Interval Trees

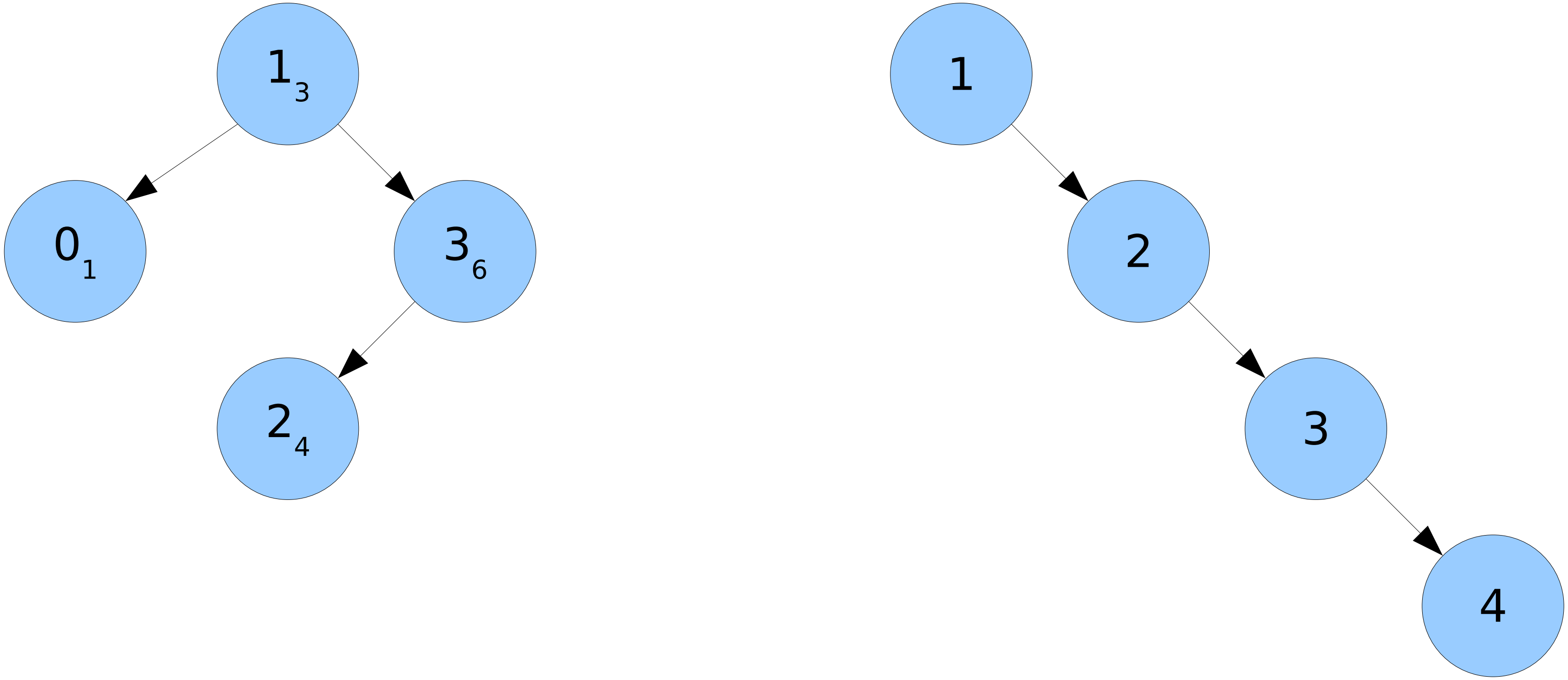
Marco Gallotta

Problem

- Given a collection of items i, each with value V_i
- Want to answer many queries of the form: How many items are in the interval [x, y]?

