

Point Location Using Persistence

John Iacono

Partial Persistent Dictionaries

1	Insert	50	50					
2	Insert	20	20	50				
3	Insert	40	20	40	50			
4	Delete	50	20	40				
5	Insert	25	20	25	40			
6	Insert	42	20	25	40	42		
7	Delete	25	20	40	42			
8	Insert	11	11	20	40	42		
9	Insert	75	11	20	40	42	75	
10	Insert	80	11	20	40	42	75	80
11	Delete	42	11	20	40	75	80	
12	Delete	20	11	40	75	80		
13	Insert	15	11	15	40	75	80	
14	Delete	15	11	15	75	80		
15	Delete	15	11	75	80			

IDEA: Be able to query
in the past as fast as in
the present

† Don't use much space

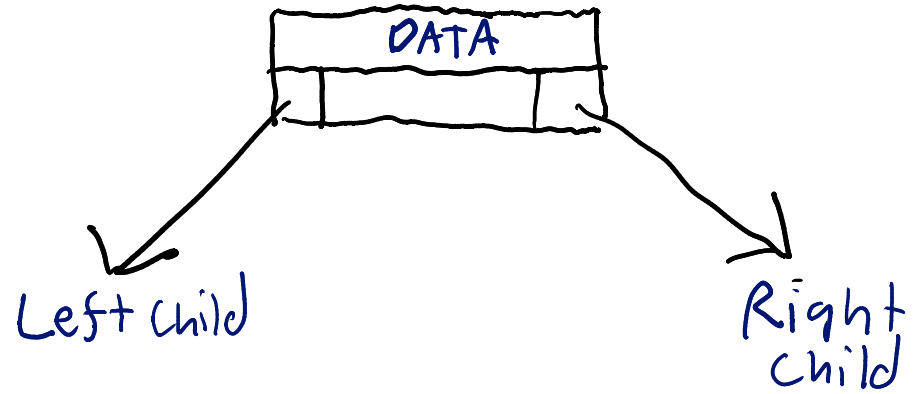
$\text{Pred}(50)$ at time 7 = 40

$\text{Pred}(50) = 11$

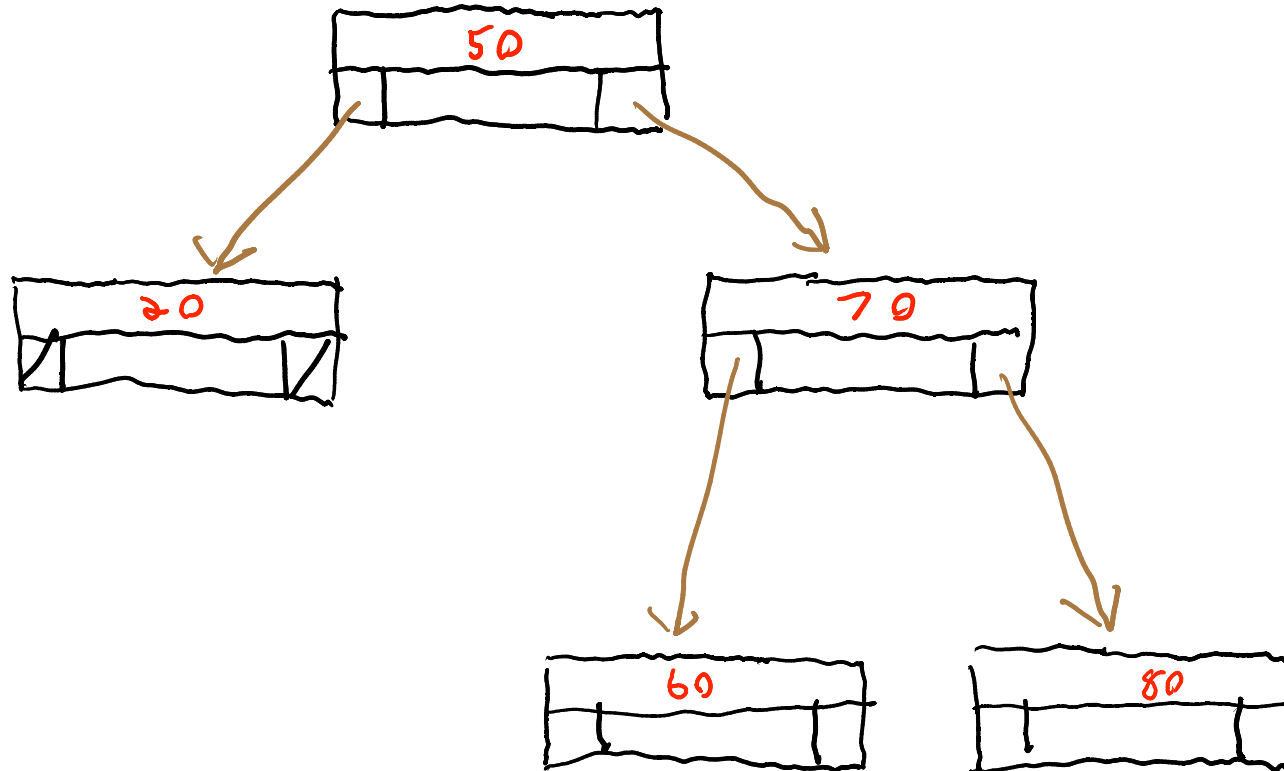
Partial Persistent Binary Trees

- Want to be able to make changes to the most current tree
- Want to be able to "see" any old version of the tree at a cost of $\log(\# \text{ of versions})$

