Recursion & Backtracking

- Karun Karthik

Contents →

D1

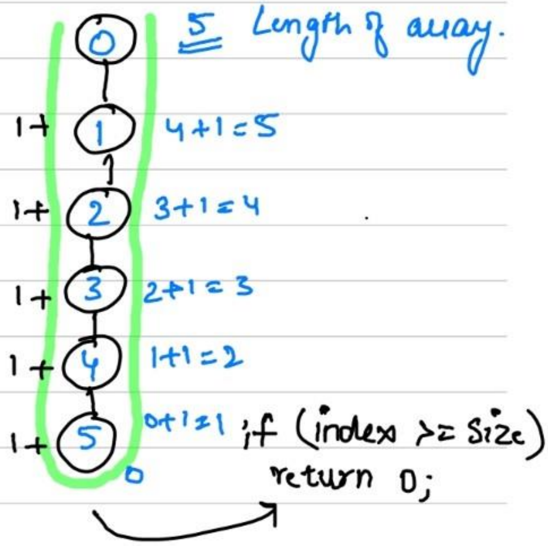
Recursion

⓪ Length of an array

[20, 10, 40, 50, 30]
0 1 2 3 4

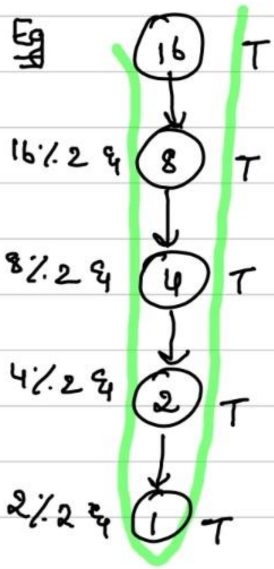
TC = O(n)

SC = O(n).



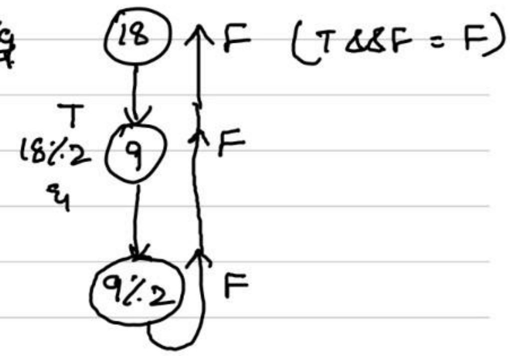
① Power of 2 → $2^x = 2^0 \cdot 2^1 \cdot 2^2 \cdots 2^n$.

Eg



if 1 then
return true.

Eg



TC = $O(\log_2 n)$

Linked List

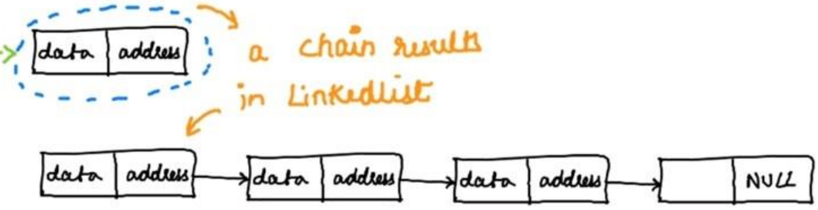
- Karun Karthik

Contents →

Linked List

Linkedlist is linear data structure, which consists of a group of nodes in a sequence.

Class Node
- data
- Node*



Advantages

1. Dynamic nature
2. Optimal insertion & deletion
3. Stacks and queues can be easily implemented
4. No memory wastage

Disadvantages

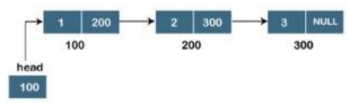
1. More memory usage due to address pointer.
2. Slow traversal compared to arrays.
3. No reverse traversal in singly linked list
4. No random access.

