Interview में which DS use
then not given

Imp Points :- 9th Nov 2025 (Sunday) Sunday III Yr IIIP

1. ArrayList v/s LinkedList

insertion: TC = O(n) , O(1)

Search: TC = O(1) , O(n)

{ यहाँ से useCase वाला DS. जहाँ ज्यादा insertion का काम हो तो use prefer ~~Linked~~ LL and Vice-versa

prop of LL

1. [data | next] [1] → [2] → [3] → NULL

variable Size, Dynamic Memory ~~Loc~~ Allocation, Non-Contiguous Size

2. हम LL में Backward ~~~~ तरफ से Node को traverse नहीं कर सकते

So that we make a copy temp

LL is Linear D.S.

Operation

- push-front → pop-front
- push-back → pop-back
- void insert (int val, int position)
- int Search (int Key)

3.

3.

9 November 2025 at 16:00

Que:- (1773) Count Items matching a Rule

arr<int> items where items[i] = [$type_i$, $color_i$, $name_i$] describe i-th items. You also given a rule represent by two str 'ruleKey', 'ruleValue'.

The i-th item said to match rule if:

R1: ruleKey == "type" and ruleValue == $type_i$

R2: ruleKey == "color" and ruleValue == $color_i$

R3: ruleKey == "name" and ruleValue == $name_i$

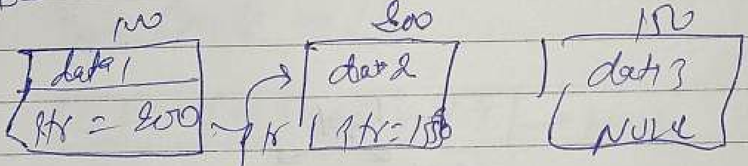
return no. of item match rule.

TestCase: items = [["phone", "blue", "pixel"], [
["computer", "silver", "lenovo"]
["phone", "gold", "iphone"]

ruleKey = "colour",
ruleValue = "silver"

Linkedlist: Linear   Arr [1 | 2 | 3 | 4] 100 104 108

dynamic DS [1] → [2] → [3] node

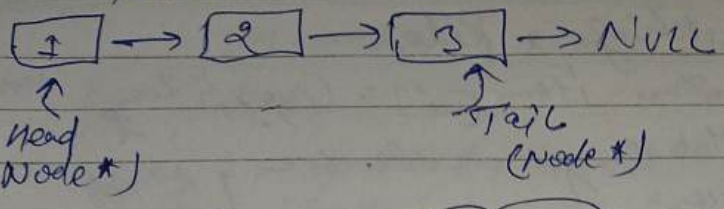


node: data, next Node*

इस LL में arr के तरह index से किसी O(1) से node को access नहीं कर सकते, इसमें O(n) से होगा T.C worst case में

```
class Node myLL {
  public
     int data;
     Node* next;
}
  Node(int val){
     data = val;
     next = NULL;
  }
```

LL implement



class Node {
public:
int data;
Node * next;

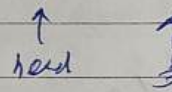
Node(int val) {
data = val;
next = NULL;
}
};

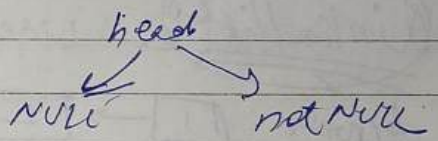
class List {
private:
Node * head;
Node * tail;
public;
List() {
head = tail = NULL;
}
};

LL functions

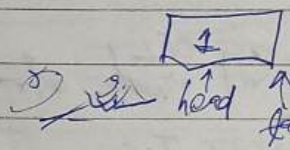
i) push_front() → pop_front
ii) push_back() → pop_back
iii)

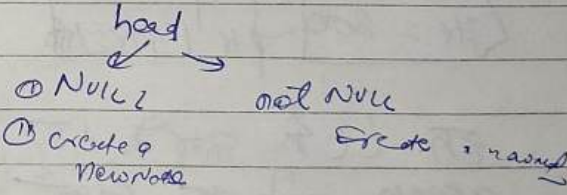
push_front(1)





push - front in LL





void push_front(int val) {
Node * newNode = new Node(val); //dynamic
Node newNode(val); //static
if (head == NULL) {
head = tail = newNode;
} else {
newNode->next = head;
head = newNode;
}

9 November 2025 at 16:01

[2] → [1]
head   tail

```
int main() {
    list L1;
    L1.push_front(1);
    L1.push_front(2);
    L1.push_front(3);

    return 0;
}
```

Steps: head → next NULL
① create a new...
② newNode → next = head
③ head = newNode

[3] → [2] → [1]