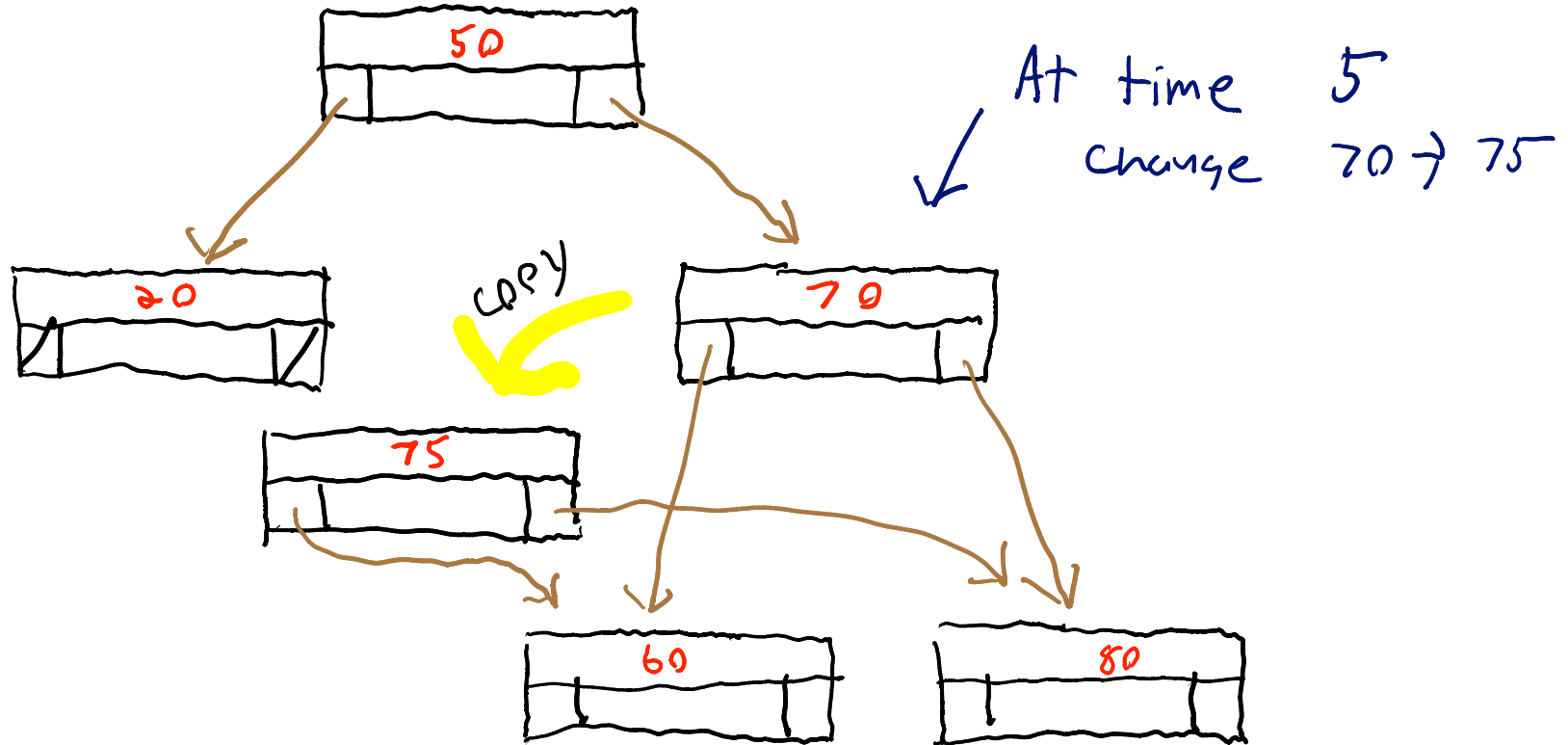
IDEA



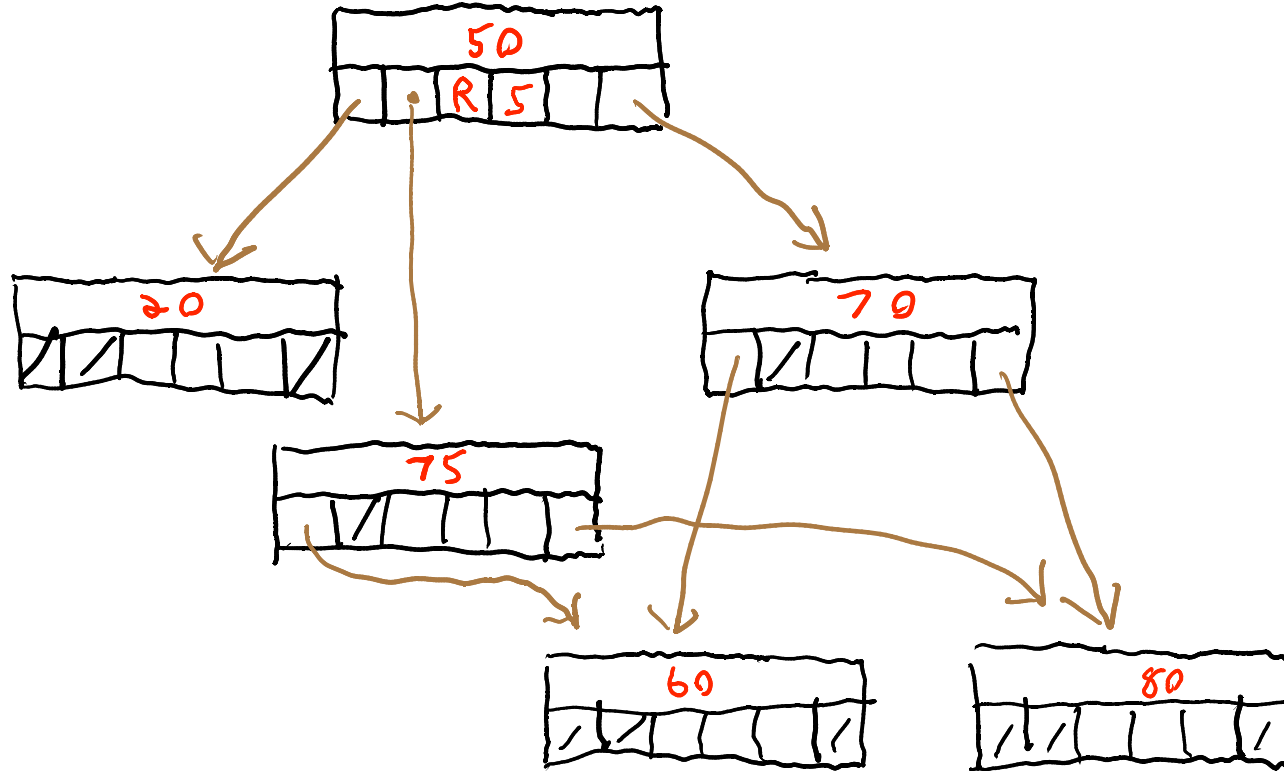
IDEA



IDEA

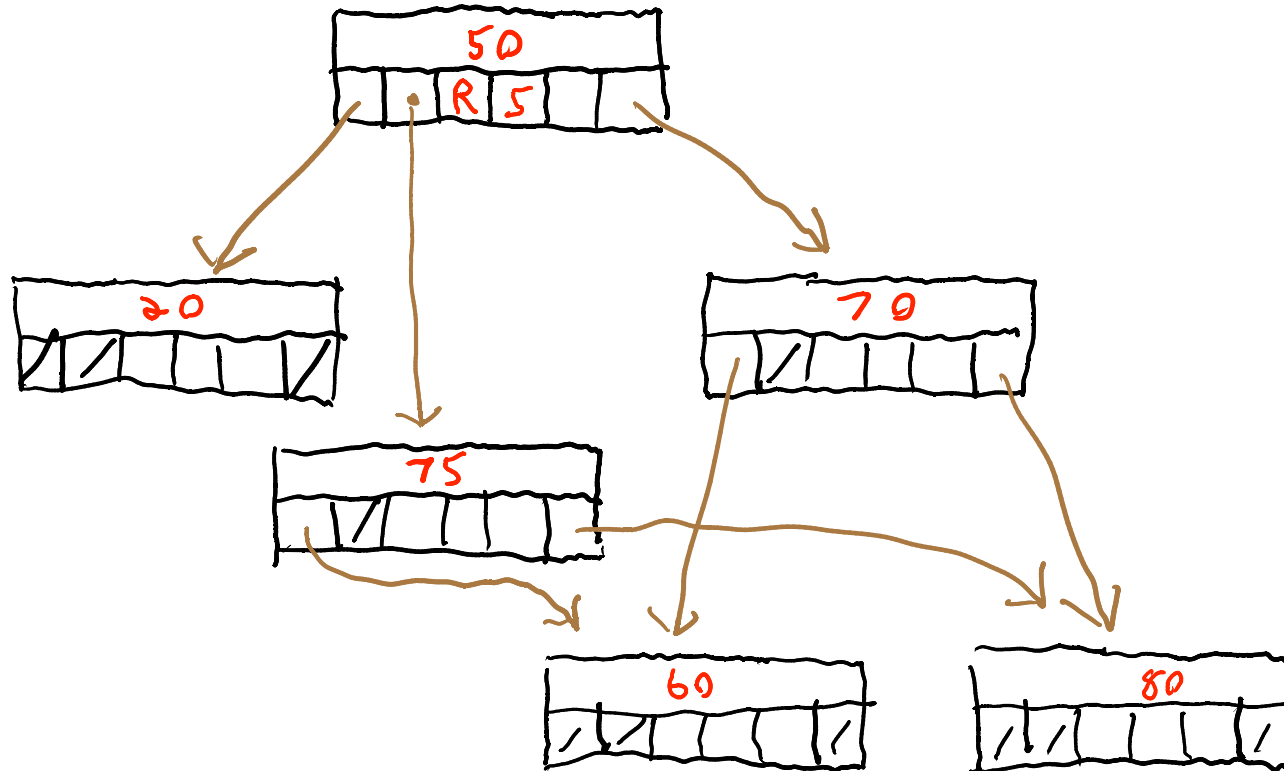


IDEA



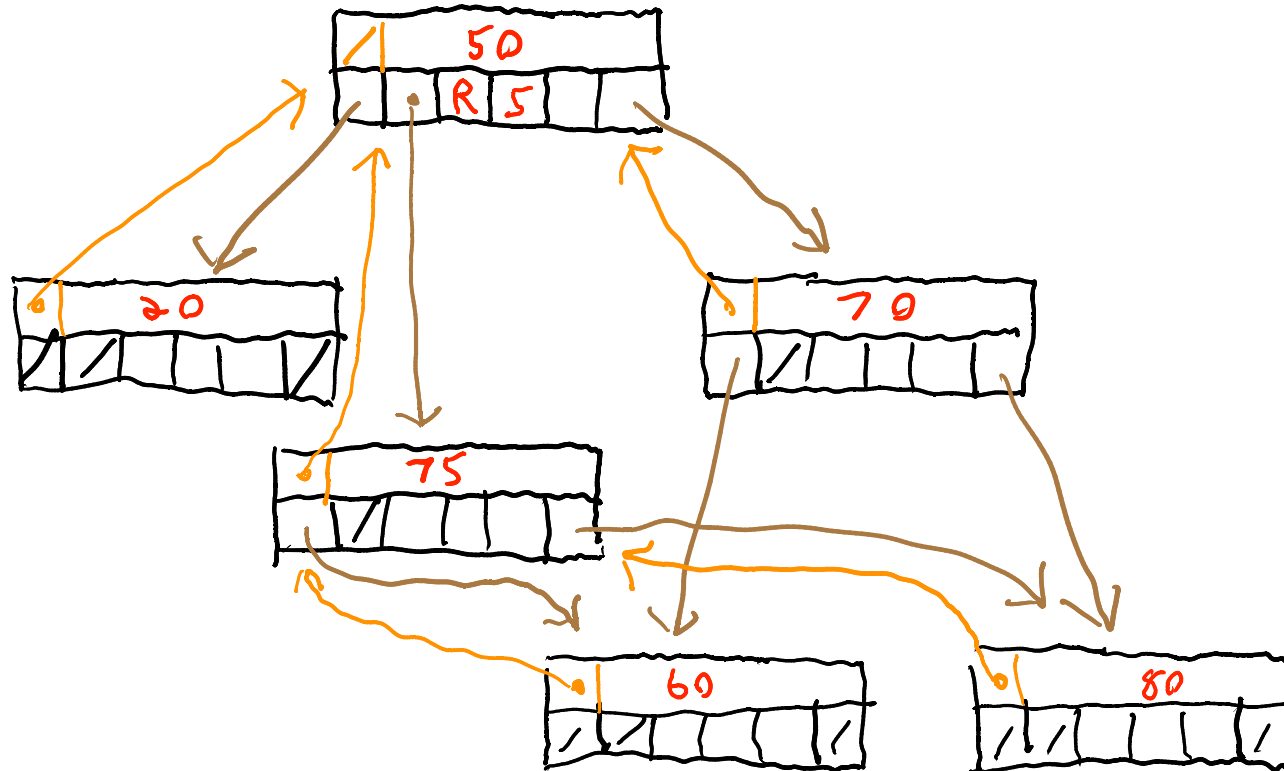
IDEA

Will need current
parent pointers

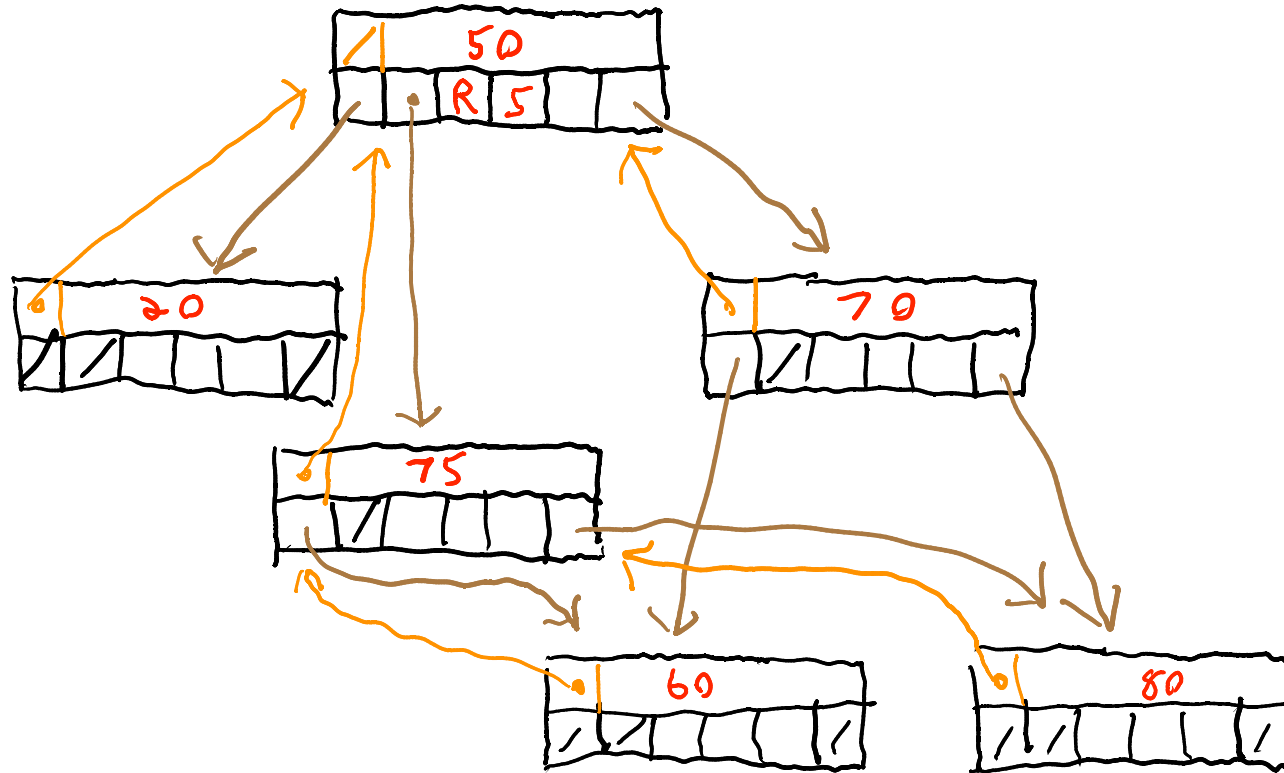


IDEA

Will need current
parent pointers

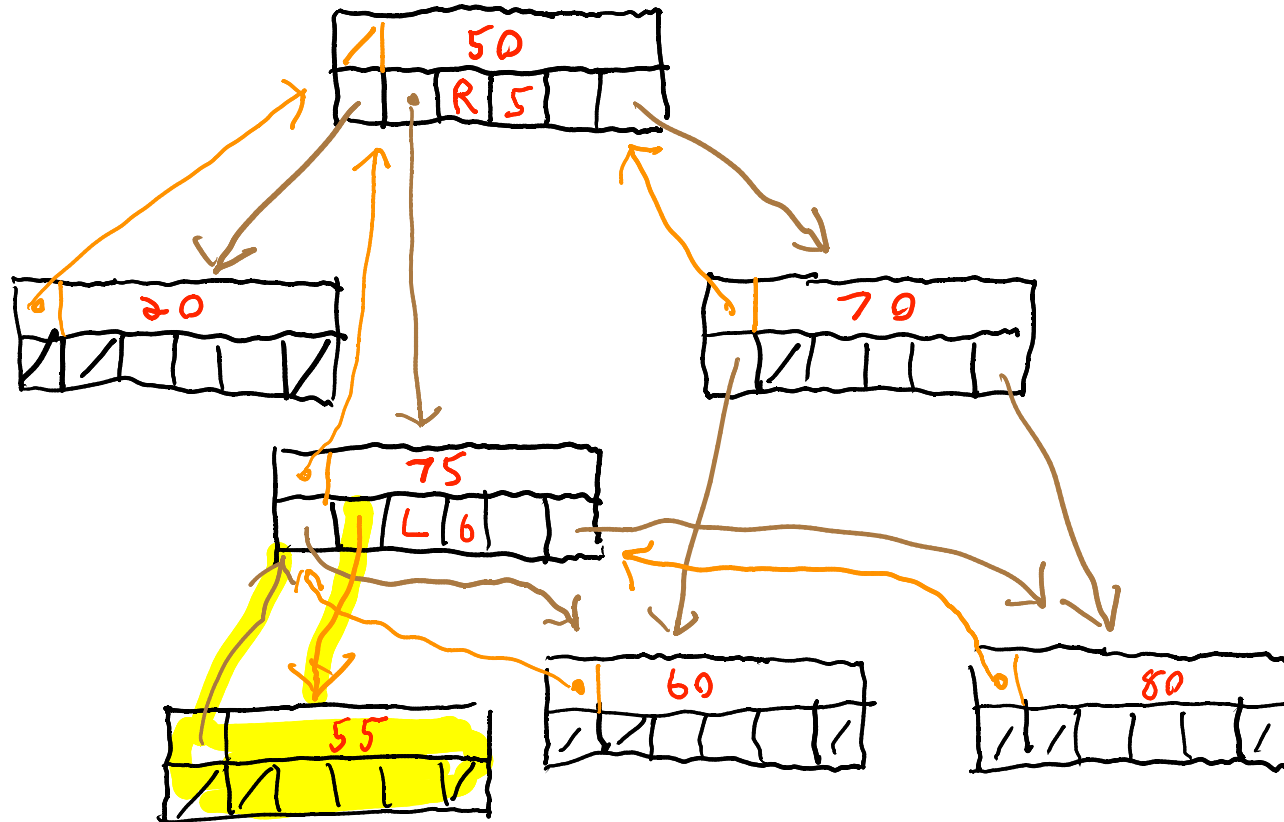


IDEA



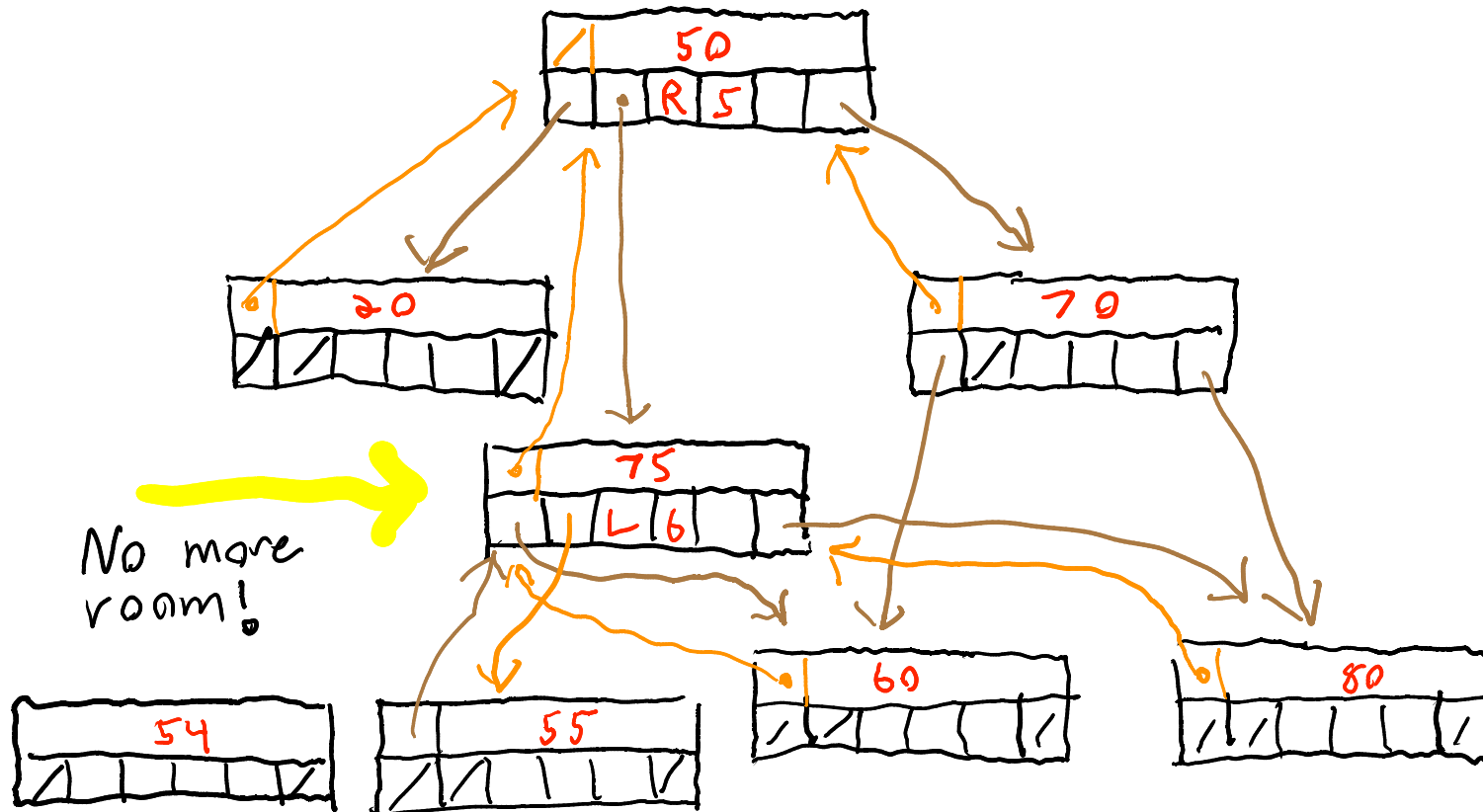
Change the Left child of 75 to 55

IDEA



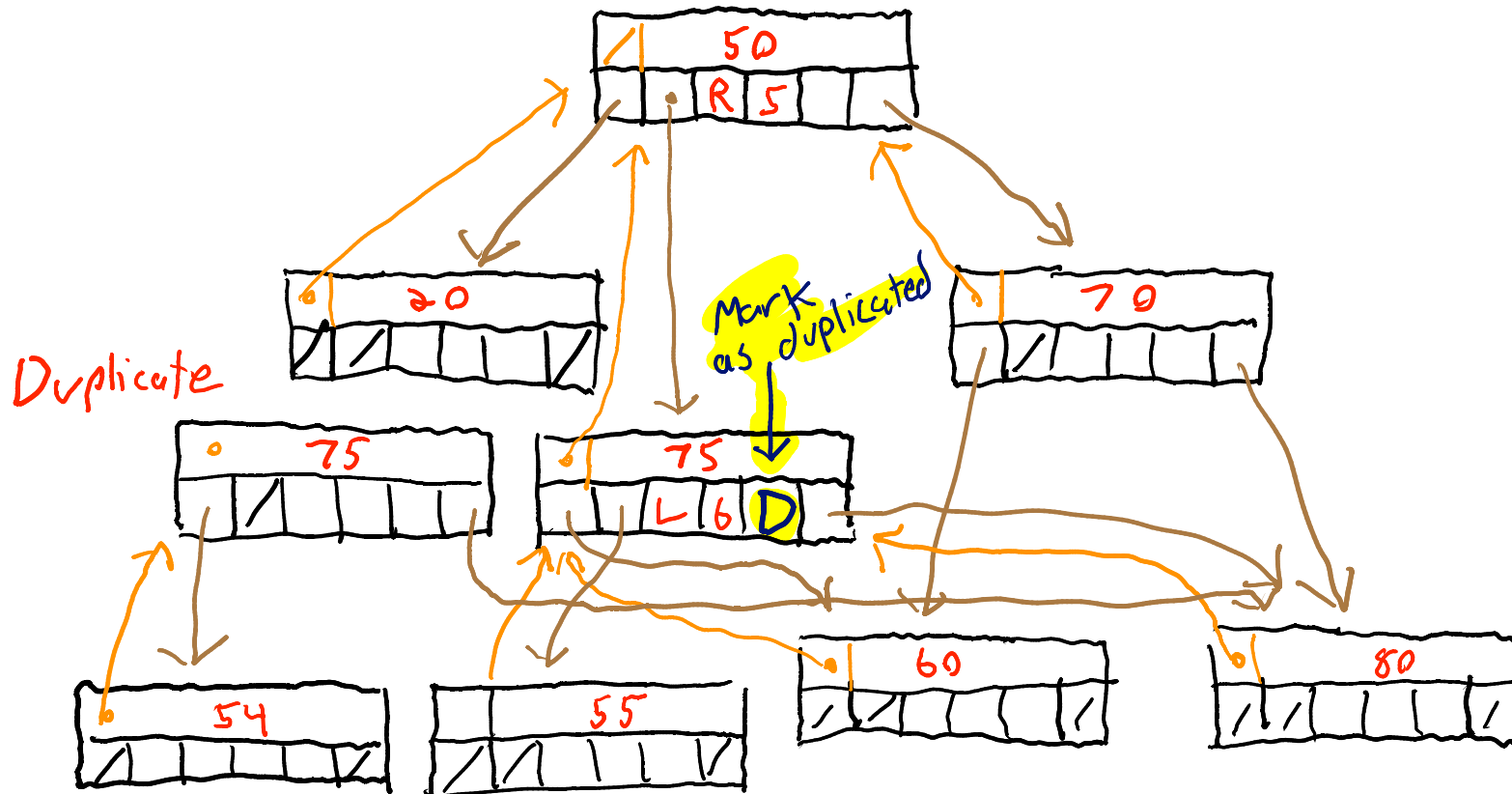
Change the Left child of 75 to 55
at time 6

IDEA



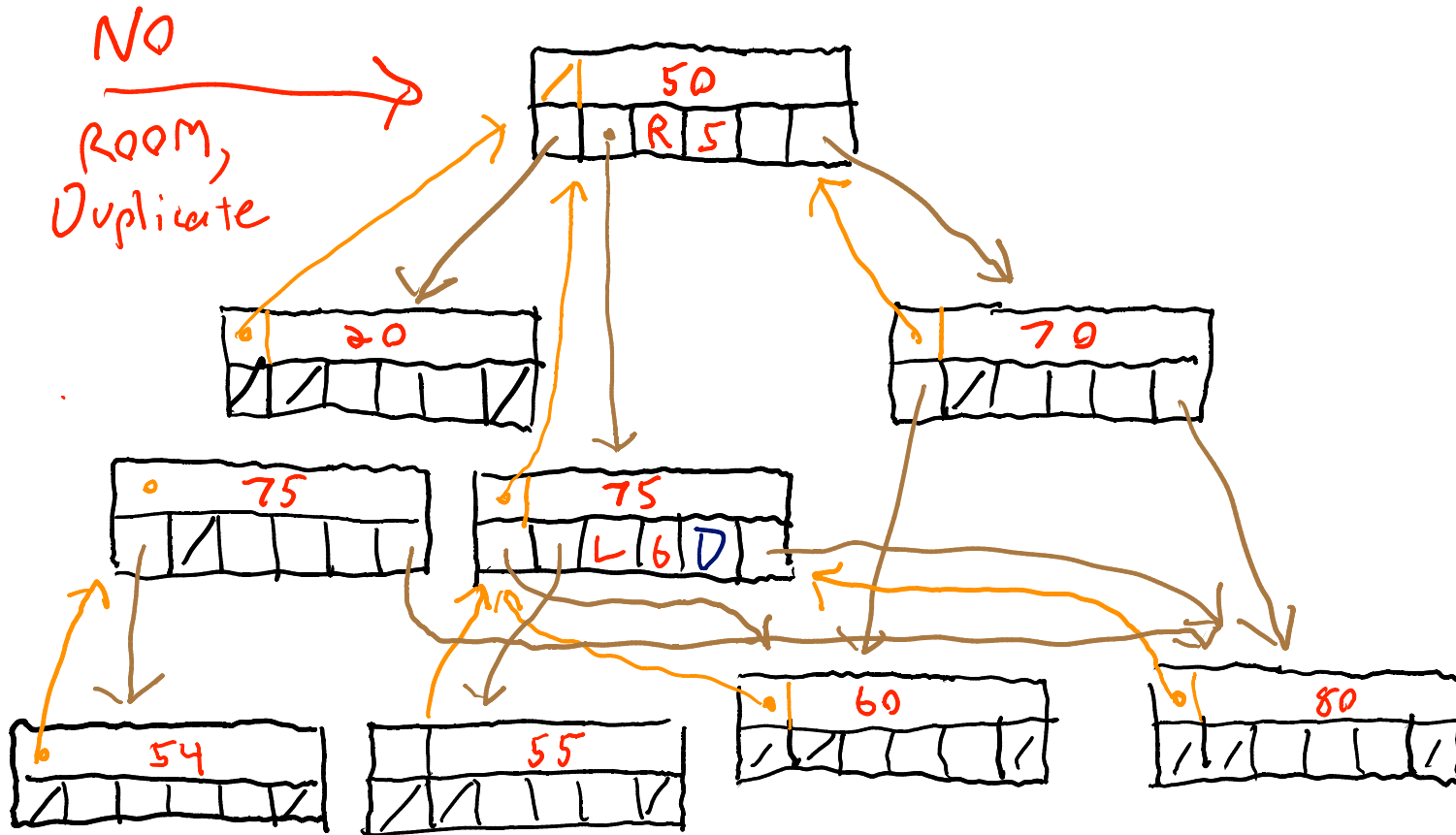
Change the Left child of 75 to 54
at time 7

