Lecture :- Linked List Basics

Agenda

class [DSA specific]

String str = "Ayush";

String phoneNO = "8287457773";

int rollno = 23;

char gender = M;

class is a user-defined data type combining all data types.

```
class student {
    String name;
    String phoneNO;
    int rollno;
    char gender;
}
```

Objects real life instance of a class.

```
class Actor {
    String name;
    int noOf100Cr Movies;
    String nextMovie;
    char gender;
}
```

Class name — Keyword — constructor

```
Actor ranbirkapoor = new Actor();
    ranbirkapoor.name = "Ranbir Kapoor";
    ranbirkapoor.noOf100CrMovies = 5;
    ranbirkapoor.nextMovie = "Animal";
    ranbir kapoor.gender = 'M';
```

Constructor

used to initialize an object.

used to " attributes of class.

name of constructor = class name. ✓

constructor is a func without any return type. ✓

Class can have multiple constructors.

```
class Actor {
    String name;
    int noOf100cr movies;
    String nextMovie;
    char gender;

no return type
    Actor ( ) {
        ⟹ default constructor.
    }

    Actor ( String n. ) {
        name = n;
    }
```

```
Actor srk = new Actor();
print (srk.name) // "", null

Actor srk1 = new Actor("Shah Rukh Khan");
print( srk1.name) || Shah Rukh Khan
```

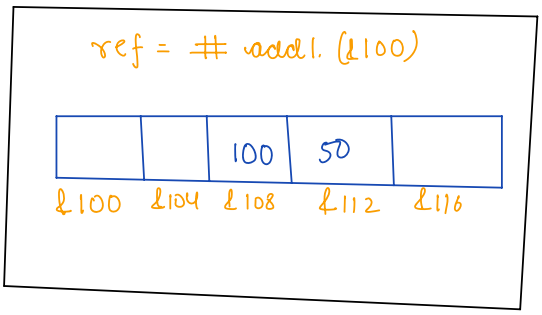
Array memory Allocation [continuous memory allocation]

```
int[] arr = new int[5];
```

↑ dynamic memory allocation

[heap space]

heap space



arr[2] = 100; // &100 + 2*4 = &108

↓

arr = &100

int = 4 bytes.

arr[3] = 50 // &100 + 3*4 = &112

arr[3] = 50 // I need to go to address &112

I only know &100 ⇒ arr[0], arr

Issues with Arrays

1. fixed size → Overcome: ArrayList, LinkedList

2.

heap space

100	104	108	112	116
120	124	128	132	136
140	144	148	152	156
160	164	168	172	176

int a = 10; & 104

int b = 20; & 168

int c = 30; & 176

int d = 40; & 120

int e = 50; & 132

int[] arr = new int[5] → not possible to accomodate the array within heap space

[not continuous allocation]

Linked List

A linked list is a linear data structure of variable/ dynamic size and the elements are stored in non-continuous manner.

heap space

100	104	108	112	116
120	124	128	132	136
140	144	148	152	156
160	164	168	172	176

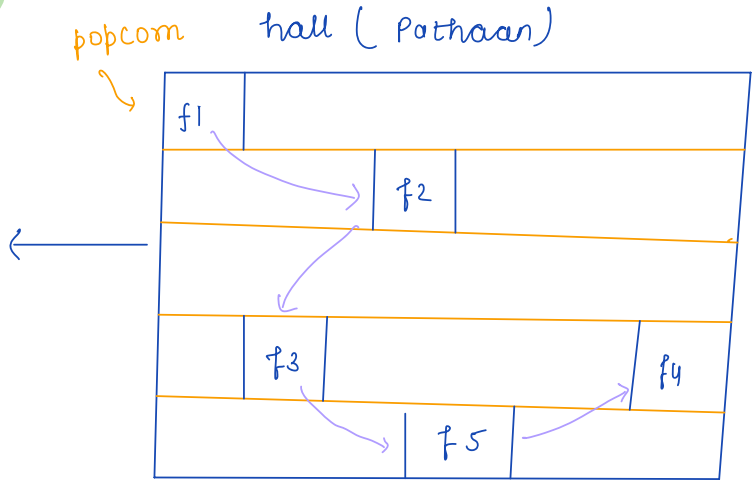
Linked list of size 5

↓

head [&104 &120 &132 &168 &170]

Structure of Linked list

5 friends



f1 → f2 → f3 → f5 → f4

observation

A friends stores / remembers the address of next friend.

heap space

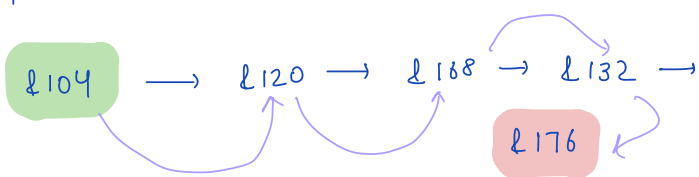
100	104	108	112	116
120	124	128	132	136
140	144	148	152	156
160	164	168	172	176

Linked list of size 5

vaud [&104, &120, &132, &168, &176]

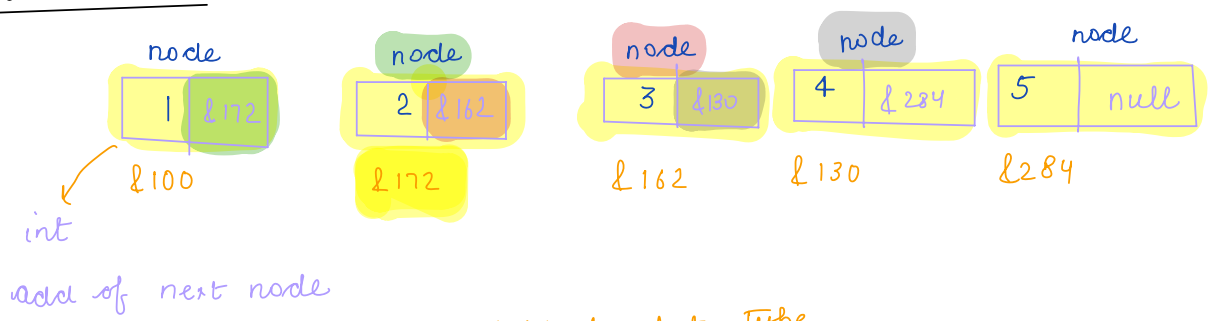
I already know &104

&176 ?



