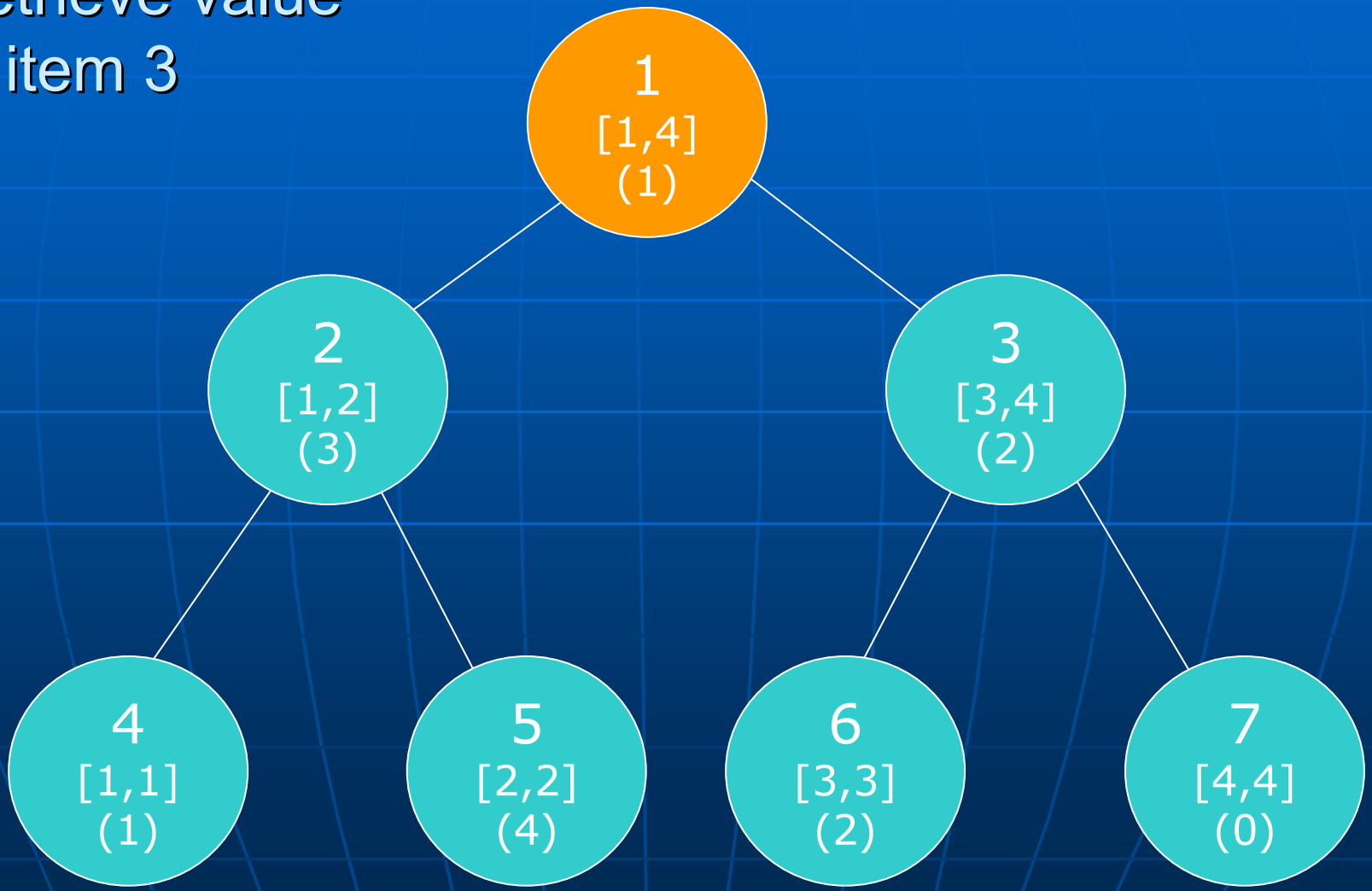
- Recurse into the tree, starting at root
- Suppose [A,B] is the current interval being considered:

```
If [x,y] overlaps [A,B]:
        Increment count of node
Else if [A,B] contains none of [x,y]:
        Stop
Else:
        Recurse into Left Child.
        Recurse into Right Child.
```