Binary Tree

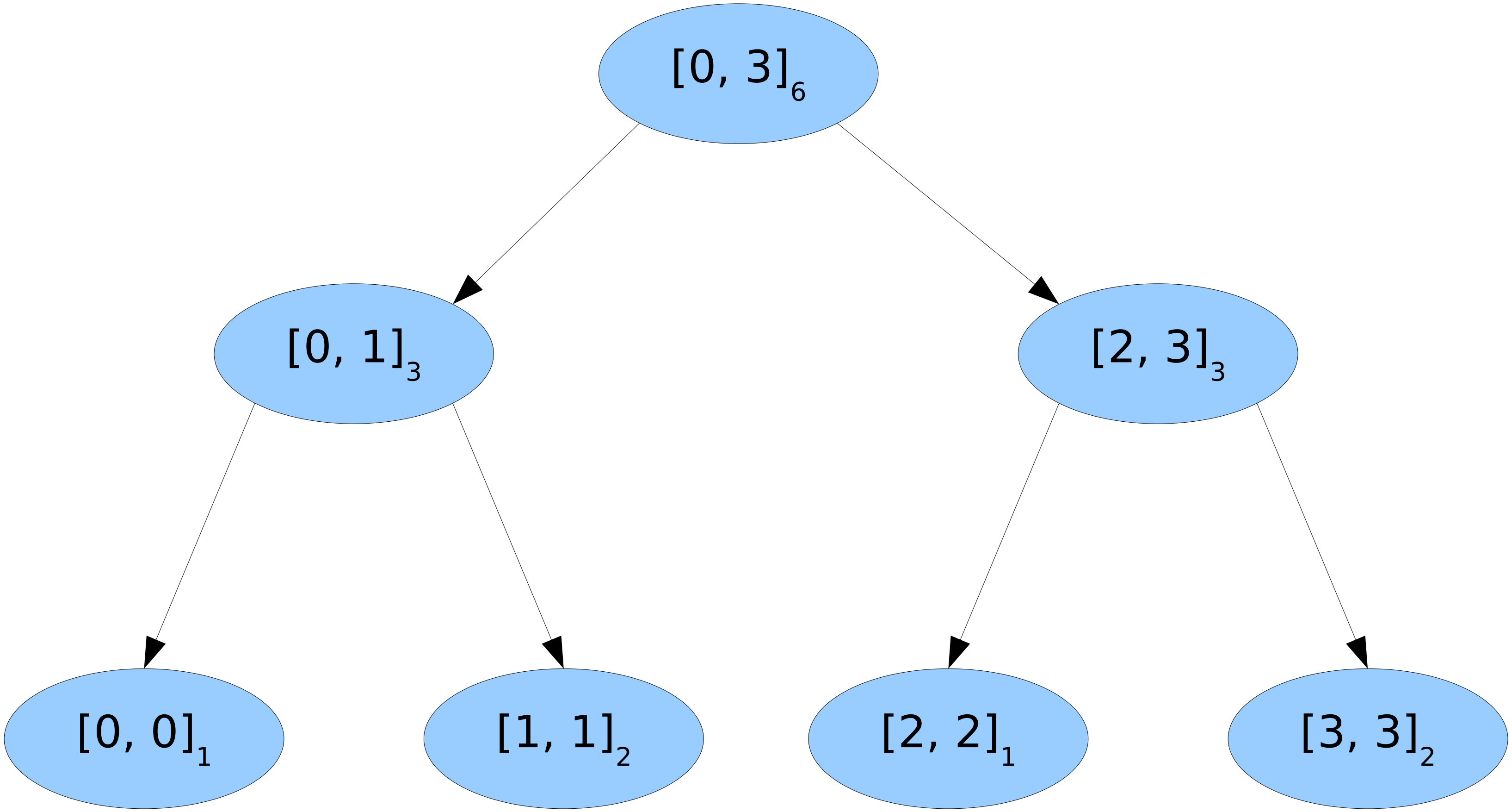
- At each node V_i, store the count o of interval $[0, V_i]$
- Problem: unbalanced trees:



Radix Tree

- Root node contains the interval [0, max]
- A node containing the interval [A, B] has children with interval [A, (A+B)/2] and [(A+B)/2 + 1, B]
- Each node holds the count over its range
- Every number has a well-defined position

Radix Tree: Example



Radix Tree: Update

- To insert the value V_i:
  - Increment the root node's count
  - If V_i <= (A+B)/2
    - Insert V_i into the left child
  - Else
    - Insert V_i into the right child
- Similar process for deletion

