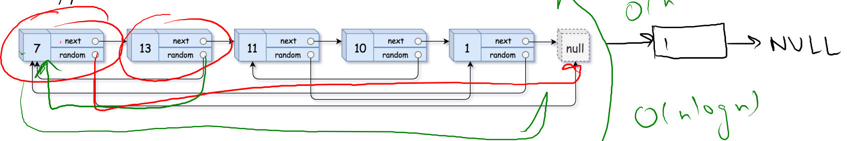
n = 1000 $O(n^2) = 10^6$ 1 sec → ~10^8–10^9

Copy List with Random ptr

Tree

$O(n^2)$



$O(n \log n)$

arr [0],[1],[2]

key

Value

map

→ $O(\log n)$

{ map[TreeNode* → TreeNode*] }

unordered_map → $O(1)$

a → 1 → → →

0	10
1	20
2	23
3	5
4	11

'a' →
'b' →
'c' →
'd' →

a → 1

b → 1

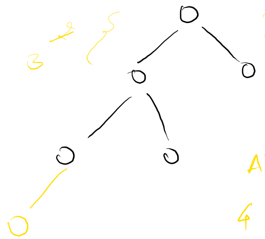
array

1 → a → b →

$O(n)$

$O(1)$

Balancing a BST



3 - 1 = 2 > 1 → Not balanced

2 - 1 = 1

AVL Tree

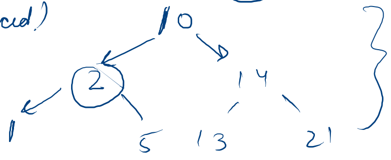
4

BST → inorder Traversal → sorted

1 2 3 4 5 6 7

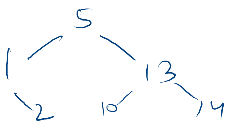
0	1	2	3	4	5	6
1	2	5	10	13	14	21

sorted → BST (balanced)



0					5
1	2	5	10	13	14

$mid = \frac{s+e}{2} = \frac{0+5}{2} = 2$



Memory Leak.

1 → 2 → 3 → 4 → 5 → 6

1 → 2 → 4 → 5 → 6

3 → Node
del

Recursion < Iterative

$O(n) = kn$

$\leq kn$

k → kn

m ≤ k

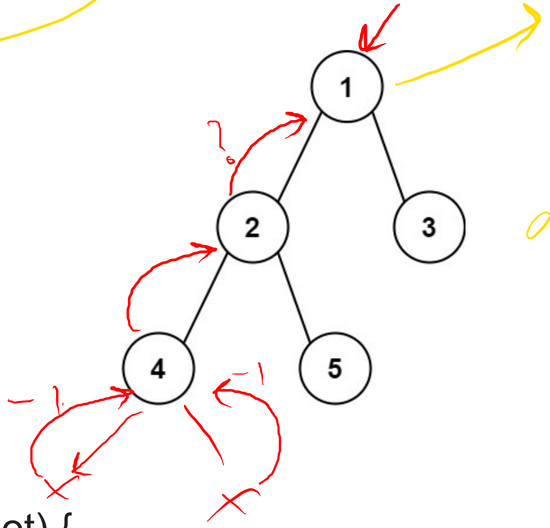
Diameter of BT

ans = 0

```
class Solution {
public:
    int ans=0;
    int diameter(TreeNode* root){
        if(root==nullptr) return -1;
        int l=diameter(root->left);
        int r=diameter(root->right);
        ans=max(ans,l+r+2);
        return max(l,r)+1;
    }
    int diameterOfBinaryTree(TreeNode* root) {
        diameter(root);
        return ans;
    }
};
```