Interval Trees: Query

Retrieve value
of item 3

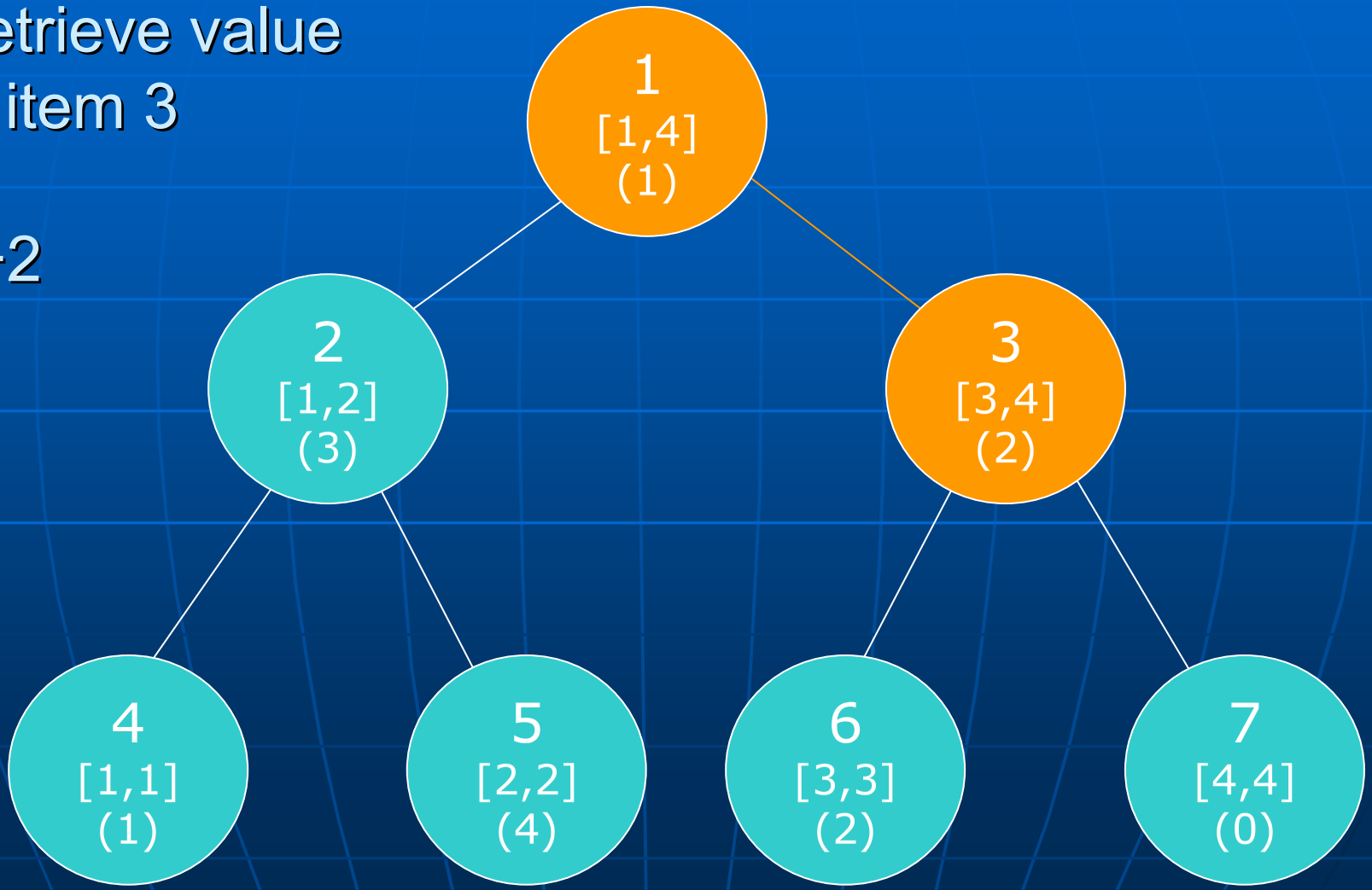
1



Interval Trees: Query

Retrieve value of item 3

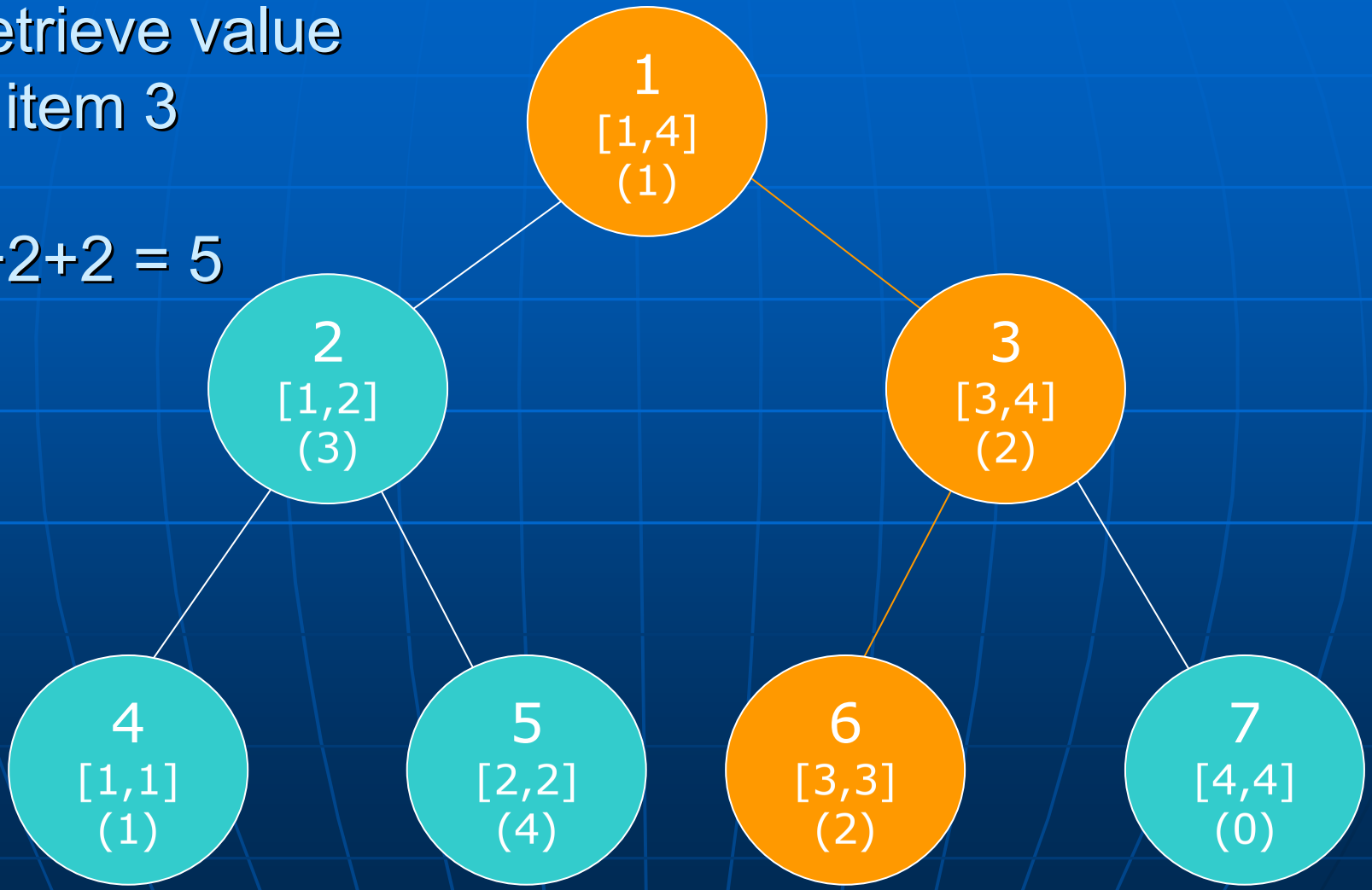
1+2



Interval Trees: Query

Retrieve value
of item 3

1+2+2 = 5



Interval Trees: Query Code

Querying the number of item i:

- Let sum = 0
- Recurse into tree.
- Considering range [A,B]:

```
sum += count of [A,B]
If (A != B):
        Recurse into child containing i
        in its range.
```

Binary Indexed Trees: Problem

Problem: Suppose we have a row of n numbers.