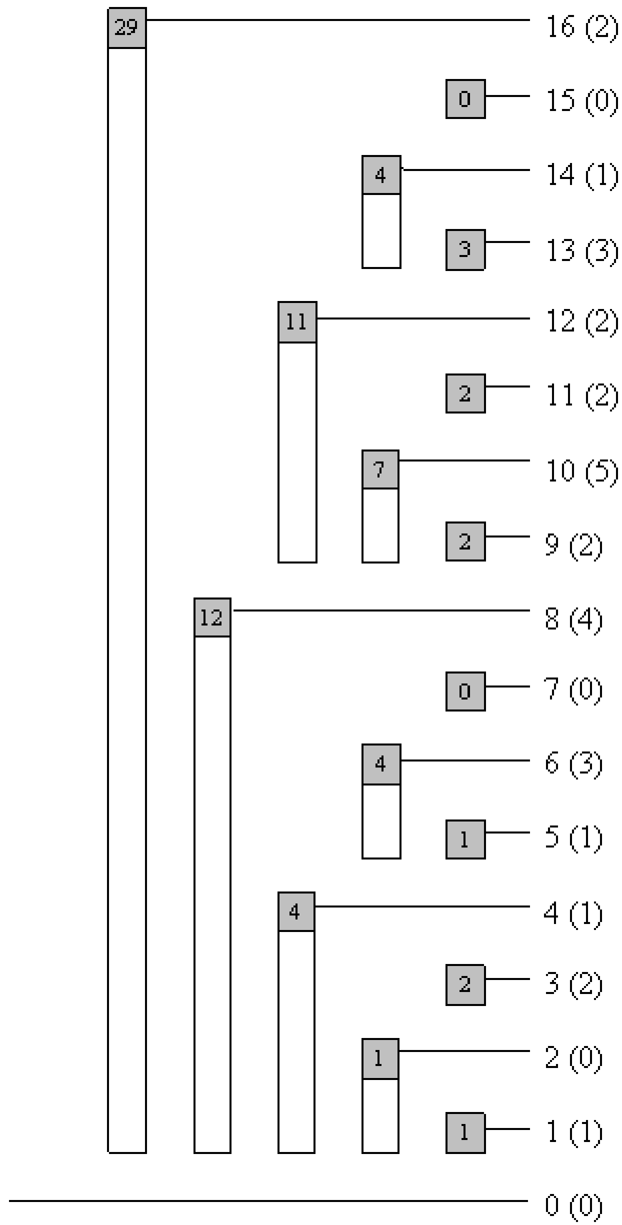
Radix Tree: Query

- To find the number of items in the interval [x, y]:
  - If [A, B] covers [x, y]
    - return root node's count
  - Let sum = 0
  - If [A, (A+B)/2] overlaps [x, y]
    - sum += count in left child over [x, y]
  - If [(A+B)/2 + 1, B] overlaps [x, y]
    - sum += count in right child over [x, y]
  - Return sum

Binary Indexed Trees

- Any number can be represented as the sum of powers of two
- Assume intervals are of the form [1, x]
- An interval count can be represented in a similar manner
- [1, 13] = [1, 8] + [9, 12] + [13, 13]
- So we only store counts of interval of the form $[x - 2^r + 1, x]$

Binary Indexed Trees: Example



Binary Indexed Trees

- $13 = 1101_2$
- c[1101] = tree[1101] + tree[1100] + tree[1000]
- To isolate last 1 in binary form: n & -n
  - Proof in reference article

Binary Indexed Trees: Update

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

Binary Indexed Trees: Query

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

Sample Problem: TopCoder SRM 310

- n cards face down on table
  - T i j: turn cards from index i to index j, include i-th and j-th card
  - Q i: is the i-th card face?)

Sample Problem: Solution

- Array f (of length n + 1) holds our BIT
- f[i]++ and f[j + 1]--
- For each card k between i and j, sum f[1] + f[2] + ... + f[k] will be incremented
- For each query, the answer is f[i] % 2

Analysis

- Binary Tree: common, but balancing can be difficult and expensive
- Radix Tree: guaranteed O(logN) for update and query
- Binary Index Tree: Same as Radix Tree, but very short code

Questions

?

References

- http://www.topcoder.com/tc?module=Static&d1=tutorials&d2=binaryIndexedTrees