Real life Applications

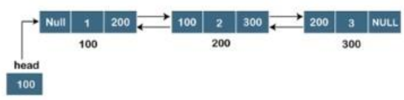
1. Previous & next page in browser
2. Image Viewer
3. Music player

Types

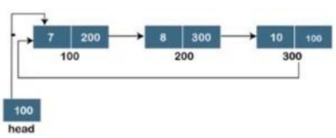
1. Singly linkedlist



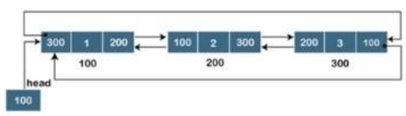
2. Doubly linkedlist



3. Circular Linkedlist



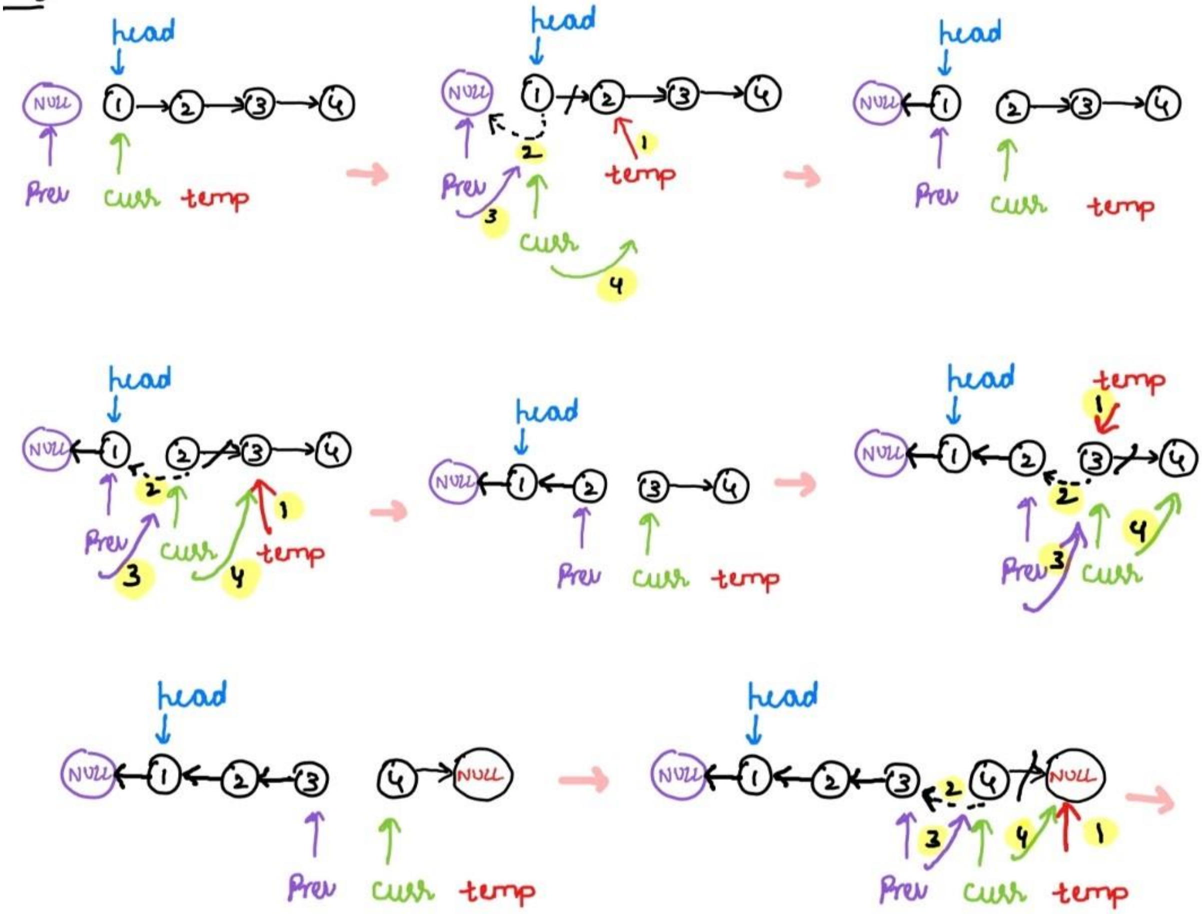
4. Doubly circular linkedlist

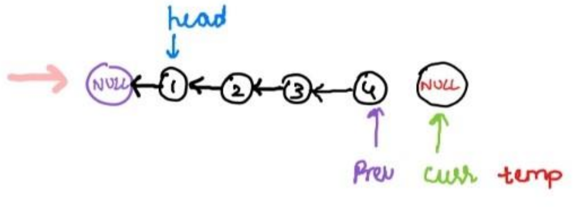


① Reverse a linkedlist → given a linkedlist, return reversed list.