We want to:

1.) Increment value of the ith number

2.) Query the sum of the values in range [k,j]

BIT implementation:

- Query can be done in O(log n) for worst case
- Short code

Radix trees are equivalent in efficiency

Binary Indexed Trees: Introduction

- Interested in counts over ranges of the form [1,i]
- Uses the fact every number can be written as the sum of powers of 2.
- We will use this to represent ranges [1,i]
- 13 = 8 + 4 + 1
- [1, 13] = [1, 8] + [9, 12] + [13, 13]

Binary Indexed Trees: Intervals

- We store interval counts of the form [i -2^r+1, i] where r is the position of the last non-zero digit of i in binary form
- Example:

  13 = 1011 in binary, position of last non-zero digit is 0.

  4 = 100 in binary, position of last non-zero digit is 2.

- How it works will become clear when we explain the structure of the tree

Binary Indexed Trees: Structure

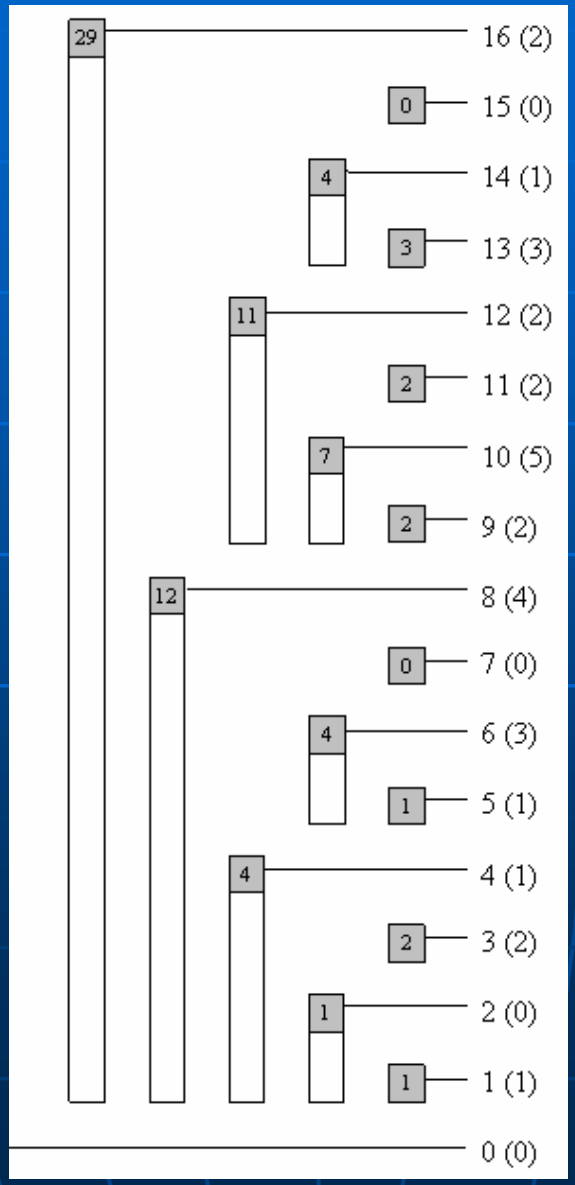
f[i] = value at index i

c[i] = f[1] + f[2] + … + f[i]

tree[i] = count of [i-2^r +1,i]

i	(i-2^r+1)…i
1	1
2	1…2
3	3
4	1…4
5	5
6	5…6
7	7
8	1…8
9	9
10	9…10
11	11
12	9…12
13	13
14	13…14
15	15
16	1…16

Binary Indexed Trees: Structure



Binary Indexed Trees: Query

```
int read(int idx)
{
    int sum = 0;
    while (idx > 0)
    {
        sum += tree[idx];
        idx -= (idx & -idx);
    }
    return sum;
}
```

Binary Indexed Trees: Update

```
void update(int idx ,int val)
{
   while (idx <= MaxVal)
   {
      tree[idx] += val;
      idx += (idx & -idx);
   }
}
```

Questions