IDEA

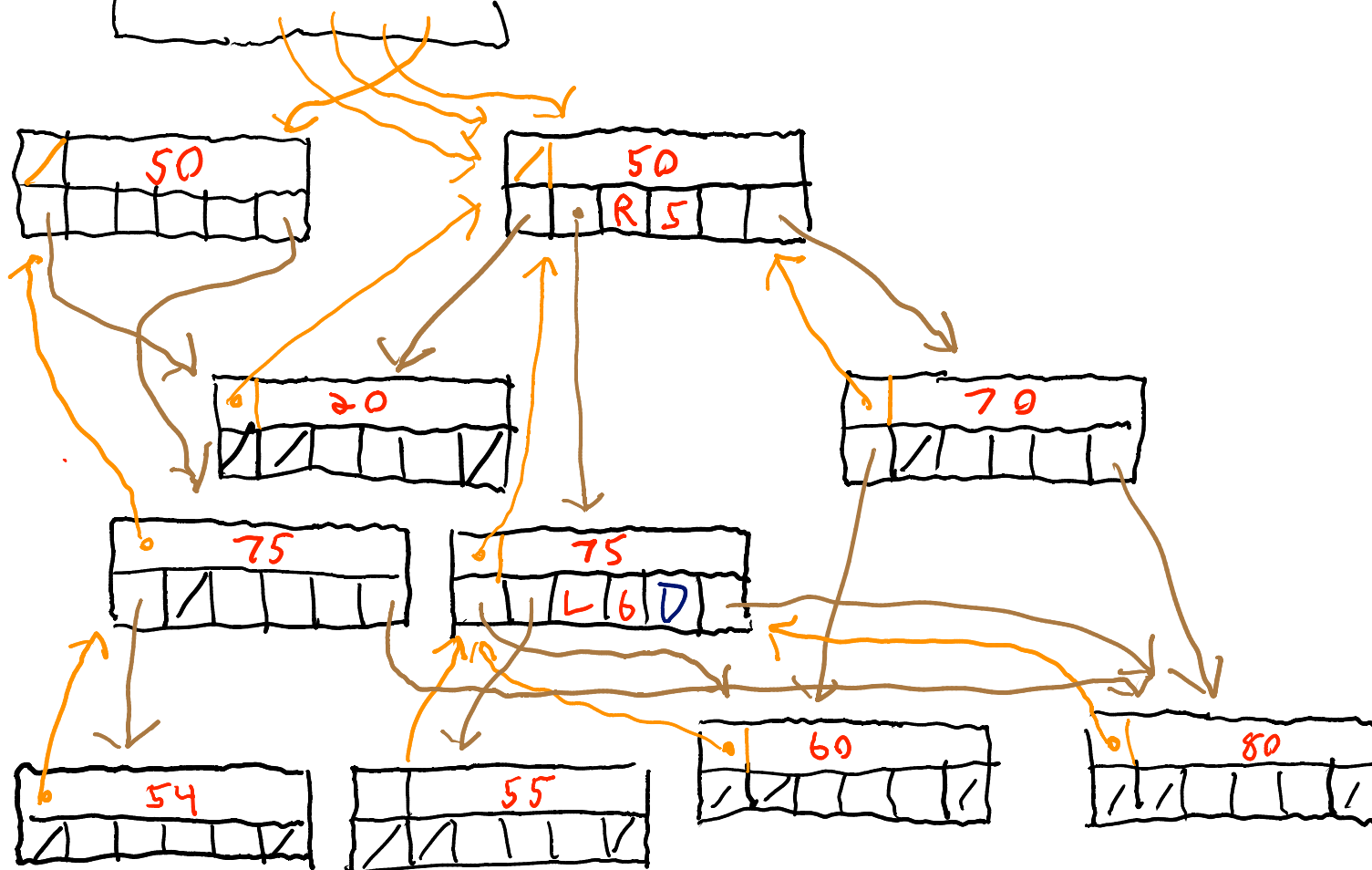


Change the Left child of 75 to 54
at time 7

IDEA



Change the Left child of 75 to 54
at time 7



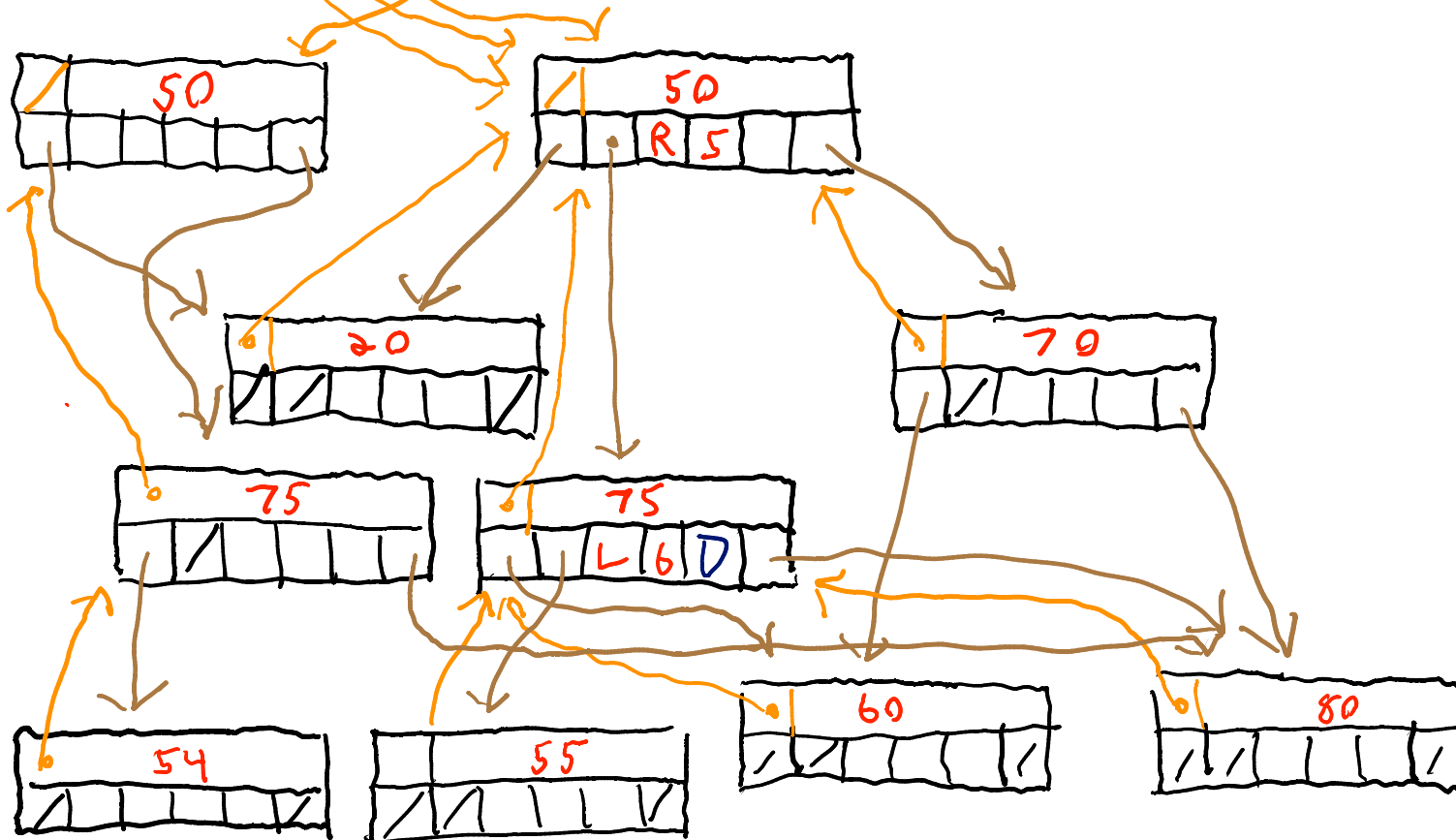
Change the Left child of 75 to 54
at time 7

Root array



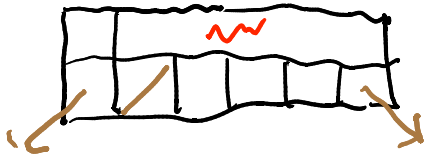
$O(1)$ Access to latest root

$O(\log \# \text{versions})$ to any root

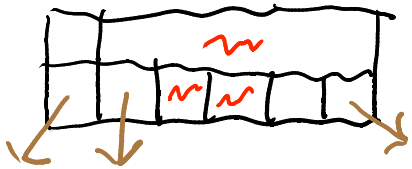


Change the Left child of 75 to 54
at time 7

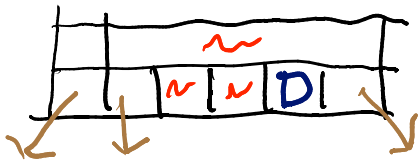
Runtime — Amortized Analysis



Potential: 0



Potential: 1



Potential: 0

Duplicating a node costs 1, reduces potential by 1

∴ Amortized time and space usage by a single change to the tree is $O(1)$

Persistent Dictionaries

- Use Red-Black trees
- $O(\log n)$ changes per Insert/Delete
- If there are n Insert/Deletes the search cost is

$$O(\log n + \log n) = O(\log n)$$

↑ versions ↑ normal search cost

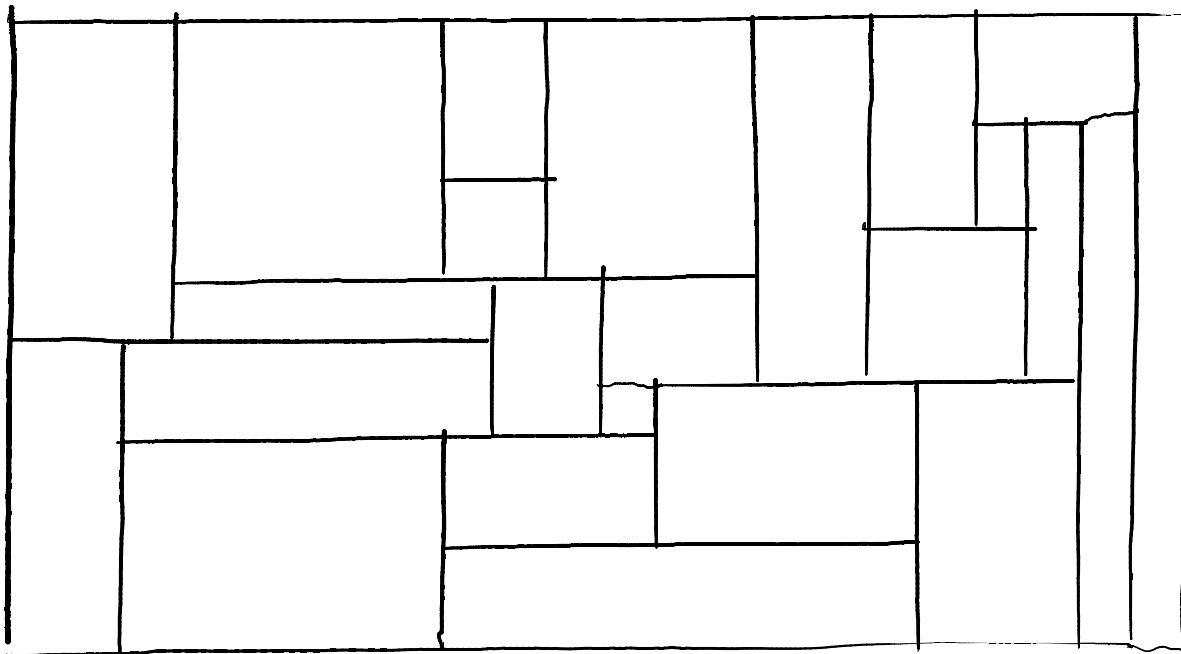
- Amortized space per insert is $O(\log n)$
- Total space usage is $O(n \log n)$

Persistent Dictionaries

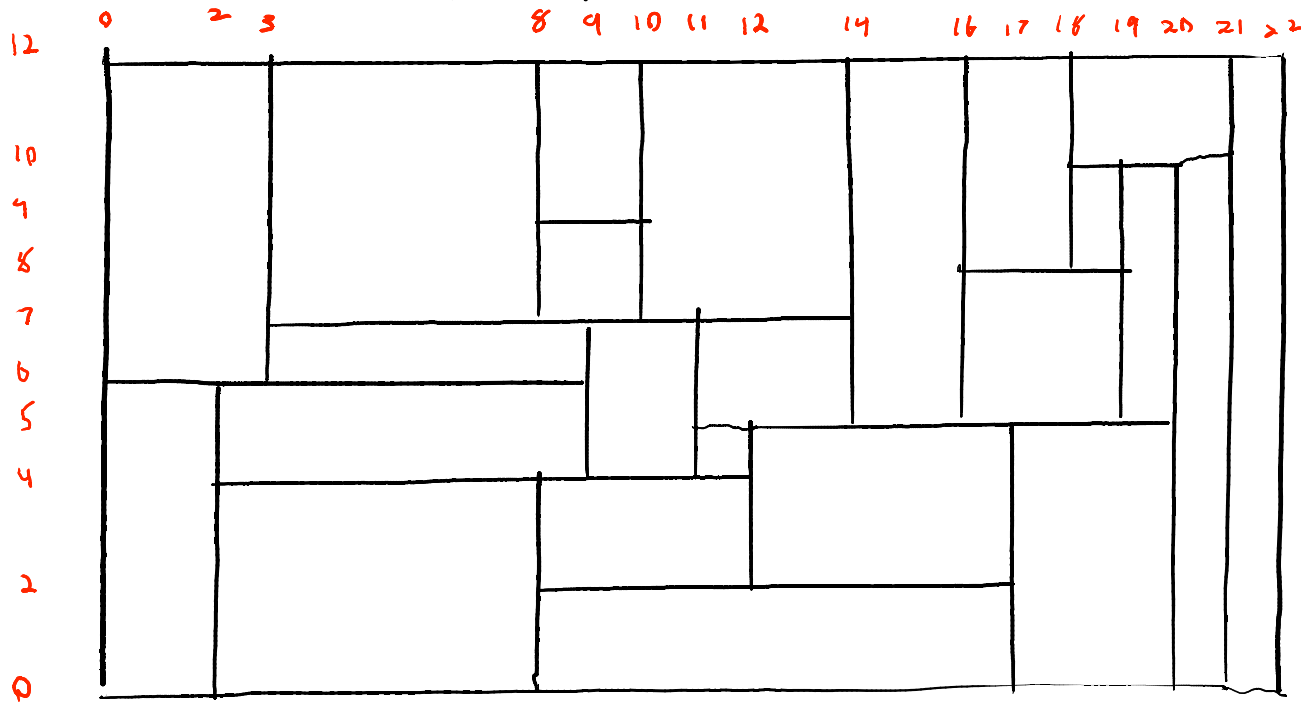
- Use Red-Black trees
- $O(\log n)$ changes per Insert/Delete
- If there are n Insert/Deletes the search cost is
$$O(\log n + \log n) = O(\log n)$$

↑ versions ↑ normal search cost
- Amortized space per insert is $O(\log n)$
- Total space usage is $O(n \log n)$
- But, red black trees do only 2 rotations per insert/delete, $O(\log n)$ recolorings,
- Don't need to remember recolorings to search,
- Thus only $O(1)$ changes per INS/DEL that need to be remembered
→ $O(n)$ space