Array

Linked list



user defined data type

```
class Node {

    int val;

    Node next;

    Node(int x) {

        val = x;
    }
}
```

val
next
val
next

Creation of LL

```
class Node {
    int val;
    Node next;
    Node(int x) {
        val = x;
    }
}
```

user defined data type

val

next → val, next

```
Node n1 = new Node(10);
```

ref1

val = 10

next =

```
Node n2 = new Node(20);
```

ref2

val = 20

next =

```
n1.next = n2.
```

ref1

val = 10

next = n2 [# ref2]

```
print(n1.val)    // 10
print(n2.val)    // 20
print(n1.next)   // null
```

```
print(n1.next)   // # ref2
```

```
print(n1.next.val)   // 20
```

n2 (# ref2) . val

```
print(n1.next.next)   // null
```

n2 (# ref2) . next

```
print(n1.next.next.val)
```

null . val

Null pointer exception

Break: 8:39 AM

Input format of LL

head → 1 → 2 → 3 → 4 → 5

&101 &202 &108 &382 &505

val = 1, next = &202

val = 2, next = &108

Always preserve your head.

Qu1 Print a LL.

1 → 2 → 3 → 4 → 5 → null

hc hc hc hc hc

```
void printLL (Node head) {

    Node hc = head;

    while ( hc != null) {

        print( hc.val); // 1 2 3 4 5

        hc = hc.next;
    }
}
```

1 → 2 → 3 → 4 → 5 → null

headcopy hc hc

print (headcopy.val) // 1

headcopy = headcopy.next

& add of 2

print (headcopy.val) // 2

headcopy = heapcopy.next;

& 3

print (headcopy.val) // 3

Qu **Insert in a LL**

Case 1: Insert at head

ip 1 → 2 → 3 → 4 → 5 → null

val = 8

op 8 → 1 → 2 → 3 → 4 → 5 → null

```
void insertAtHead(Node head, int val) {
    // Create a node that you want to insert
    Node newNode = new Node(val);
    // connect newNode to head
    Update the head
    newNode.next = head;    // 8 → 1
    head = newNode;
}
```

TC: O(1)

Case2: Insert at end of LL

ip 1 → 2 → 3 → 4 → 5 → null

val = 8

op: 1 → 2 → 3 → 4 → 5 → 8 → null

&5.next = &8

create this node

observation

1. Create the node of ip val
2. Go to add 5. &5.next = &8.

ip 1 → 2 → 3 → 4 → 5 → null

hc hc hc hc hc (hc.next != null)

```
void insertAtEnd( Node head, int val) {
    // Create a node that you want to insert
    Node newNode = new Node( val);
    Node hc = head;
    while( hc.next != null) {
        hc = hc.next;
    }
    hc.next = newNode;
}
```

val = 8
next = null

TC: O(n)

case3: Add after any idx.

ip: 0: 1 → 1: 2 → 2: 3 → 3: 4 → 4: 5 → null

idx = 3 , val = 60

op: 0: 1 → 1: 2 → 2: 3 → 3: 60 → 4: 4 → 5: 5 → null

↑ head

create this node

ip: 0: 1 → 1: 2 → 2: 3 → 3: 4 → 4: 5 → null

↑ hc

i = 0 , hc = head. &1

i = 1 , hc = hc.next &2

i = 2 , hc = hc.next &3.

while (i < idx)

temp = &3

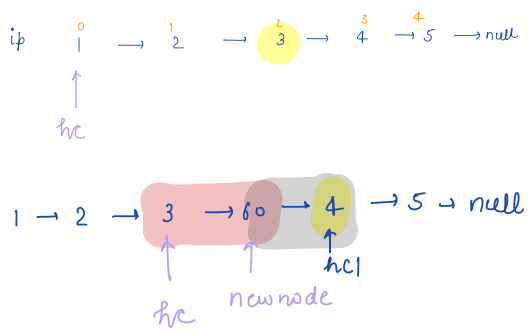
temp1 = temp.next = &4

newNode = &60

temp.next = newNode

newNode.next = temp1.

```
void insertAtK(Node head, int k, int val) {
    // Create a node that you want to insert
    Node newNode = new Node(val);

    Node hc = head;

    int i = 0;

    while( i < k) {
        hc = hc.next;
        i++;
    }

    Node hcl = hc.next;

    hc.next = newNode

    newNode.next = hcl;
}
```



TC: $O(k) \simeq O(n)$

Thanks you ☺

Doubts:

user defined data type

```
class Node {
    string name
    int val;
    Node next;
    int val2;
    Node(int x) {
        val = x;
    }
}
```

val

next → val, next

```
Node n1 = new Node(10);
    n1.val = 10;
    n1.next = null
    n1.val2 = 0
    n1.name = "" (or) null
```

A = 4

B = 4

Row1 : 0

Row2: 0 1

Row3. 0 1 1 0

Row4 : 0 1 1 0 1 0 0 1 = 8

4

row 4

row3

row4 comp (row3)

fun(A, B) {

4 6

= fun((A-1), (B/2))

B is even — same

B is odd — complement