diameter

height



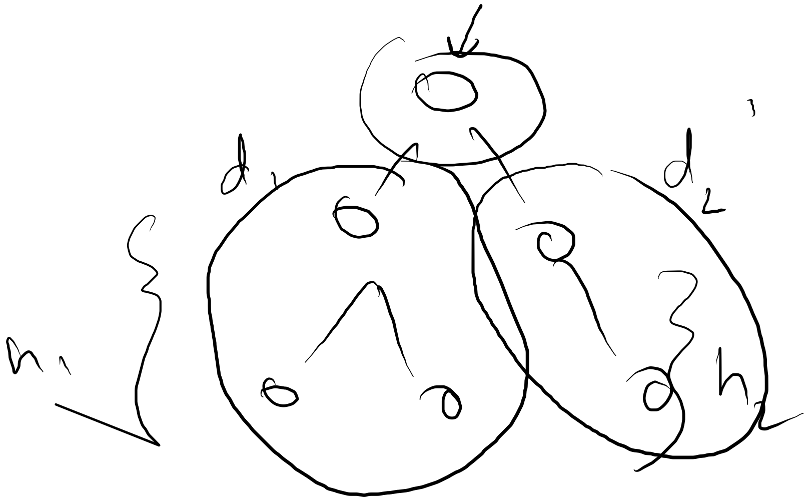
~~d_1 = left~~

h_1 → left

h_2 → right

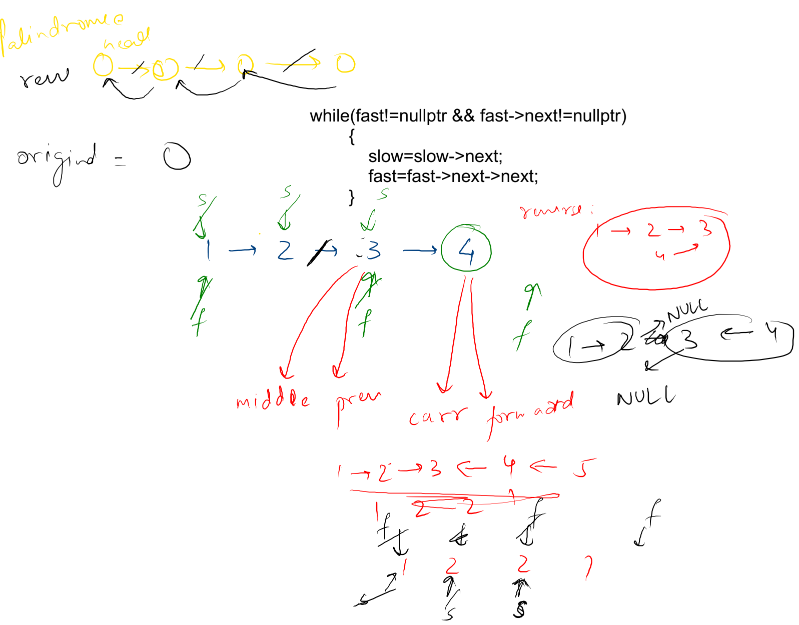
diam = max(diam, $h_1 + h_2 + 2$)

diam → length of edge



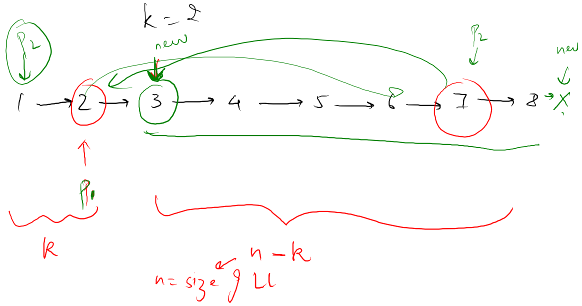
$d_3 = h_1 + h_2 + 2$

Palindrome



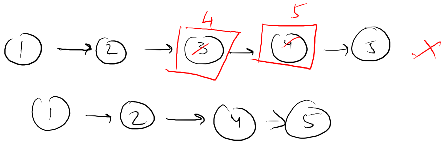
```
while(fast!=nullptr && fast->next!=nullptr)
    {
        slow=slow->next;
        fast=fast->next->next;
    }
```

Swapping Nodes



Deleting a node from LL (w/o head)

node → val = node → next → val



```
c = 0
for (int i = 1 ; i < n ; i++ )
  if arr[i] > arr[i-1]
     c++
```