Eg 1 → 2 → 3 → 4 ⇒ 4 → 3 → 2 → 1

Sol)





→ As curr points to NULL, we reached end of linkedlist

→ Return Prev as it is the starting pointer of reversed list.

https://www.linkedin.com/in/karun-karthik

Trees - Part 1

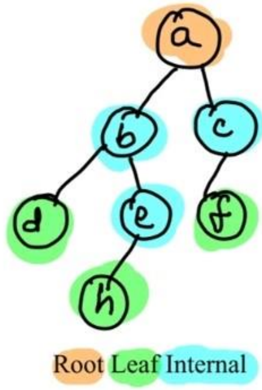
- Karun Karthik

Contents

Trees

Why trees?

1. Hirarchy
2. Computer system. (UNIX)



Tree - collection of tree-nodes

① Class Treenode
- → data
- → list <Treenode> children

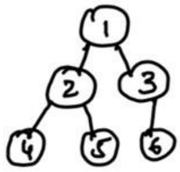
② Binary Tree → atmost 2 children (0,1,2)
- data
- → leftchild
- → rightchild

③ Types →

Ⓐ Complete Binary Tree

↳ all levels are completely filled except last one

Eg



Ⓑ Perfect Binary Tree

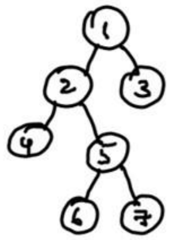
↳ every internal node has exactly 2 children

Eg

Ⓒ Full Binary tree

↳ if every node has 0 or 2 children

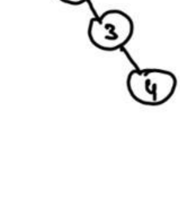
Eg



Ⓓ Skewed Binary Tree (* used for finding complexity)

↳ all nodes have either one or no child.

Eg



Trees - Part 2

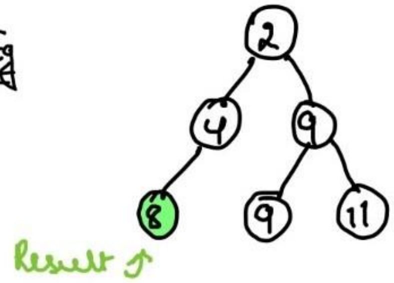
- Karun Karthik

Contents

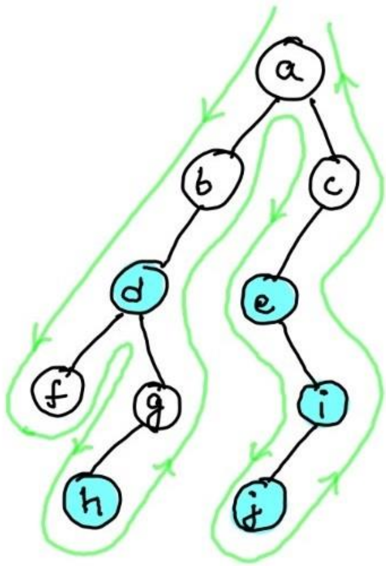
https://www.linkedin.com/in/karun-karthik-5a0794187/

D7 (21) Print all nodes that donot have any Siblings

Eg



Sibling → Same level, Same parent



TC → O(n)

SC → O(n)

At every node, check if

both branches exist → then call both of them recursively

only left branch exist → then call left branch recursively

only right branch exist → then call right branch recursively

https://www.linkedin.com/in/karun-karthik-5a0794187/

DSA HUB
dsa-hub.netlify.app

DSA HUB

2 / 23

Recursion & Backtracking

– Karun Karthik

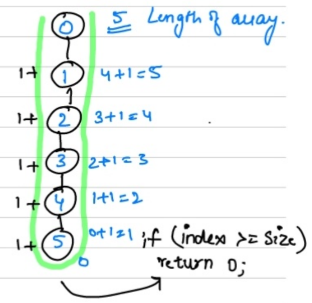
Contents →

D1 Recursion

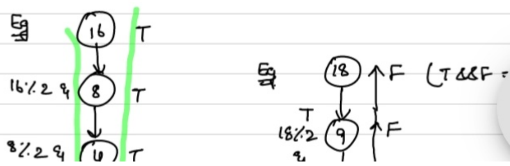
(0) Length of an array

[20, 10, 40, 50, 30]
0 1 2 3 4

TC = O(n)
SC = O(n).



(1) Power of 2 → $2^x = 2^0 . 2^1 . 2^2 .. 2^n$.



DSA HUB
dsa-hub.netlify.app

Contents

Application link in the comments section