06-01

Monday, June 1, 2020 6:30 PM

Count inversions

$A=[1,3,2,4]$ # of inversions is 1

$(A[i], A[j])$ is an inversion
- $i < j$
- $A[i] > A[j]$

3 2

$A=[4,3,2,1]$ # of inversions is 6

(4,3) (4,2) (4,1)
(3,2) (3,1) (2,1)

---

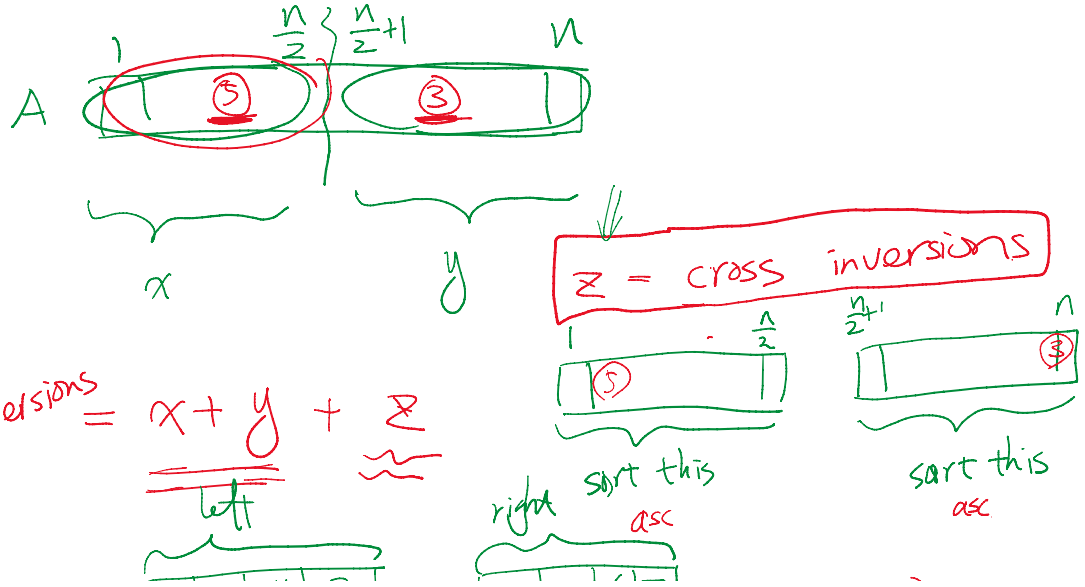
base line iterative algo

```
for (i = 1 ... n) {
    for (j = i+1, ..., n) {
        if (A[i] > A[j])
            inversion_cnt += 1;
    }
}
return inversion_cnt;
```

$O(n^2)$

---

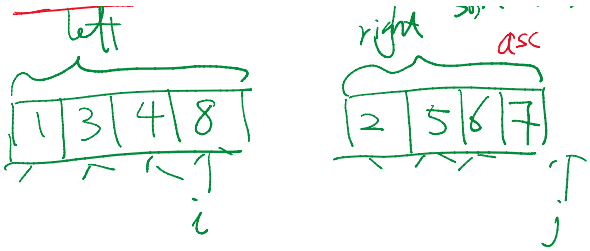
only want the count



# of inversions = x + y + z

left

right



O(n)

(1,2) 1 is useless now / move on to next

(3,2) inversion found!! 3 inversions!!

(3,2) (4,2) (8,2)

2 is useless now

(3,5) 3 is useless

(4,5) 4 is useless

(8,5) 1 inversion

(8,6) 1 inversion

(8,7) 1 inversion

```
count_inversion (A[1...n]) {          # inversion count
    if (n ≤ 1) return 0;
    x = count_inversion (A[1... n/2]);
    y = count_inversion (A[n/2 ... n]);
    sort (A[1... n/2])
    sort (A[n/2+1 ... n])
    z = cross_inversion (A[1... n/2], A[n/2+1 ... n]);   → O(n)
    return x+y+z
}
```