क्योंकि (*newNode).next = head
या newNode → next = head

print LL

indexing इसमें head से किया जाता है, not tail, most case

[3] → [2] → [1] → NULL
head temp tail

```
Node* temp = head
while (temp != NULL) {
    cout << temp->data << " ";
    temp = temp->next
```

प्रश्न:- हम क्यों यहाँ temp बना रहे why not direct use head?
उत्तर:- क्योंकि LL में Backward traversal करना impossible है।
इसलिए copy बनाते हैं।

Push Back in LL

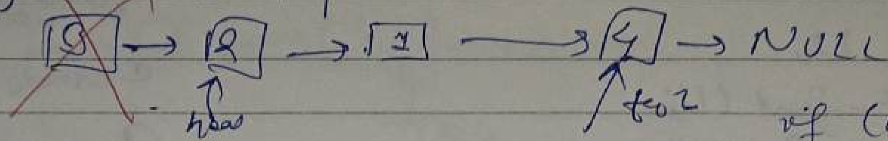
[3] → [2] → [1] → [4] → NULL
head   tail

head != NULL
① create newNode
② tail → next = newNode
③ tail = newNode

```
void push_back(int val) {
    Node* newNode = new Node(val);
    if (head == NULL) {
        head = tail = newNode;
    } else {
        tail -> next = newNode;
        tail = newNode;
```

9 November 2025 at 16:01

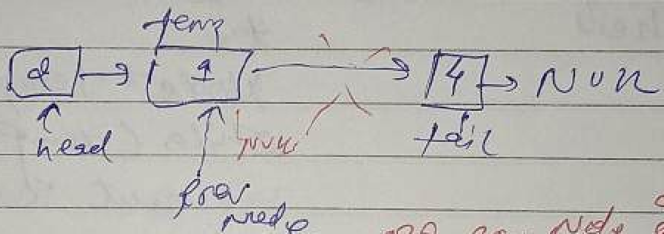
Pop-front & Pop Back




```
void pop-front() {
  if (head == NULL) {
     cout << "LL is empty \n";
     return;
  }
  Node* temp = head;
  head = head -> next;
  temp -> next = NULL;
  delete temp;
```

```
if (head == NULL)
Node* temp = head
head = head -> next
temp -> next = NULL
delete temp;
```

Pop Back



TC = O(n)

```
if (head == NULL)
      return;
Node * temp = head;
while (temp -> next != tail) {
         temp = temp -> next;
} // Break connect prev node
  temp -> next = NULL

delete tail
tail = temp;
```

यहाँ, prev Node को disconnect करें

temp -> next == tail)

या temp -> next -> next == NULL)

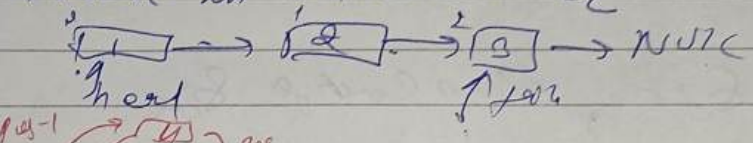
9 November 2025 at 16:02

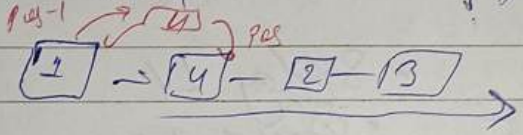
```
void pop_back() {
    if (head == NULL) {
        cout << "LL is empty";
        return;
    } else {
        Node* temp = head;
        while (temp->next != tail) {
            temp = temp->next;
        }
        temp->next = NULL;
        delete tail;
        tail = temp;
    }
}
```

Insert in Middle in LL

insert(val, pos);

insert(4, 2)





① Create newNode

Node* temp = head

```
void insert(int val, int pos) {
    if (pos < 0) {
        cout << "Invalid";
        return;
    }
    if (pos == 0) {
        push_front(val);
        return;
    }
    Node* temp = head;
    for (int i = 0; i < pos-1; i++) {
        temp = temp->next;
    }
    Node* newNode = new Node(val);
    newNode->next = temp->next;
    temp->next = newNode;
}
```

for (int i=0; i<pos-1; i++) {
    temp = temp->next;
} temp;

new Node ->next = temp->next

temp->next = new Node

9 November 2025 at 16:02

Contig → (ArrayList) V/S LinkedList → non contig mem

Insert - O(n) O(1)

Search - O(1) O(n) search (key)

SEARCH in LL

[1] → [2] → [3] → NULL

0 1 2

temp idx = 0

Key = 2 → (1)

-1

```
int search (int key) {
    Node* temp = head;
    int idx = 0;
    while (temp != NULL) {
        if (temp→data == key) {
            return idx;
        }
        temp = temp→next;
        idx++;
    }
    return -1;
}
```