Point Location

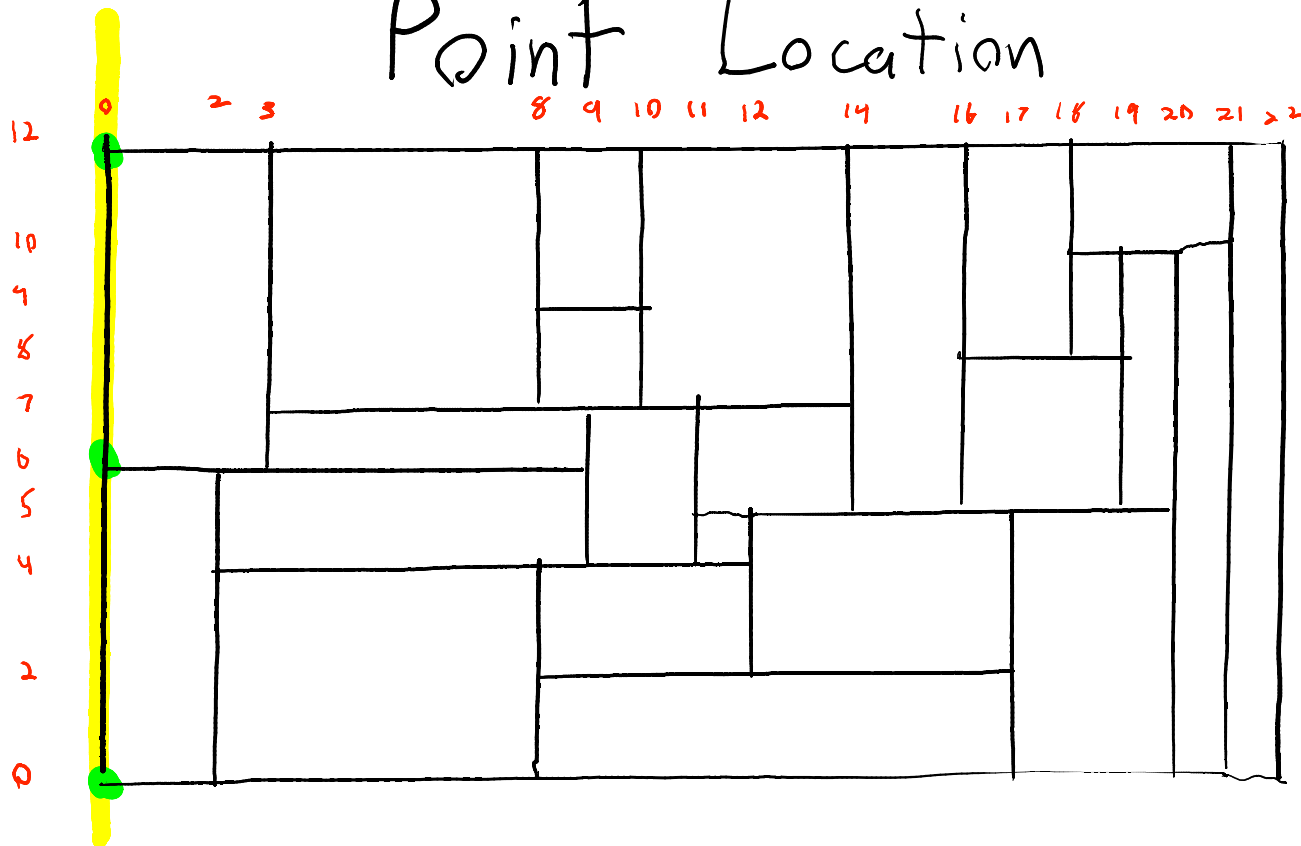


Point Location



sort by x coordinate, sweep left to right with a

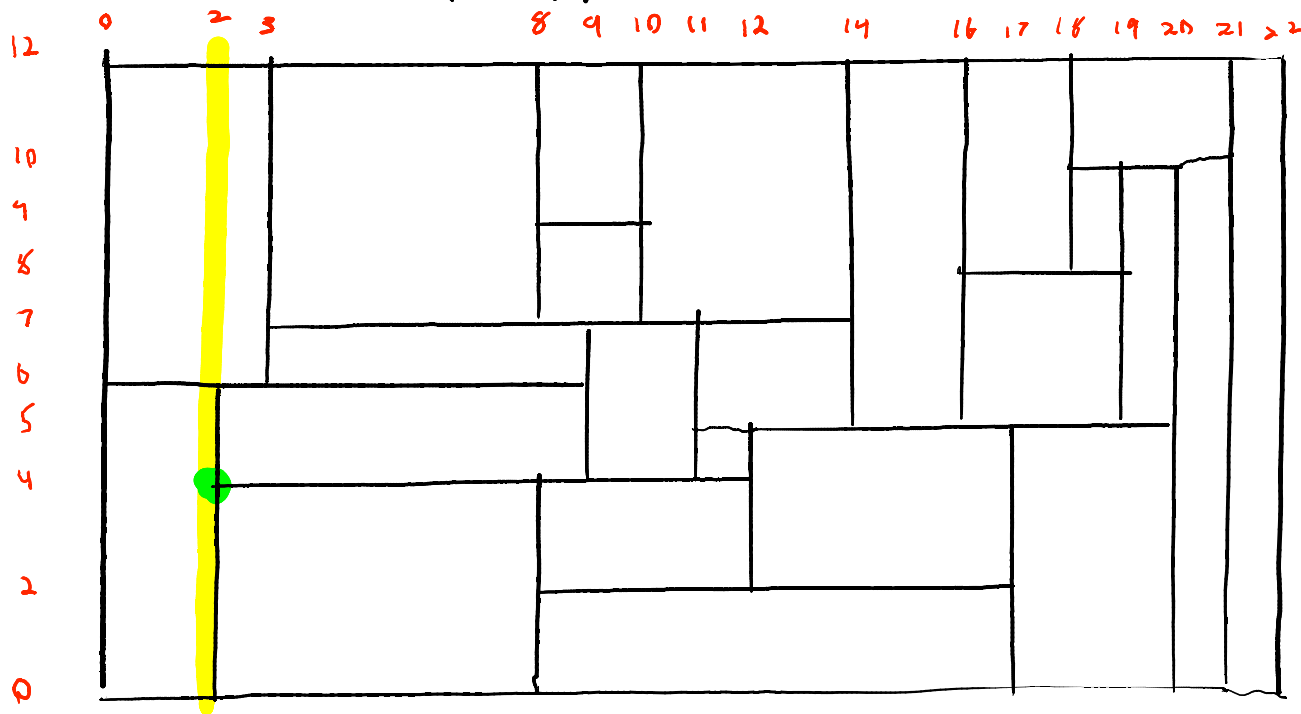
Point Location



0 I 0
I 6
I 12

sort by x coordinate, sweep left to
right with a

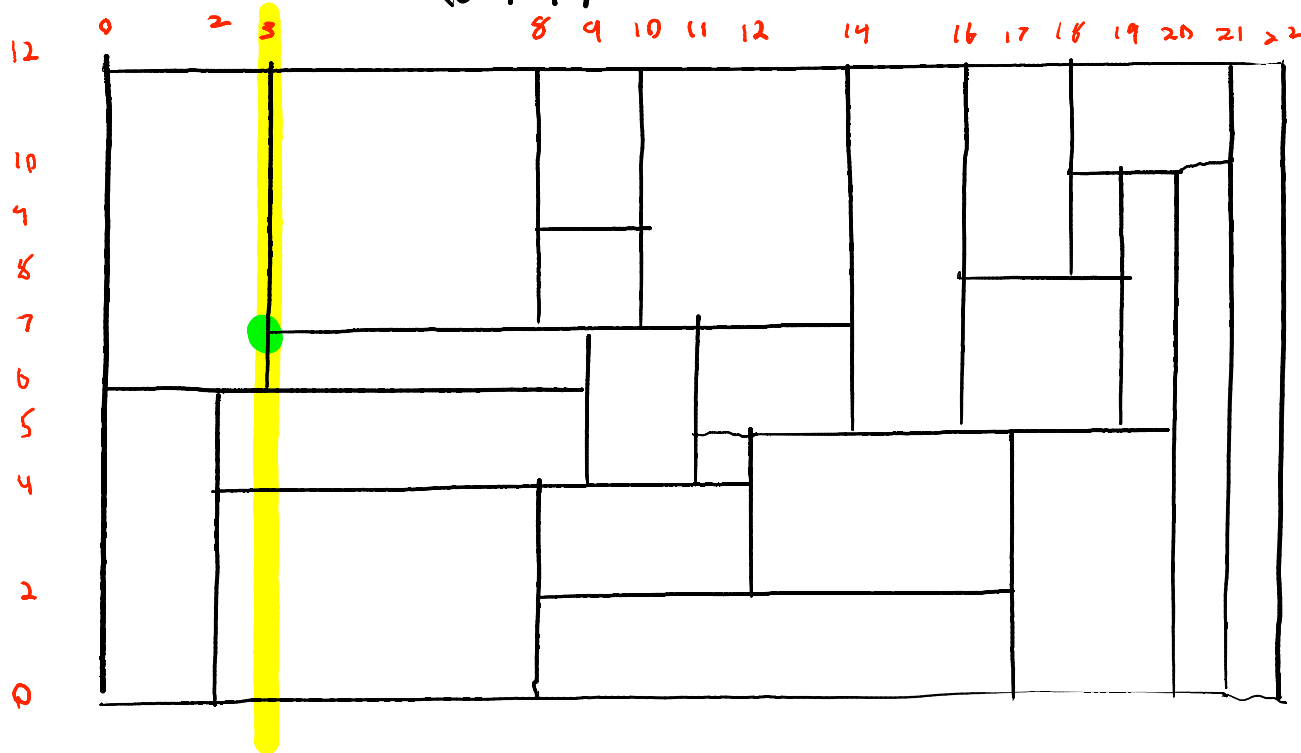
Point Location



0 I 0
 I 6
 I 12
 2 I 4

sort by x coordinate, sweep left to
 right with a

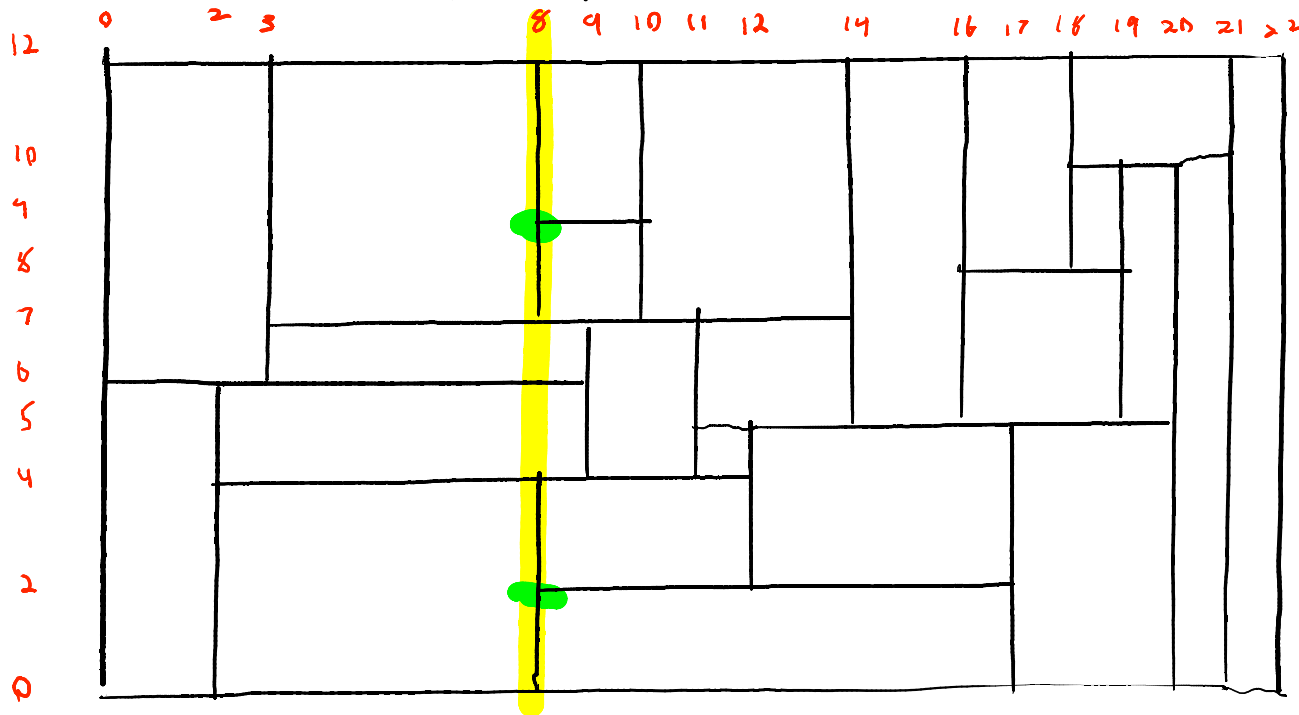
Point Location



0 I 0
 I 6
 I 12
 1 I 4
 3 I 7

sort by x coordinate, sweep left to right with a

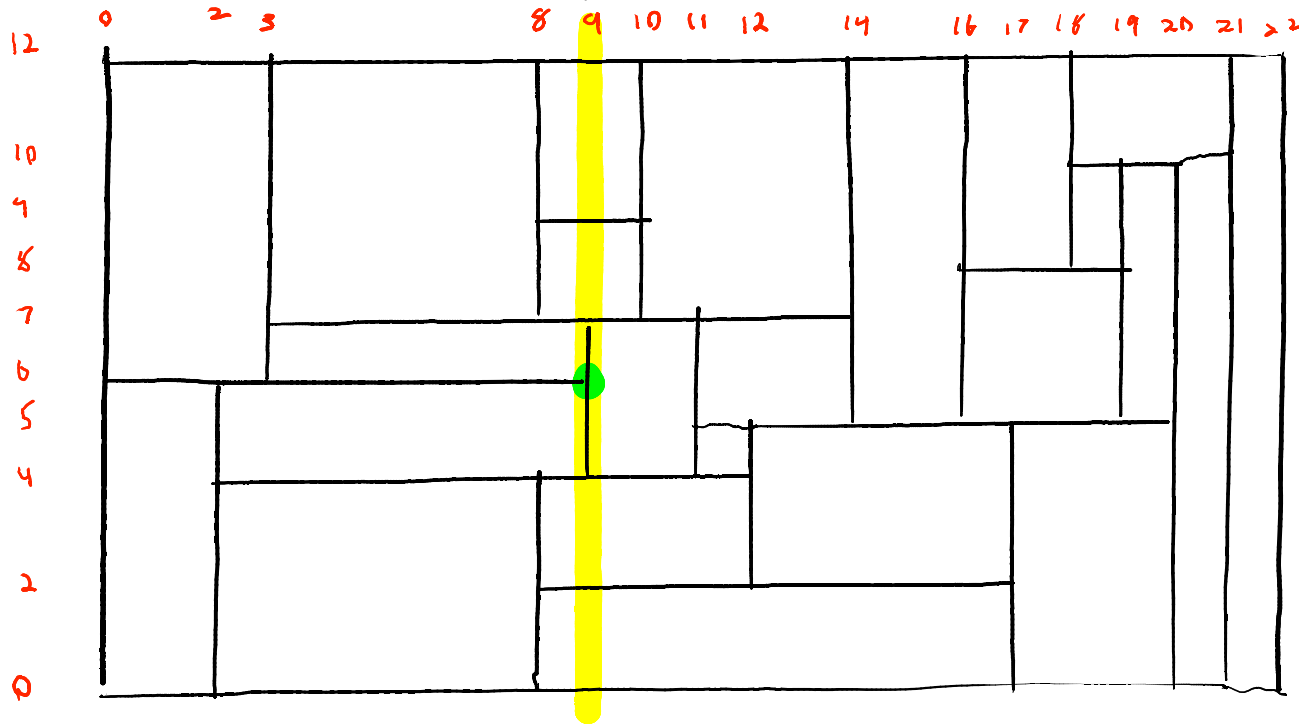
Point Location



0	I	0
	I	6
	I	12
1	I	4
3	I	7
8	I	2
	I	9

sort by x coordinate, sweep left to
right with a

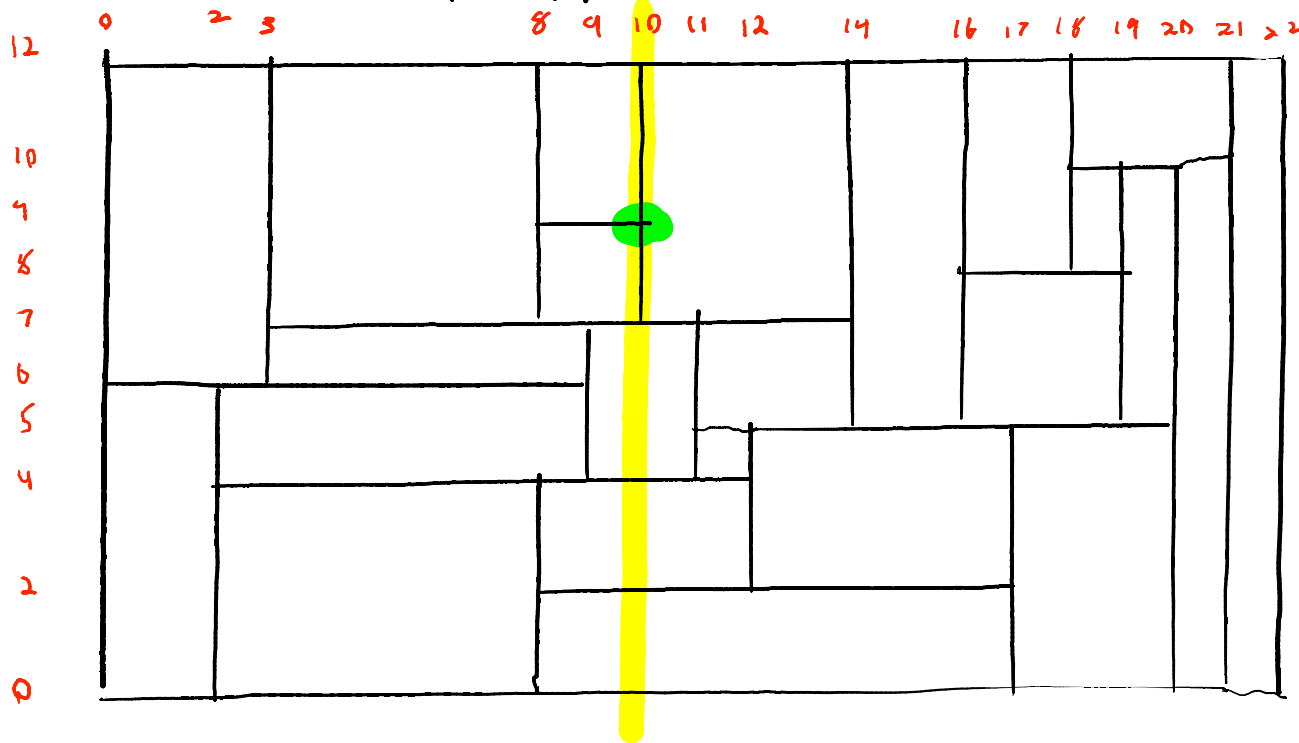
Point Location



0	I	0
	I	6
	I	12
1	I	4
3	I	7
8	I	2
	I	9
9	D	6

sort by x coordinate, sweep left to right with a

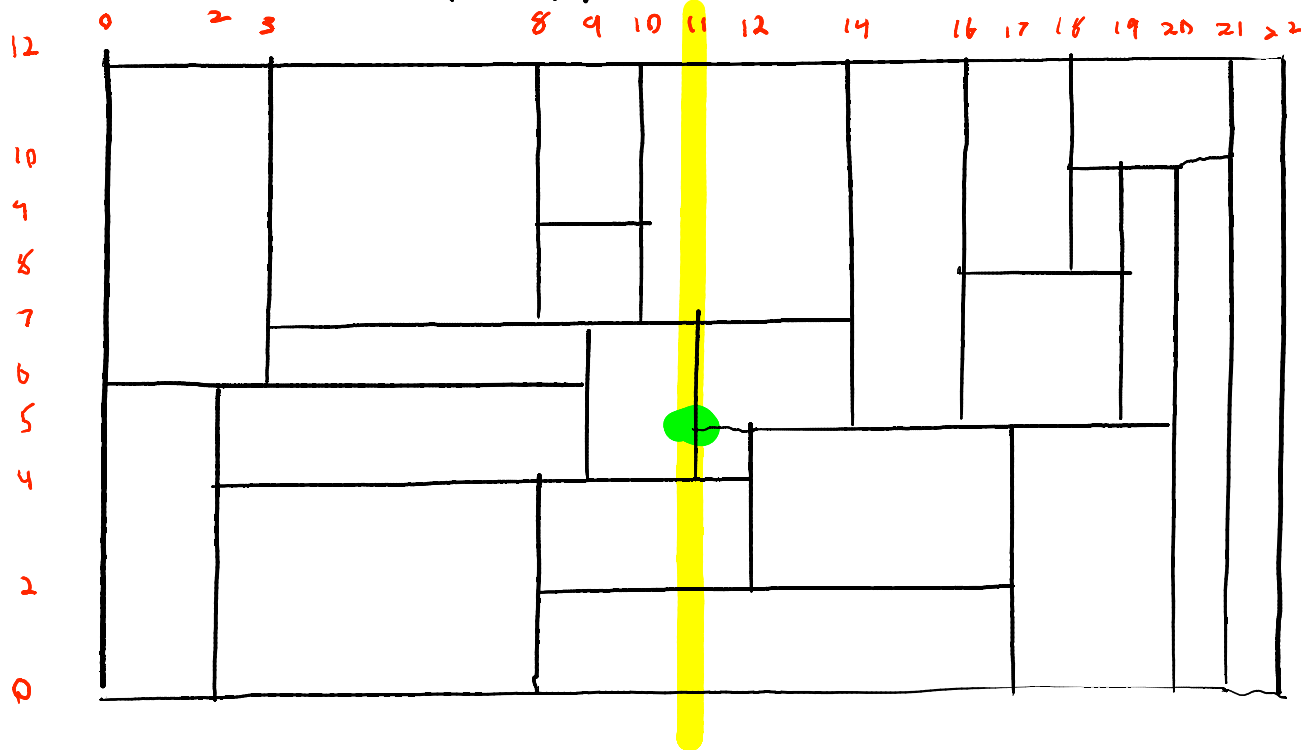
Point Location



0	I	0
	I	6
	I	12
1	I	4
3	I	7
8	I	2
	I	9
9	D	6
10	D	9

sort by x coordinate, sweep left to right with a

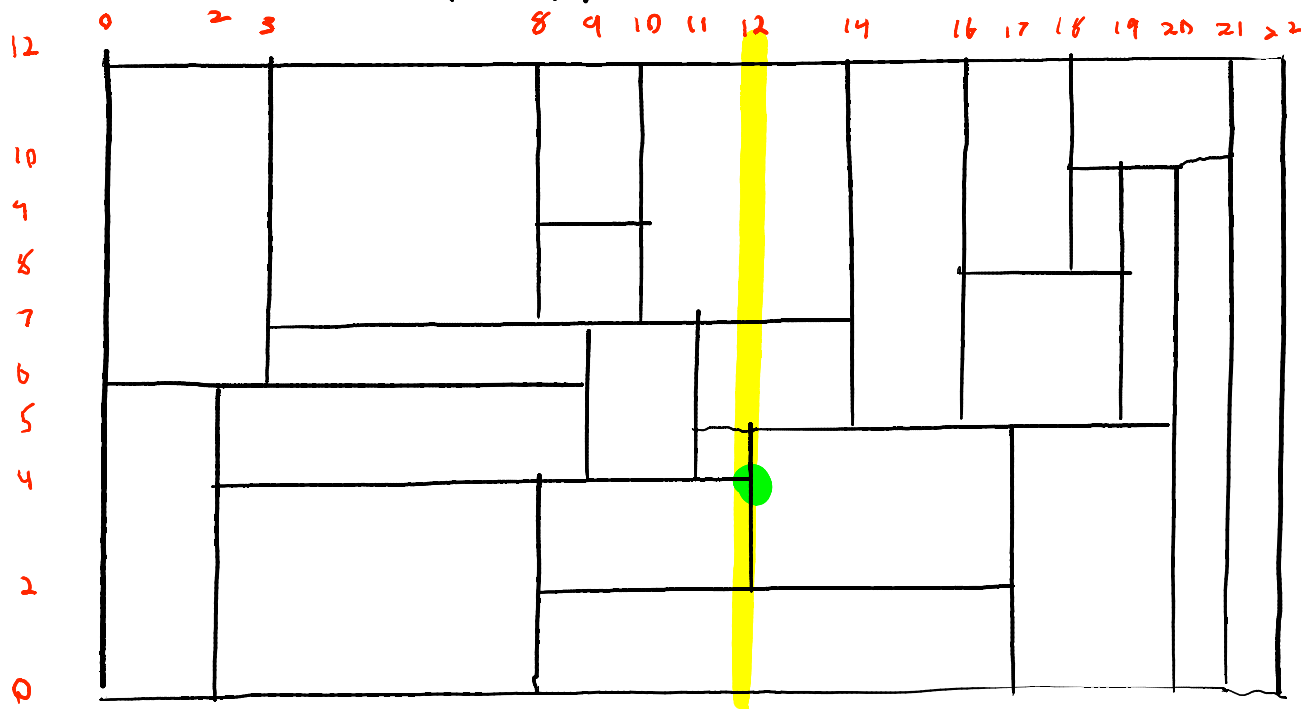
Point Location



0	I	0
	I	6
	I	12
1	I	4
3	I	7
8	I	2
	I	9
9	O	6
10	O	9
11	I	5

sort by x coordinate, sweep left to right with a