$T(n) = 2T(\frac{n}{2}) + n(\log n)$ $\quad T(n) = O(n \log^2 n)$

```
count_inversion (A[1...n]) {          # inversion
                                      sort
    if (n≤1) return 0;
    x = count_inversion (A[1... n/2])
    y = count_inversion (A[n/2+1 ... n])
    z = cross_inversion (A[1... n/2], A[n/2+1 ... n])
    merge (A[1 ... n/2] A[n/2+1 ... n])
```

z = cross-inv... (A[1...n/2], A[n/2+1...n])

A[ ] = merge (A[1...n/2], A[n/2+1...n])

return x+y+z;

↳ $T(n) = 2T(\frac{n}{2}) + n$ $T(n) = \Theta(n \lg n)$

(A) O(n lg n)

pros - easier
- allow extra

cons - slow

O(n lg n) (B)

pros - slightly better r.t.

cons - more space

---

10 movie

⇒ A [1 2 ... 10]

B [2 1 10 ... 9]

---

count_inversion (const vector<int>& A);

count_inversion (vector<int>& A);

BST

set

map

dynamic collection of item

dynamic collection of <key, value>

TreeMap

HashMap

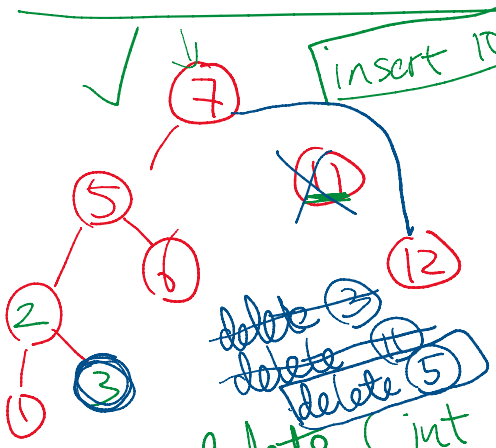
hashing

insert (k, v)
get (k)
delete (k)

---

	insert	get	delete		
BST	O(lg n)	O(lg n)	O(lg n)	O(h)	ordered traversal
hash	$\Theta(1)^*$	$O(1)^*$	$\Theta(1)^*$	avg.	

hash O(1) O(1) O(1) avg.



- left subtree BST
- right subtree BST
- root > all values in left subtree
- root < all values in right subtree

delete (int key)

delete (TreeNode node) sorted given BST

Binary Tree Traversal

- pre-order
- in-order
- post-order
- level-order

```
preorder (TreeNode r){
    process(r);
    preorder(r.left);
    preorder(r.right);
}
```

```
class TreeNode {
    int key;
    TreeNode left;
    TreeNode right;
}
```

O(h)

```
insert (TreeNode r, int key){
    if (r == null)
        return new TreeNode(key)
    if (r.key < key)
        r.right = insert(r.right, key)
    else
        r.left = insert(r.left, key)
    return r;
}
```

delete

delete

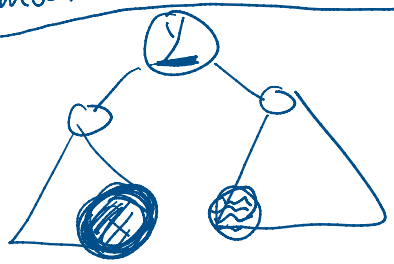
- case 1: no children.
  delete && update parent's pointer

- case 2: single child
  delete && parent points directly to the child

- case 3: two children.
  - 3.1 find predecessor p
  - 3.2 swap p with x
  - 3.3 delete recursively

predecessor → right most node in x's left subtree

successor → left most node in right subtree



```
delete (TreeNode r, int key){
    if ( r.key == key ) {
        if (r.left == null && r.right == null){
            return null;
        }
        if (r.left == null)
            return r.right;
        if (r.right == null)
            return r.left;

        p = Rightmost (r.left)
```

```
        p = rightmost(r.left);
        swap_key(r, p);
        r.left = delete(r.left);
    }
    if (key < r.key)
        r.left = delete(r.left, key);
    else
        r.right = delete(r.right, key);
    return r;
}
```

---

quiz start 8:30