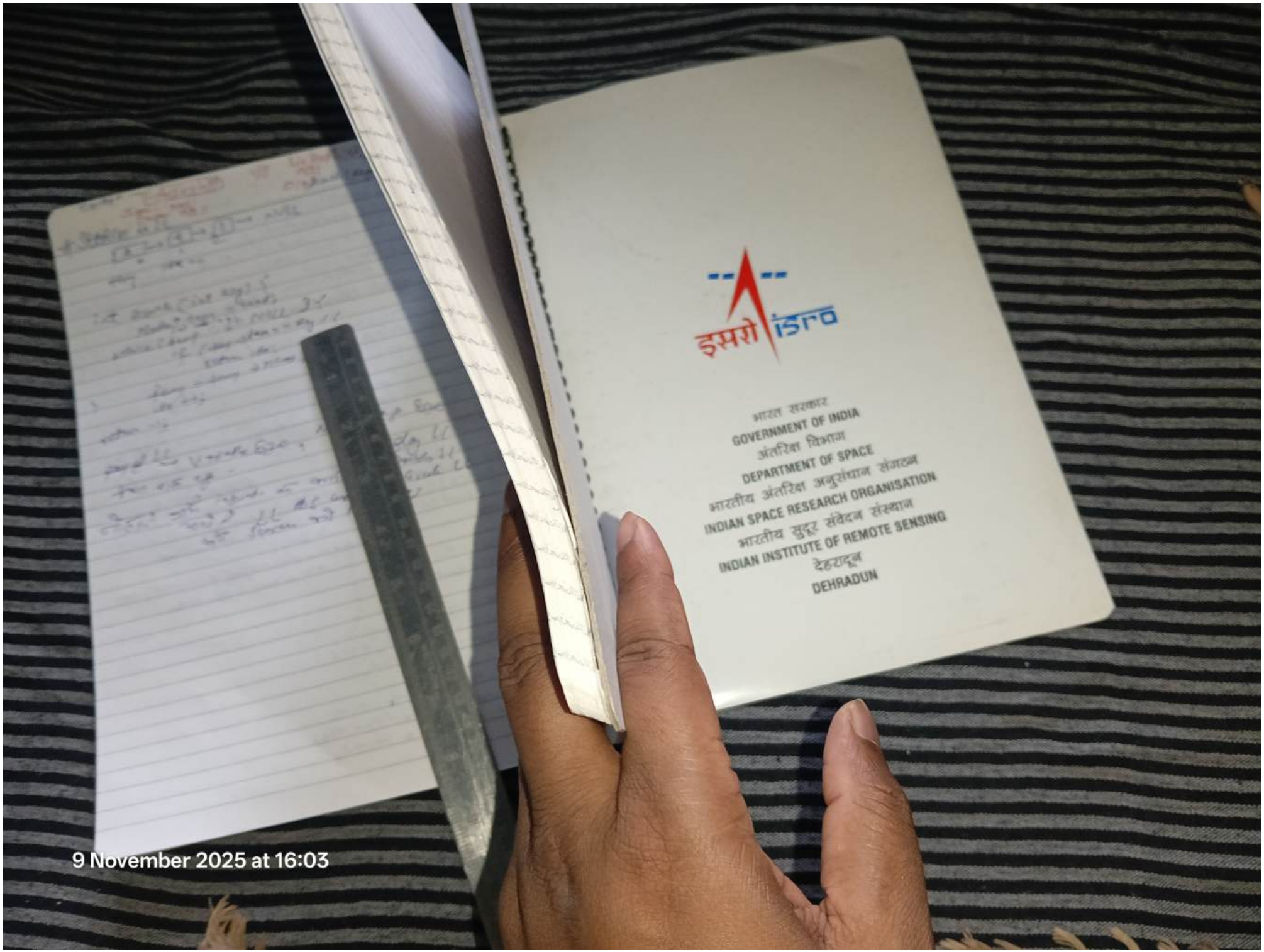
prog of LL → Variable size, Non Contig size

train दिब्बा eg -

- Singly LL
- Doubly LL
- Circular LL

नोट:- जहाँ indexing का काम आता है LL वहाँ यूज़ नहीं कर सकते search करने के लिए O(1)

9 November 2025 at 16:02


इसरो isro
भारत सरकार
GOVERNMENT OF INDIA
अंतरिक्ष विभाग
DEPARTMENT OF SPACE
भारतीय अंतरिक्ष अनुसंधान संगठन
INDIAN SPACE RESEARCH ORGANISATION
भारतीय सुदूर संवेदन संस्थान
INDIAN INSTITUTE OF REMOTE SENSING
देहरादून
DEHRADUN
9 November 2025 at 16:03