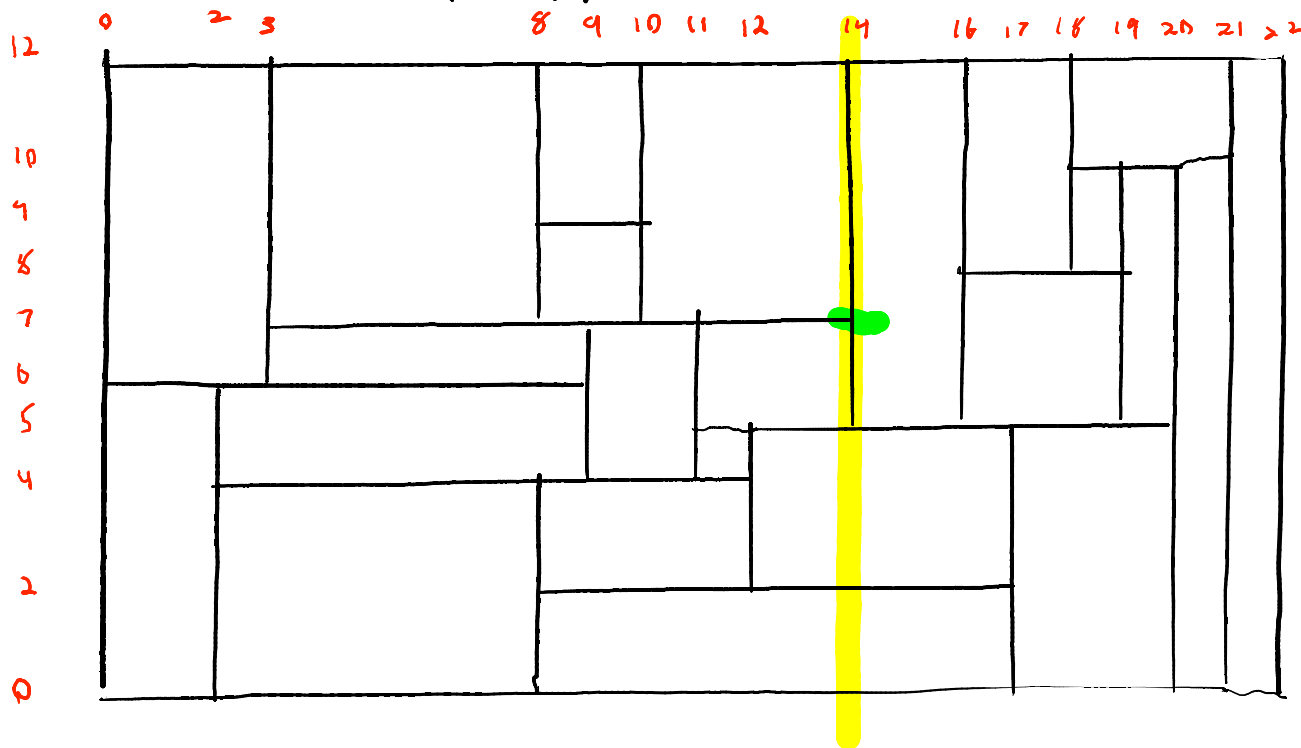
Point Location



0	I	0
	I	6
	I	12
1	I	4
3	I	7
8	I	2
	I	9
9	D	6
10	D	9
11	I	5
12	D	4

sort by x coordinate, sweep left to right with a

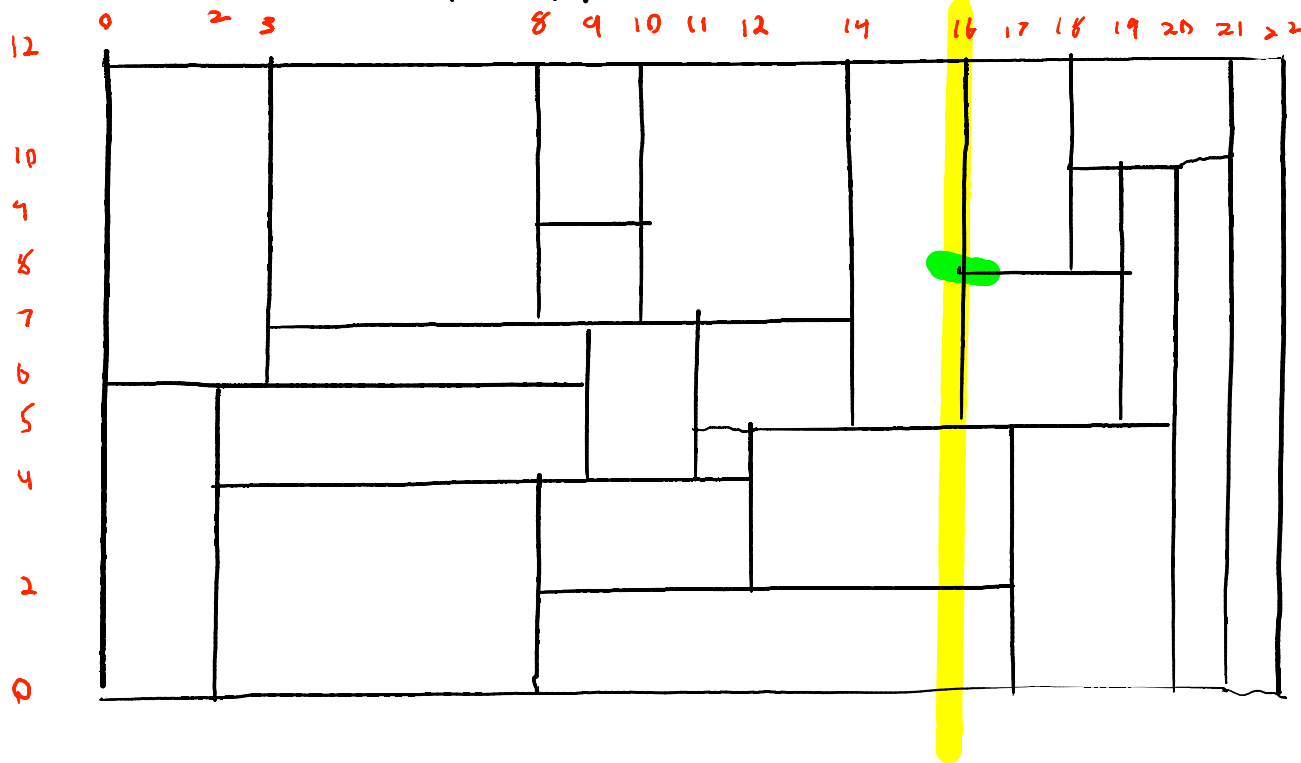
Point Location



0	I	0
	I	6
	I	12
1	I	4
3	I	7
8	I	2
	I	9
9	D	6
10	D	9
11	I	5
12	D	4
14	D	7

sort by x coordinate, sweep left to right with a

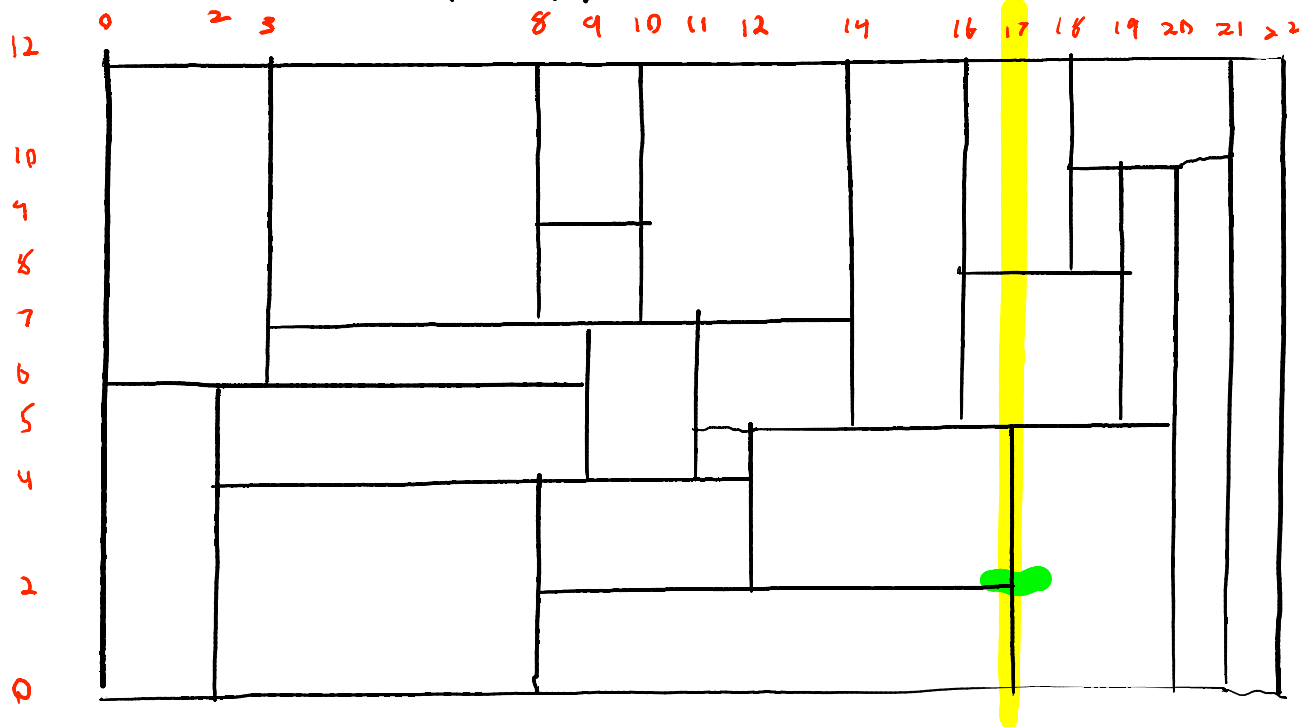
Point Location



0	I	0
	I	6
	I	12
1	I	4
3	I	7
8	I	2
	I	9
9	D	6
10	D	9
11	I	5
12	D	4
14	D	7
16	I	8

sort by x coordinate, sweep left to right with a

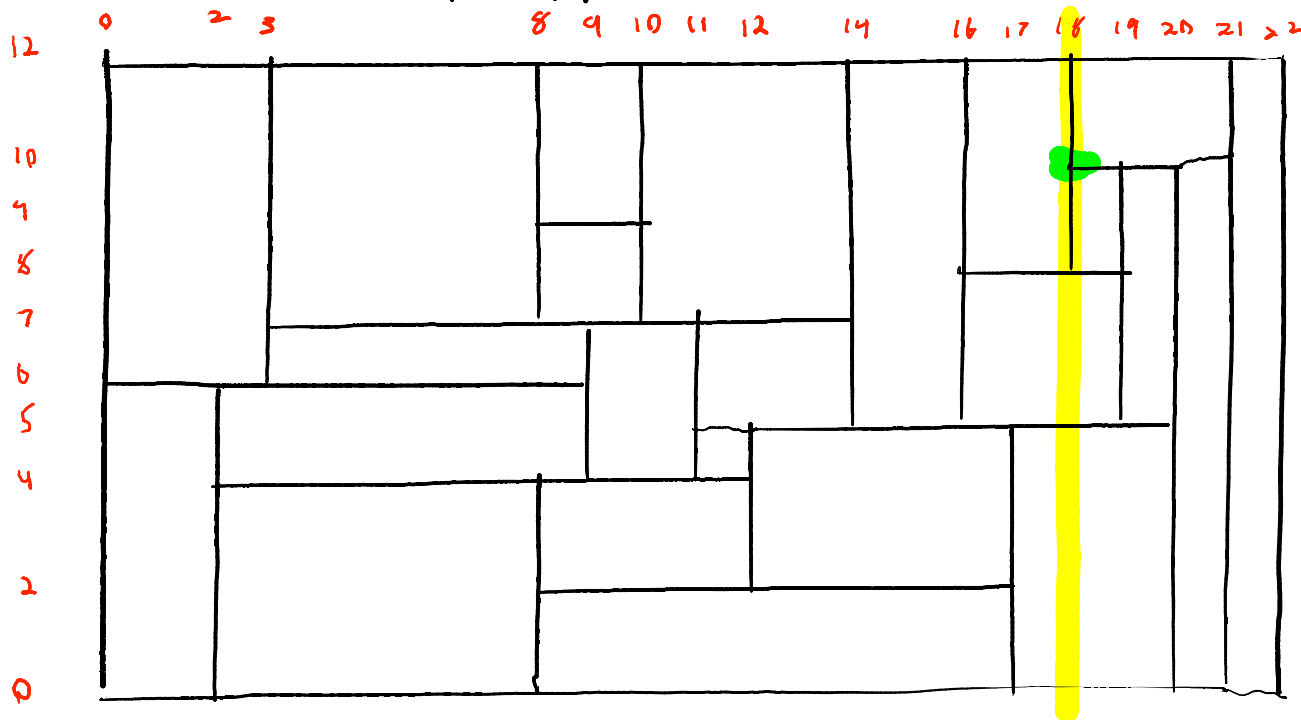
Point Location



0	I	0
	I	6
	I	12
1	I	4
3	I	7
8	I	2
	I	9
9	D	6
10	D	9
11	I	5
12	D	4
14	D	7
16	I	8
17	D	2

sort by x coordinate, sweep left to right with a

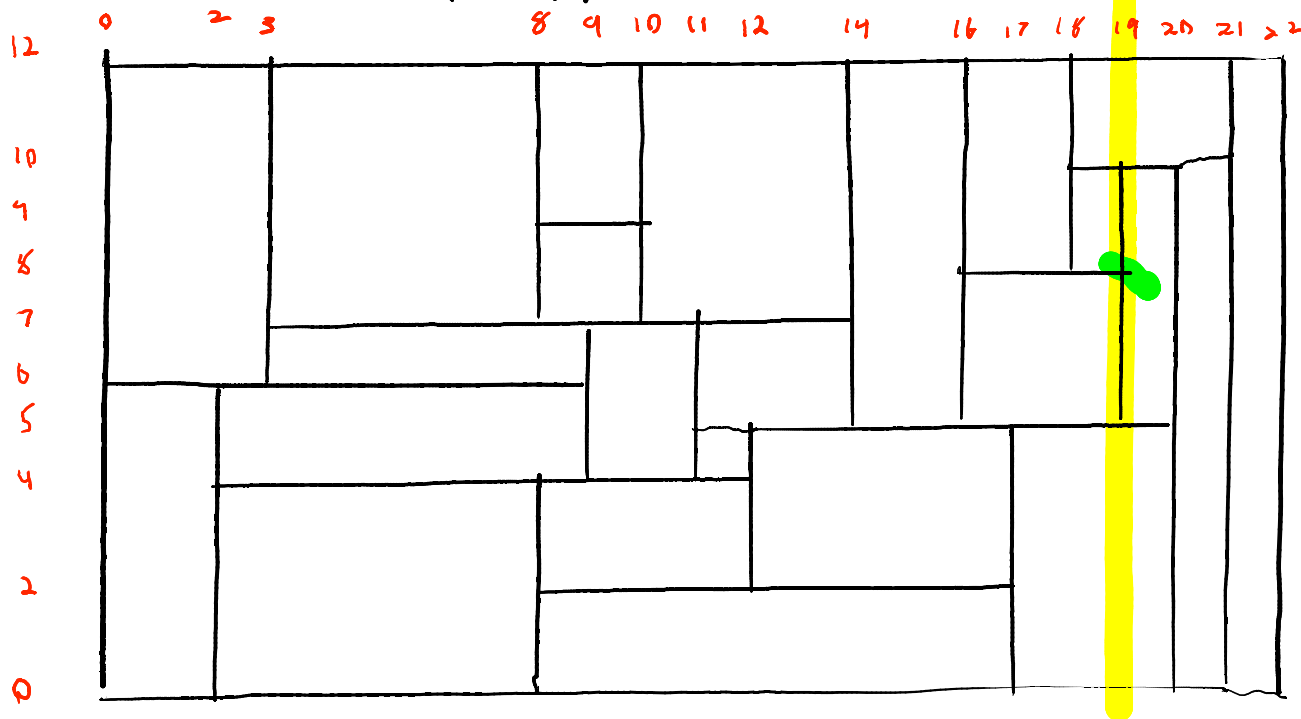
Point Location



0	I	0
	I	6
	I	12
1	I	4
3	I	7
8	I	2
	I	9
9	D	6
10	D	9
11	I	5
12	D	4
14	D	7
16	I	8
17	D	2
18	I	10

sort by x coordinate, sweep left to right with a

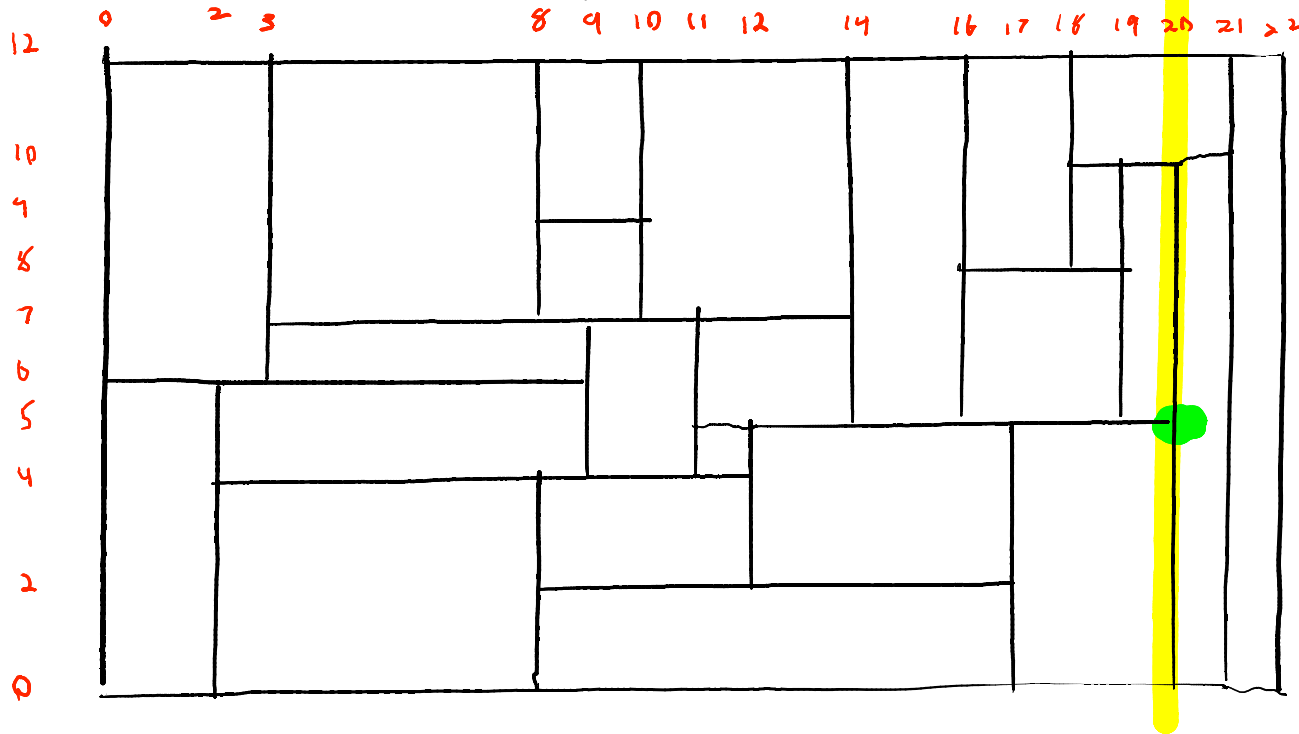
Point Location



0	I	0
	I	6
	I	12
1	I	4
3	I	7
8	I	2
	I	9
9	D	6
10	D	9
11	I	5
12	D	4
14	D	7
16	I	8
17	D	2
18	I	10
19	D	8

sort by x coordinate, sweep left to right with a

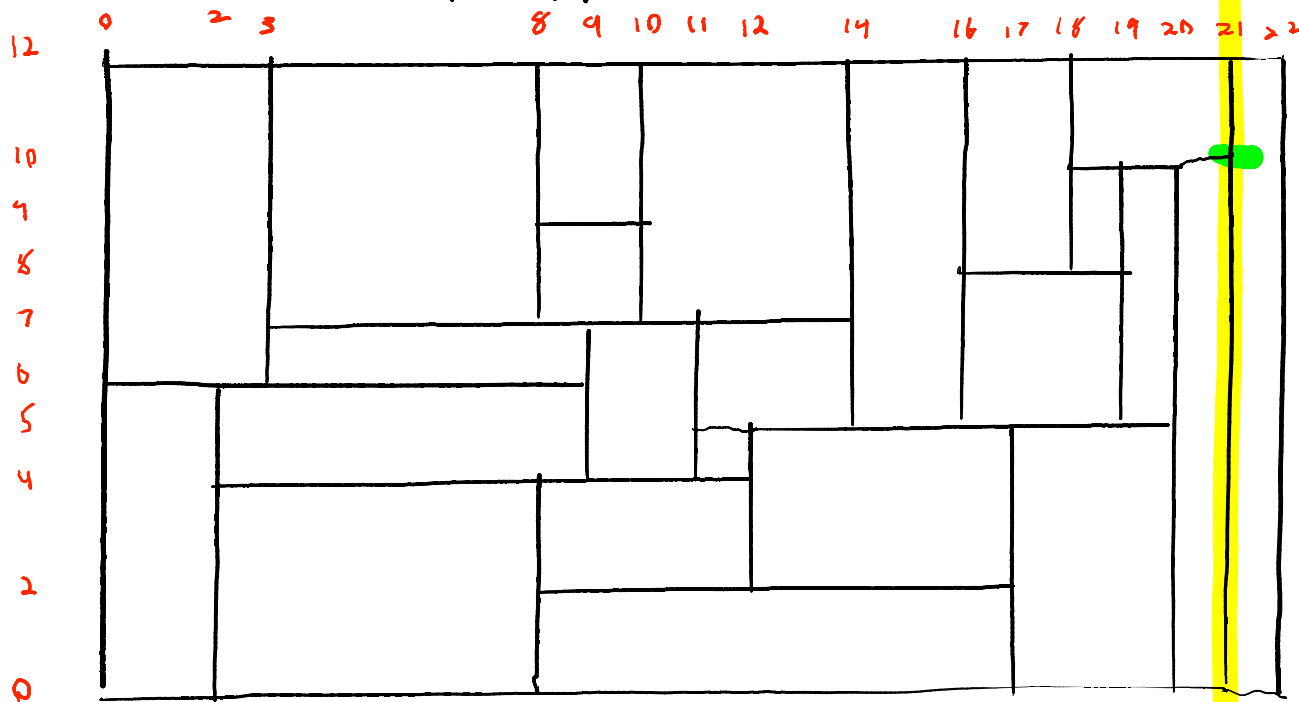
Point Location



0	I	0
	I	6
	I	12
1	I	4
3	I	7
8	I	2
	I	9
9	D	6
10	D	9
11	I	5
12	D	4
14	D	7
16	I	8
17	D	2
18	I	10
19	D	8
20	D	5

sort by x coordinate, sweep left to right with a

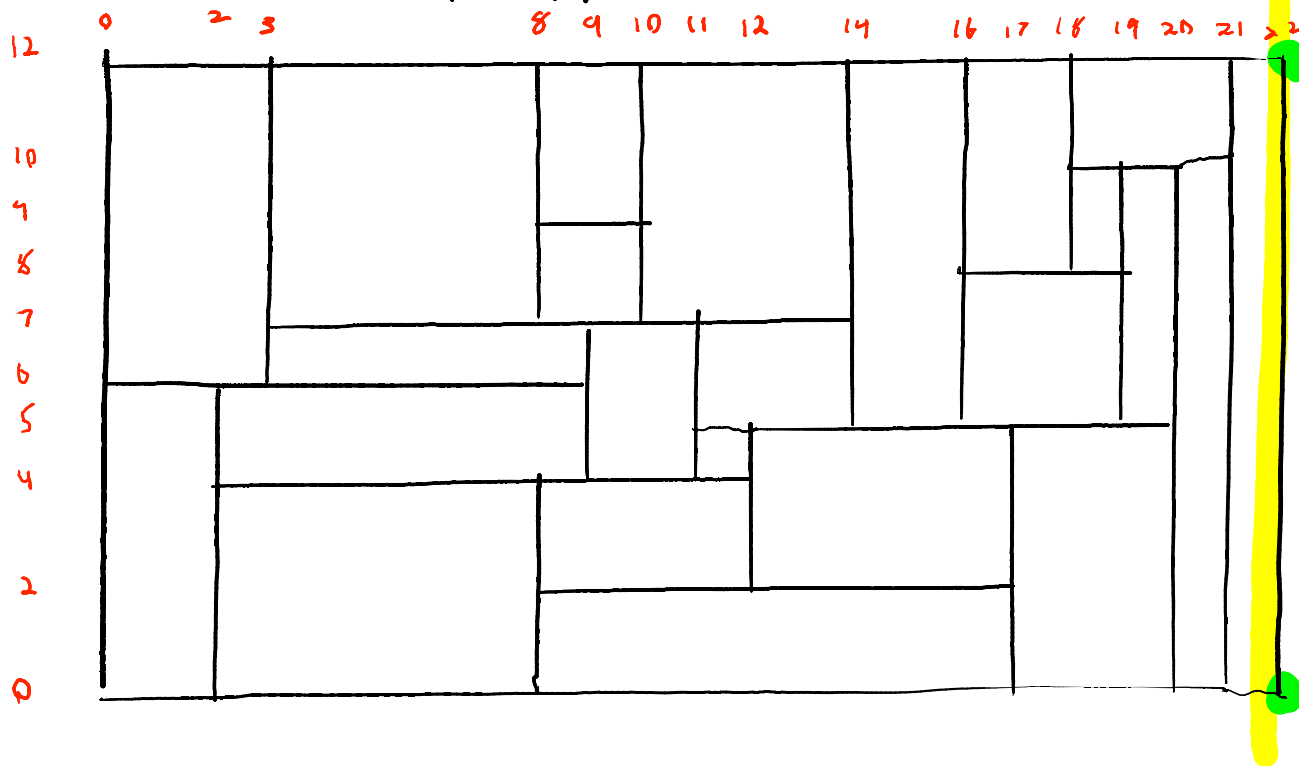
Point Location



0	I	0
	I	6
	I	12
1	I	4
3	I	7
8	I	2
	I	9
9	D	6
10	D	9
11	I	5
12	D	4
14	D	7
16	I	8
17	D	2
18	I	10
19	D	8
20	D	5
21	D	10

sort by x coordinate, sweep left to right with a

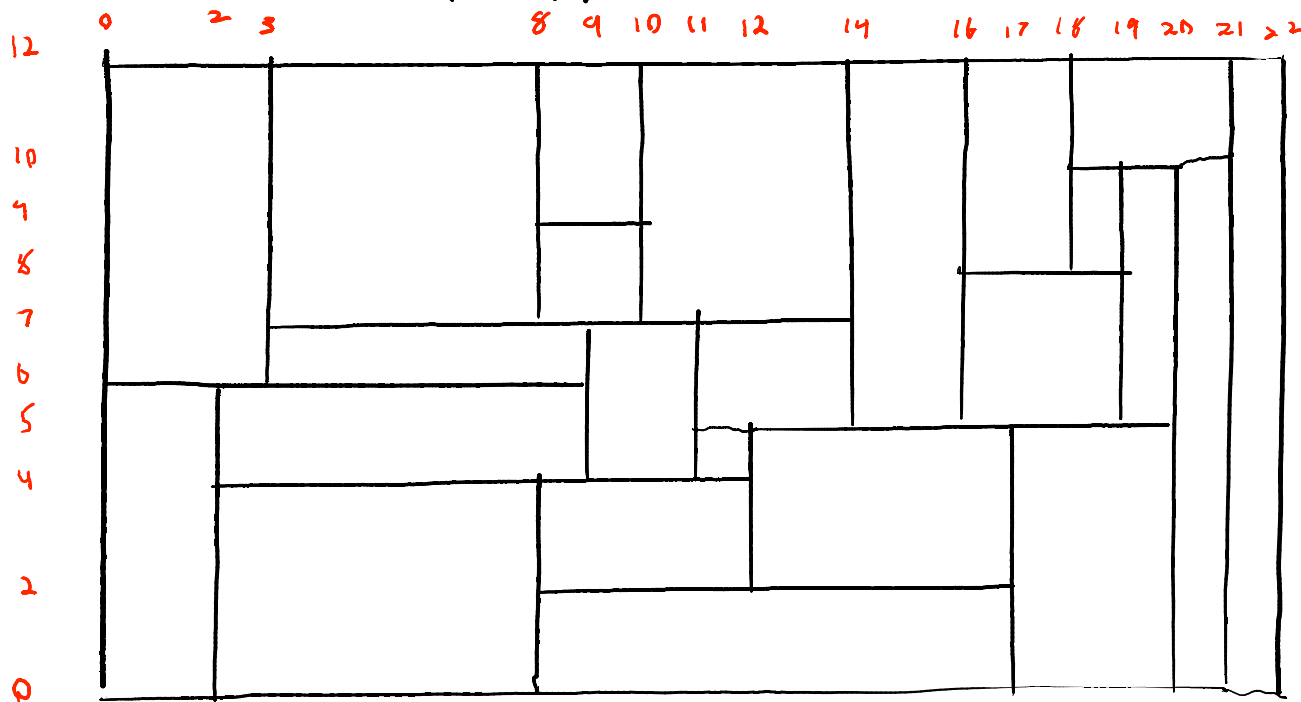
Point Location



0	I	0
	I	6
	I	12
1	I	4
3	I	7
8	I	2
	I	9
9	D	6
10	D	9
11	I	5
12	D	4
14	D	7
16	I	8
17	D	2
18	I	10
19	D	8
20	D	5
21	D	10
22	D	0
	D	12

sort by x coordinate, sweep left to right with a

Point Location

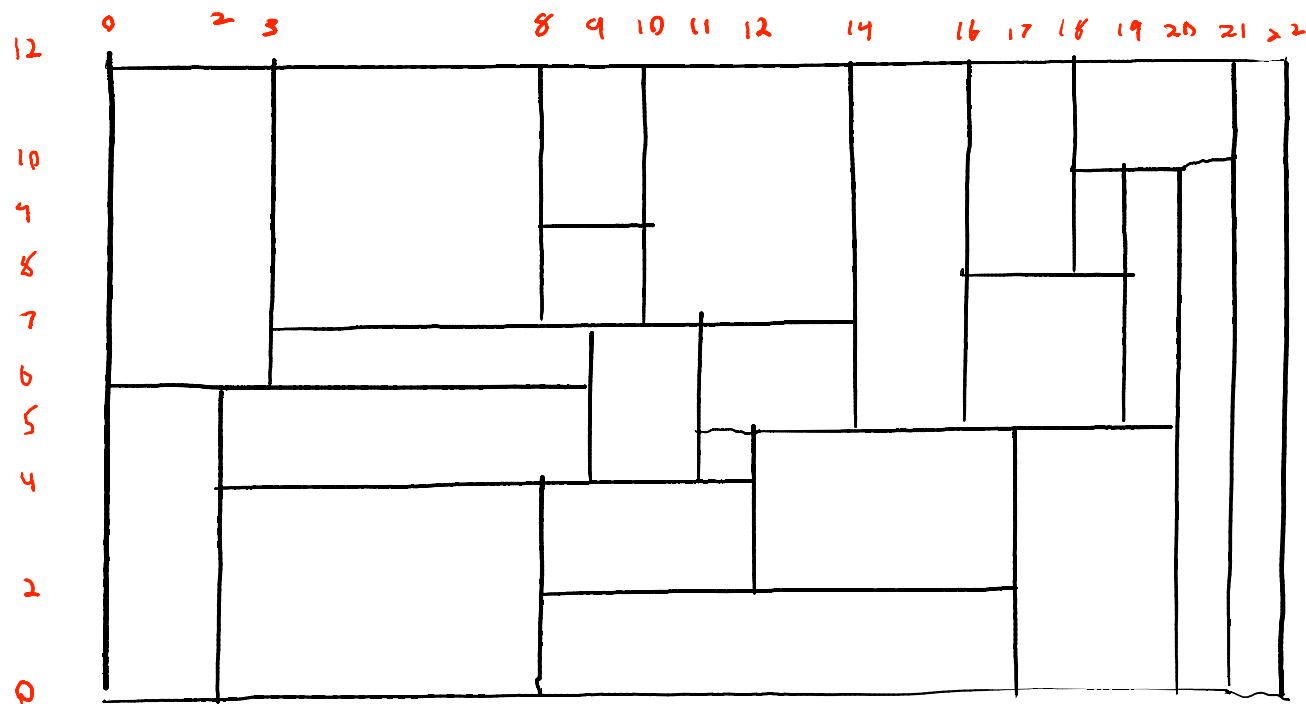


sort by x coordinate, sweep left to right with a

0	I	0	0
	I	6	0, 6
	I	12	0, 6, 12
1	I	4	0, 4, 6, 12
3	I	7	0, 4, 6, 7, 12
8	I	2	0, 2, 4, 6, 7, 12
	I	9	0, 2, 4, 6, 7, 9, 12
9	D	6	0, 2, 4, 7, 9, 12
10	D	9	0, 2, 4, 7, 12
11	I	5	0, 2, 4, 5, 7, 12
12	D	4	0, 2, 5, 7, 12
14	D	7	0, 2, 5, 12
16	I	8	0, 2, 5, 8, 12
17	D	2	0, 5, 8, 12
18	I	10	0, 5, 8, 10, 12
19	D	8	0, 5, 10, 12
20	D	5	0, 10, 12
21	D	10	0, 12
22	D	0	0
	D	12	

Array

0 1 2 3 8 9 10 11 12 14 16 17 18 19 20 21 22



sort by x coordinate, sweep left to right with a

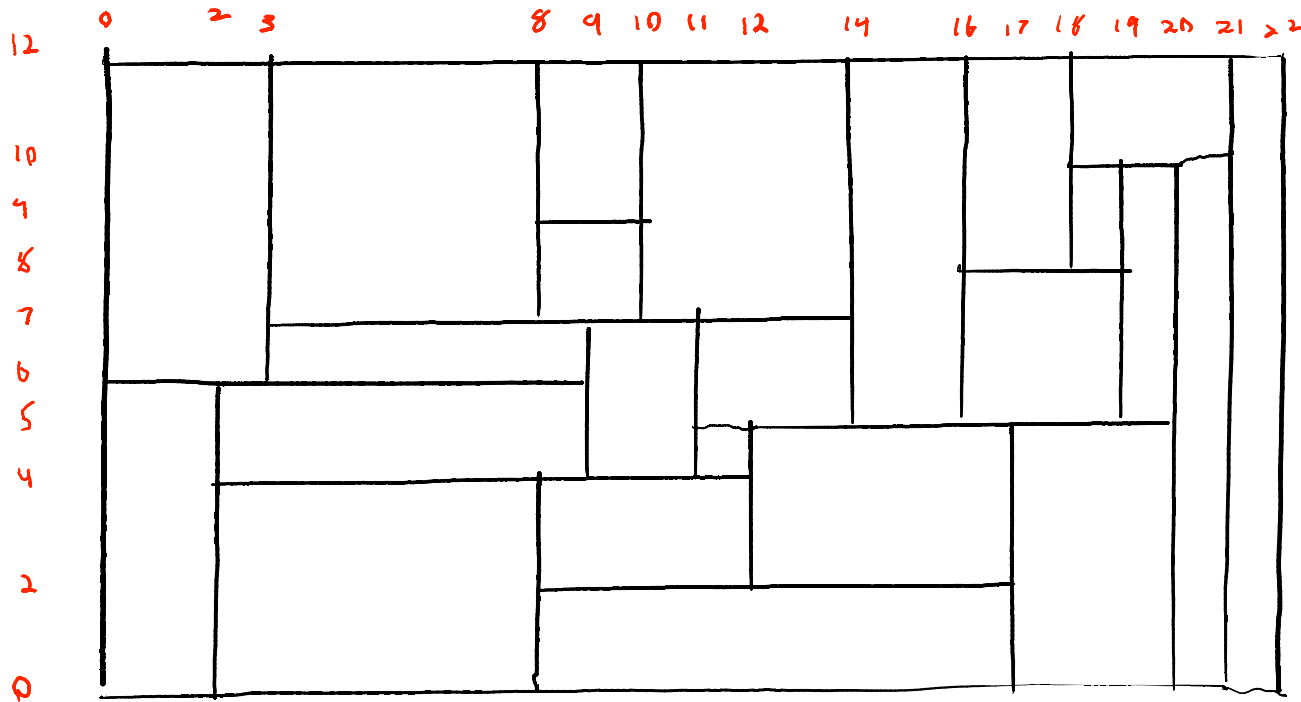
Persistent Dictionary

0	I	0	0
	I	6	0, 6
	I	12	0, 6, 12
1	I	4	0, 4, 6, 12
3	I	7	0, 4, 6, 7, 12
8	I	2	0, 2, 4, 6, 7, 12
	I	9	0, 2, 4, 6, 7, 9, 12
9	D	6	0, 2, 4, 7, 9, 12
10	D	9	0, 2, 4, 7, 12
11	I	5	0, 2, 4, 5, 7, 12
12	D	4	0, 2, 5, 7, 12
14	D	7	0, 2, 5, 12
16	I	8	0, 2, 5, 8, 12
17	D	2	0, 5, 8, 12
18	I	10	0, 5, 8, 10, 12
19	D	8	0, 5, 10, 12
20	D	5	0, 10, 12
21	D	10	0, 12
22	D	0	0
	D	12	

Vertical Ray Shooting

Array

0 1 2 3 8 9 10 11 12 14 16 17 18 19 20 21 22



sort by x coordinate, sweep left to right with a

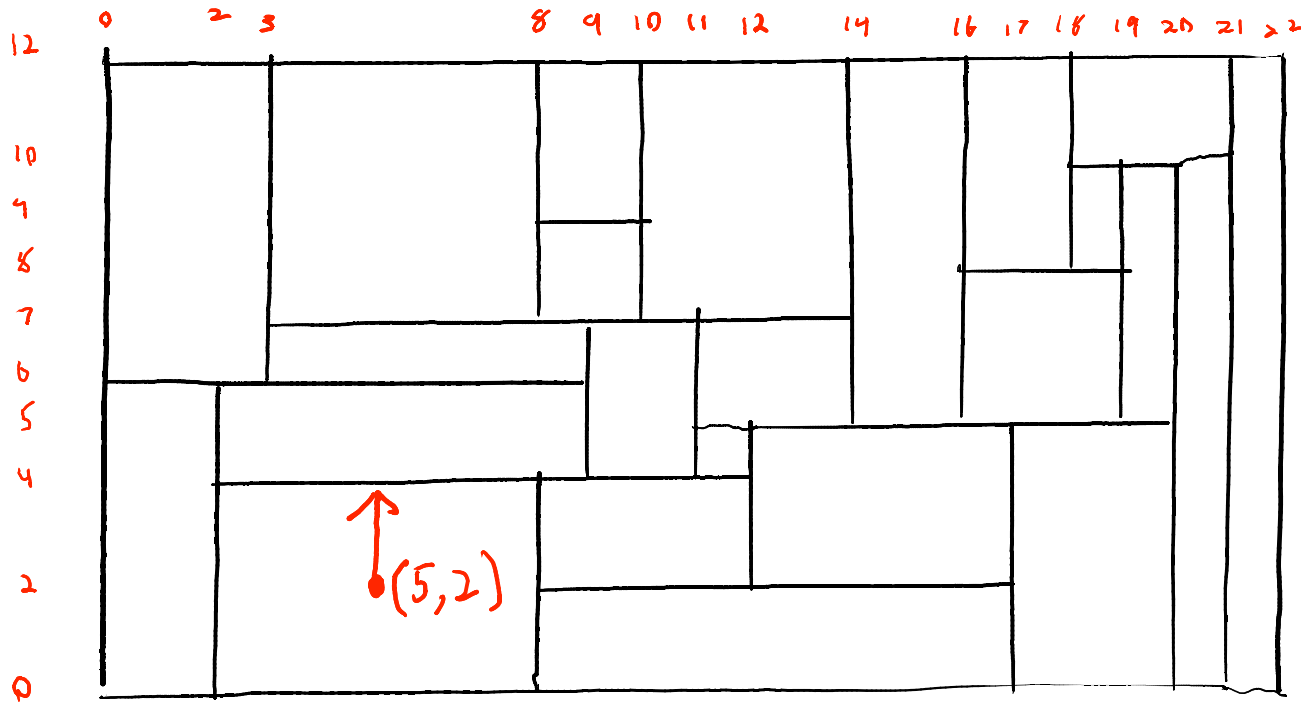
Persistent Dictionary

0	I	0	0
	I	6	0, 6
	I	12	0, 6, 12
1	I	4	0, 4, 6, 12
3	I	7	0, 4, 6, 7, 12
8	I	2	0, 2, 4, 6, 7, 12
	I	9	0, 2, 4, 6, 7, 9, 12
9	D	6	0, 2, 4, 7, 9, 12
10	D	9	0, 2, 4, 7, 12
11	I	5	0, 2, 4, 5, 7, 12
12	D	4	0, 2, 5, 7, 12
14	D	7	0, 2, 5, 12
16	I	8	0, 2, 5, 8, 12
17	D	2	0, 5, 8, 12
18	I	10	0, 5, 8, 10, 12
19	D	8	0, 5, 10, 12
20	D	5	0, 10, 12
21	D	10	0, 12
22	D	0	0
	D	12	

Vertical Ray Shooting

Array

0 1 2 3 8 9 10 11 12 14 16 17 18 19 20 21 22



sort by x coordinate, sweep left to right with a

Persistent Dictionary

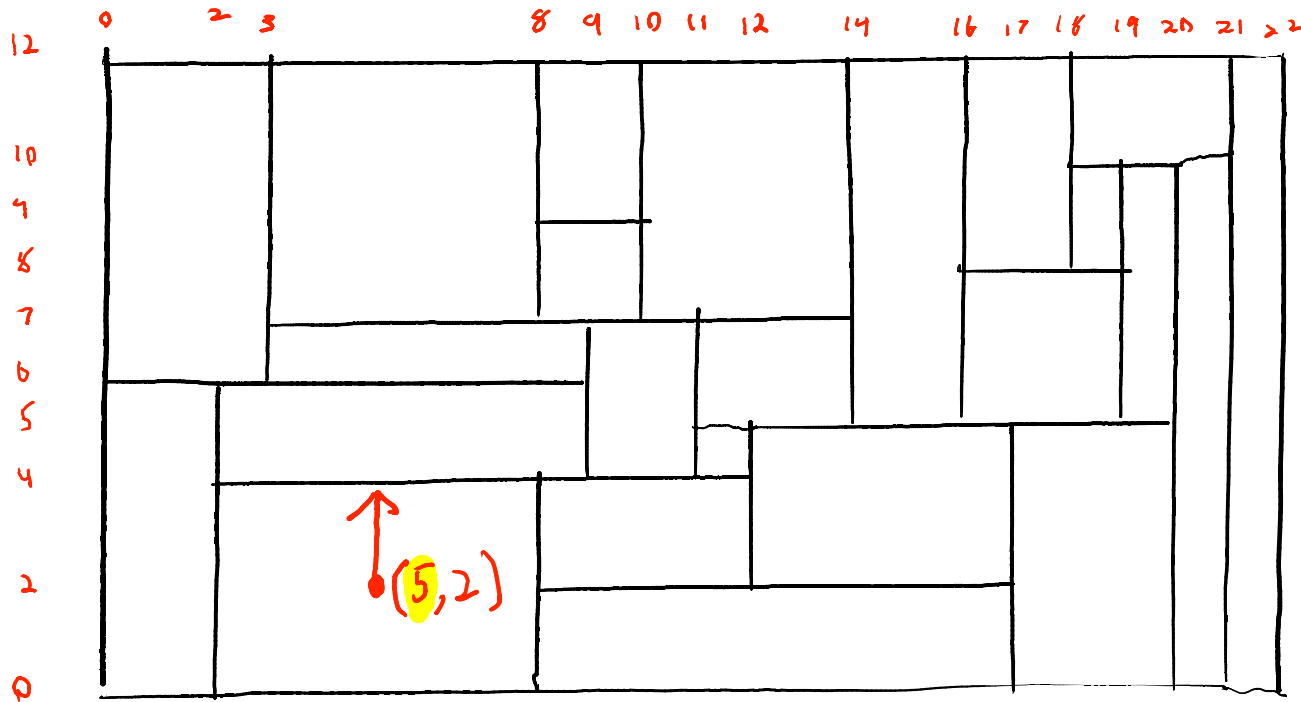
0	I	0	0
	I	6	0, 6
	I	12	0, 6, 12
1	I	4	0, 4, 6, 12
3	I	7	0, 4, 6, 7, 12
8	I	2	0, 2, 4, 6, 7, 12
	I	9	0, 2, 4, 6, 7, 9, 12
9	D	6	0, 2, 4, 7, 9, 12
10	D	9	0, 2, 4, 7, 12
11	I	5	0, 2, 4, 5, 7, 12
12	D	4	0, 2, 5, 7, 12
14	D	7	0, 2, 5, 12
16	I	8	0, 2, 5, 8, 12
17	D	2	0, 5, 8, 12
18	I	10	0, 5, 8, 10, 12
19	D	8	0, 5, 10, 12
20	D	5	0, 10, 12
21	D	10	0, 12
22	D	0	0
	D	12	

Vertical Ray Shooting

Array

0 1 2 3 8 9 10 11 12 14 16 17 18 19 20 21 22

5



sort by x coordinate, sweep left to right with a

Persistent Dictionary

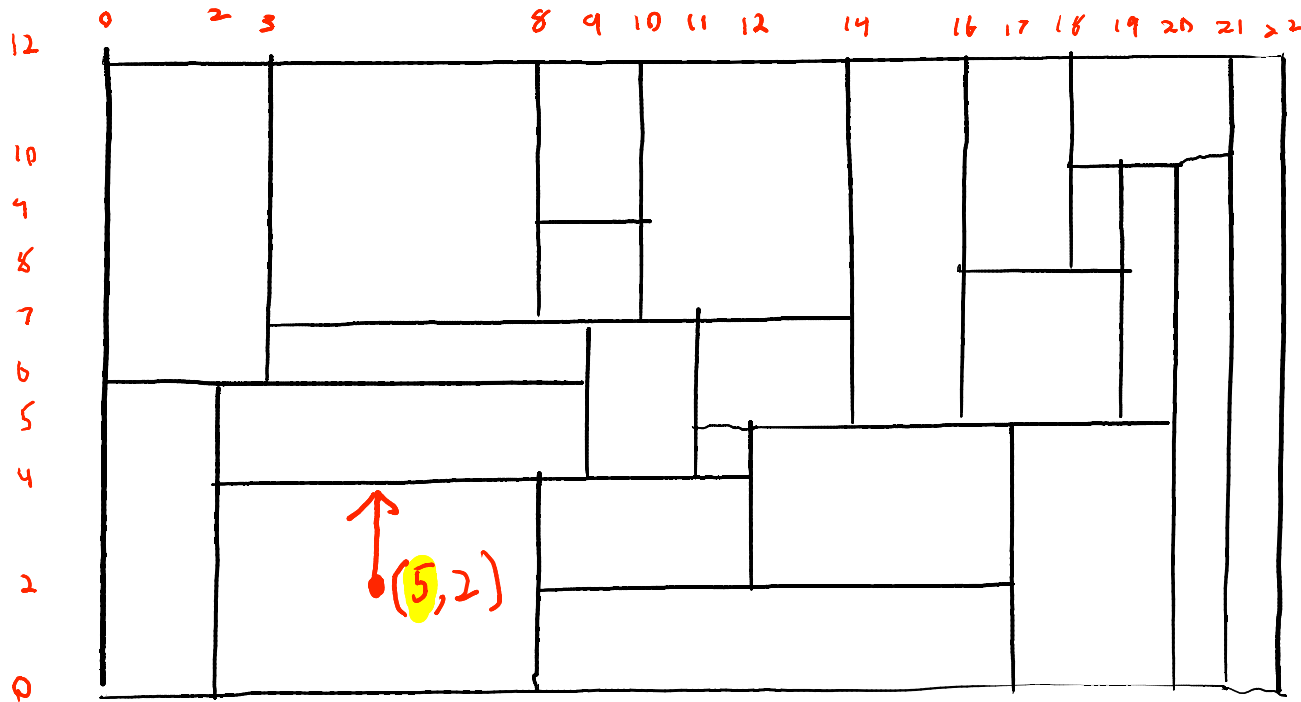
0	I	0	0
	I	6	0, 6
	I	12	0, 6, 12
1	I	4	0, 4, 6, 12
3	I	7	0, 4, 6, 7, 12
8	I	2	0, 2, 4, 6, 7, 12
	I	9	0, 2, 4, 6, 7, 9, 12
9	D	6	0, 2, 4, 7, 9, 12
10	D	9	0, 2, 4, 7, 12
11	I	5	0, 2, 4, 5, 7, 12
12	D	4	0, 2, 5, 7, 12
14	D	7	0, 2, 5, 12
16	I	8	0, 2, 5, 8, 12
17	D	2	0, 5, 8, 12
18	I	10	0, 5, 8, 10, 12
19	D	8	0, 5, 10, 12
20	D	5	0, 10, 12
21	D	10	0, 12
22	D	0	0
	D	12	

Vertical Ray Shooting

Array

0 1 2 3 8 9 10 11 12 14 16 17 18 19 20 21 22

5



sort by x coordinate, sweep left to right with a

Persistent Dictionary

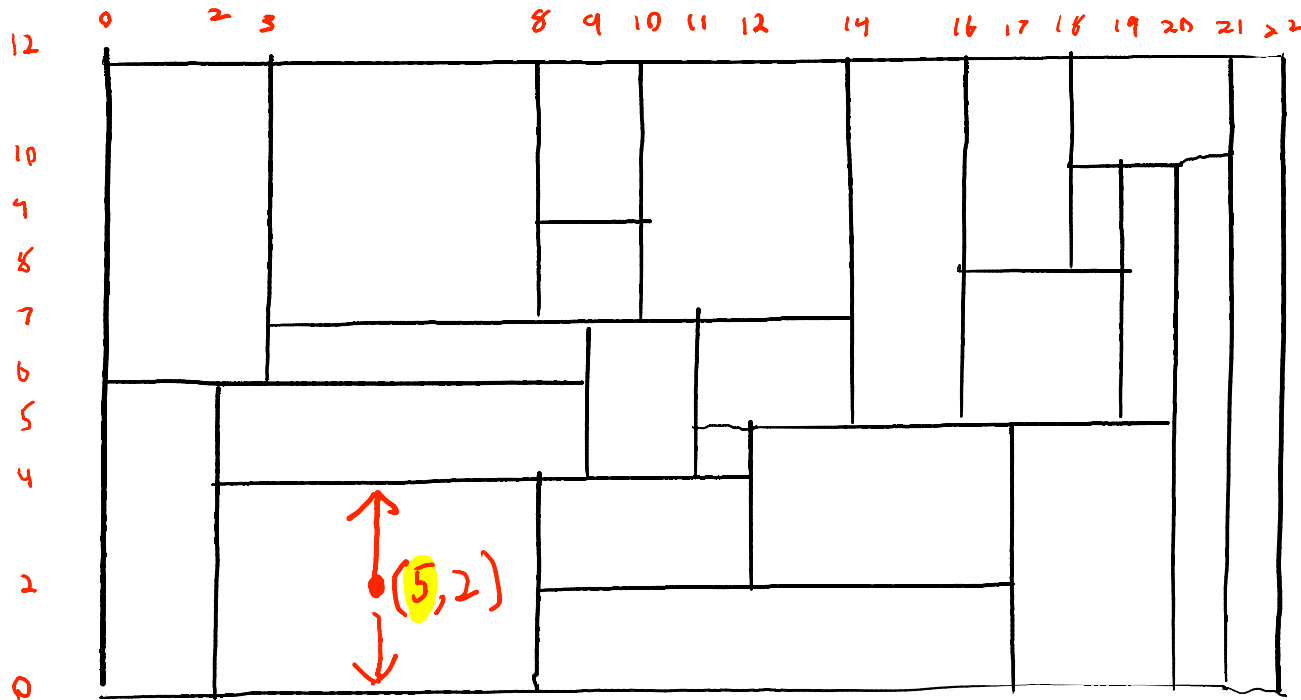
0	I	0	0
	I	6	0, 6
	I	12	0, 6, 12
1	I	4	0, 4, 6, 12
3	I	7	0, 4, 6, 7, 12
8	I	2	0, 2, 4, 6, 7, 12
	I	9	0, 2, 4, 6, 7, 9, 12
9	D	6	0, 2, 4, 7, 9, 12
10	D	9	0, 2, 4, 7, 12
11	I	5	0, 2, 4, 5, 7, 12
12	D	4	0, 2, 5, 7, 12
14	D	7	0, 2, 5, 12
16	I	8	0, 2, 5, 8, 12
17	D	2	0, 5, 8, 12
18	I	10	0, 5, 8, 10, 12
19	D	8	0, 5, 10, 12
20	D	5	0, 10, 12
21	D	10	0, 12
22	D	0	0
	D	12	

Vertical Ray Shooting

Array

0 1 2 3 8 9 10 11 12 14 16 17 18 19 20 21 22

5



sort by x coordinate, sweep left to right with a

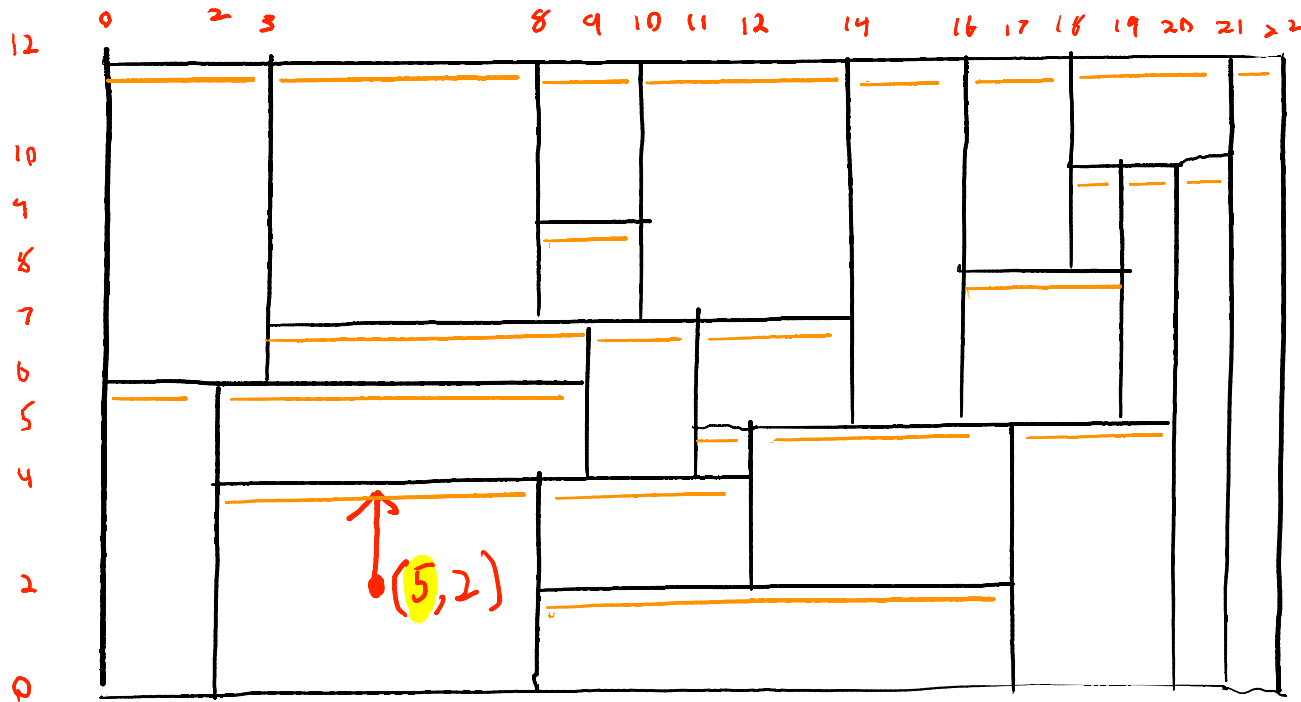
Persistent Dictionary

0	I	0	0
	I	6	0, 6
	I	12	0, 6, 12
1	I	4	0, 4, 6, 12
3	I	7	0, 4, 6, 7, 12
8	I	2	0, 2, 4, 6, 7, 12
	I	9	0, 2, 4, 6, 7, 9, 12
9	D	6	0, 2, 4, 7, 9, 12
10	D	9	0, 2, 4, 7, 12
11	I	5	0, 2, 4, 5, 7, 12
12	D	4	0, 2, 5, 7, 12
14	D	7	0, 2, 5, 12
16	I	8	0, 2, 5, 8, 12
17	D	2	0, 5, 8, 12
18	I	10	0, 5, 8, 10, 12
19	D	8	0, 5, 10, 12
20	D	5	0, 10, 12
21	D	10	0, 12
22	D	0	0
	D	12	

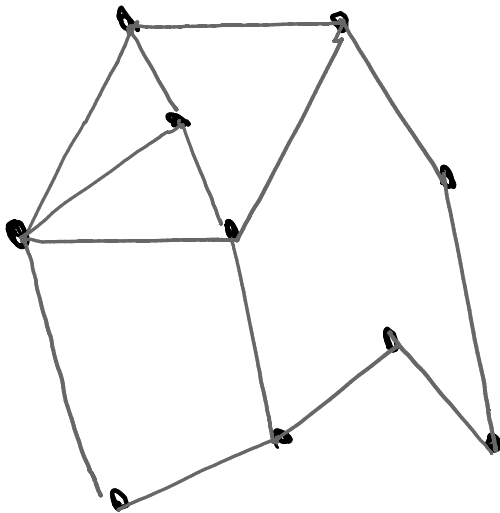
Vertical Ray Shooting

→ Point location

Use the "top" of each box as input

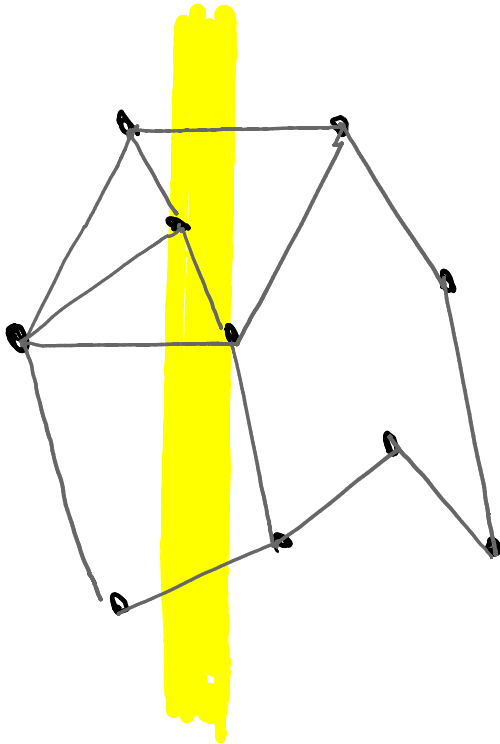


Non-orthogonal



Non-orthogonal

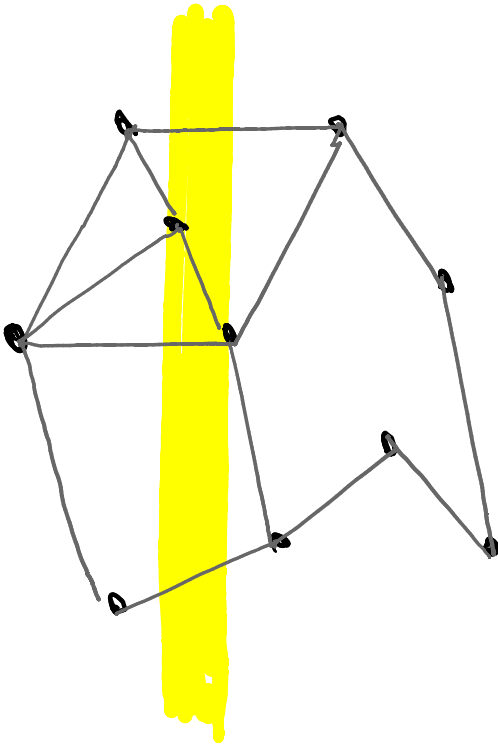
Do the exact same thing



Non-orthogonal

Do the exact same thing

- Instead of putting y-coordinates in a B-B tree, put line segments
- Not a problem so long as they don't cross.



Summary

- $O(\log n)$ search time
- $O(n)$ space
- $O(n \log n)$ construction (sorting)
- works well on non-triangular subdivisions
- Constants not crazy (about $5 \log n$ comparisons
 $3 \log n$ links to follow)

Some More about Persistence

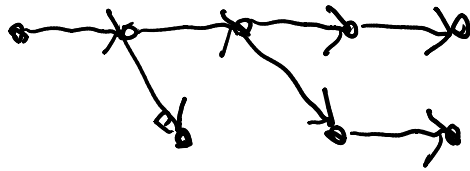
- Works on any pointer-based structure with $O(1)$ indegree.

- Partial persistence



Query all versions
Update current version
Version Path

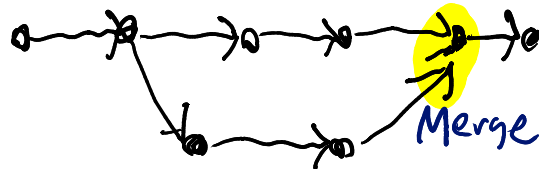
- Full Persistence



Query all versions
Fork off a new version
by updating any version

Version Tree

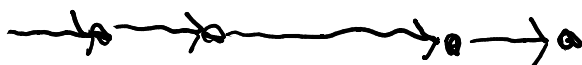
- Confluent Persistence



Allows merge-type operations which combine previous versions.

Version DAG
SLOW

- Retroactive

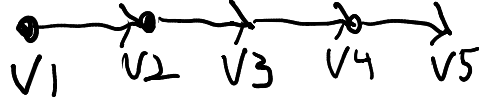


Allows changes to the execution sequence **No General Results**

Some Move about Persistence

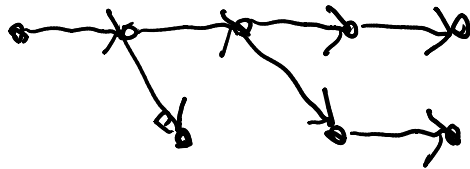
- Works on any pointer-based structure with $O(1)$ indegree.

- Partial Persistence



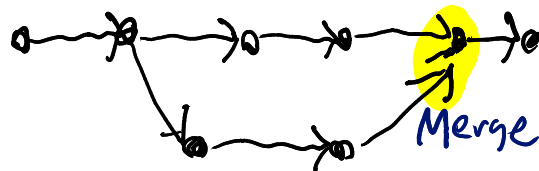
Query all versions
Update current version
Version Path

- Full Persistence



Query all versions
Fork off a new version
by updating any version
Version Tree

- Confluent Persistence



Allows merge-type operations
which combine previous versions.

Version DAG
SLOW

- Retroactive

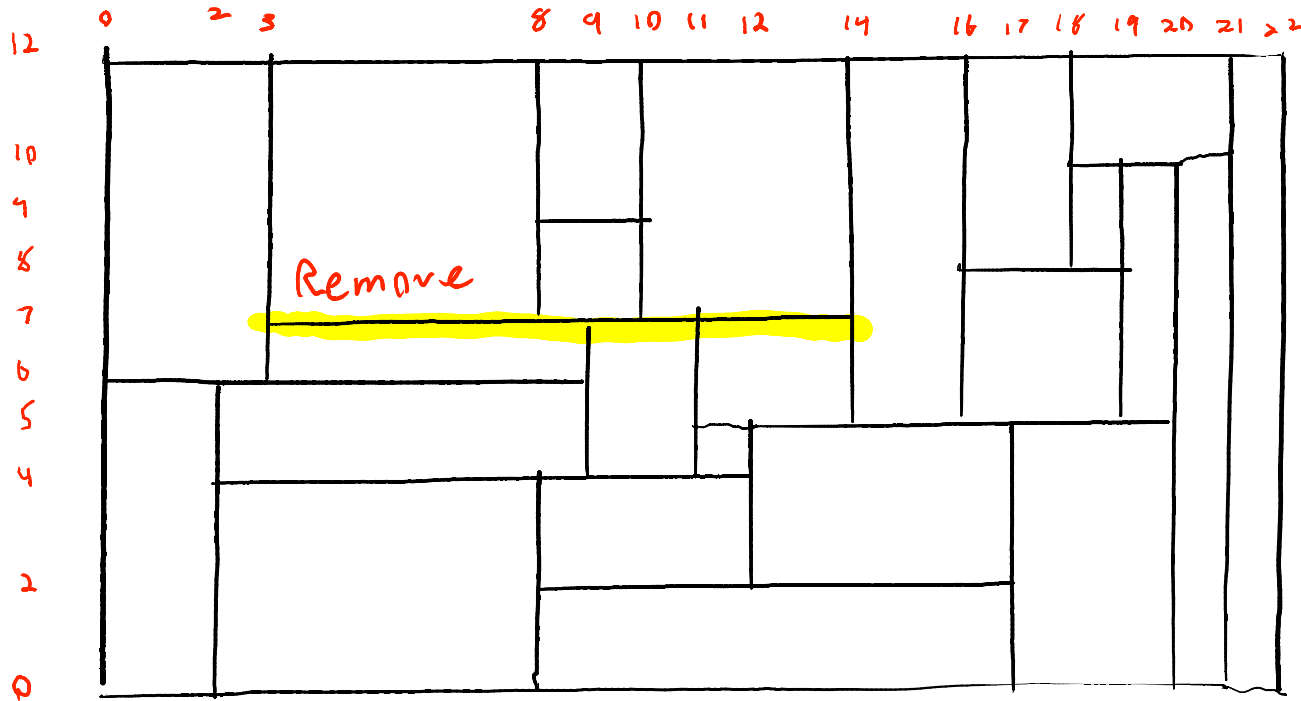


Allows changes to the execution
sequence No General Results

Retroactive Dictionary = Dynamic Point Location

Array

0 1 2 3 8 9 10 11 12 14 16 17 18 19 20 21 22



Persistent Dictionary

0	I	0	0
	I	6	0, 6
	I	12	0, 6, 12
1	I	4	0, 4, 6, 12
3	I	7	0, 4, 6, 7 , 12
8	I	2	0, 2, 4, 6, 7 , 12
	I	9	0, 2, 4, 6, 7 , 9, 12
9	D	6	0, 2, 4, 7 , 9, 12
10	D	9	0, 2, 4, 7 , 12
11	I	5	0, 2, 4, 5, 7 , 12
12	D	4	0, 2, 5, 7 , 12
14	D	7	0, 2, 5, 12
16	I	8	0, 2, 5, 8, 12
17	D	2	0, 5, 8, 12
18	I	10	0, 5, 8, 10, 12
19	D	8	0, 5, 10, 12
20	D	5	0, 10, 12
21	D	10	0, 12
22	D	0	0
	D	12	

We don't know how to do this efficiently

The
End