unacademy

# Linked List - Part II

Course on Data Structure

**Pankaj Sharma** • Lesson 8 • July 16, 2024

unacademy

# CS & IT Engineering

## Data Structure

### Linked List-1

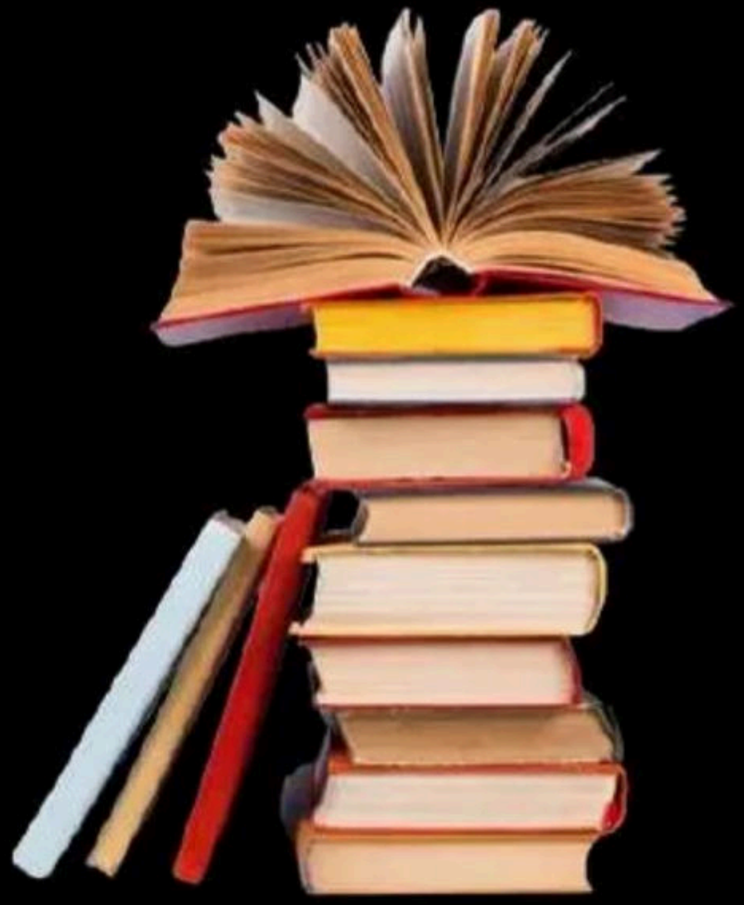
Lecture Number- 07

By- Pankaj Sir

unacademy

# Topics

*to be covered*

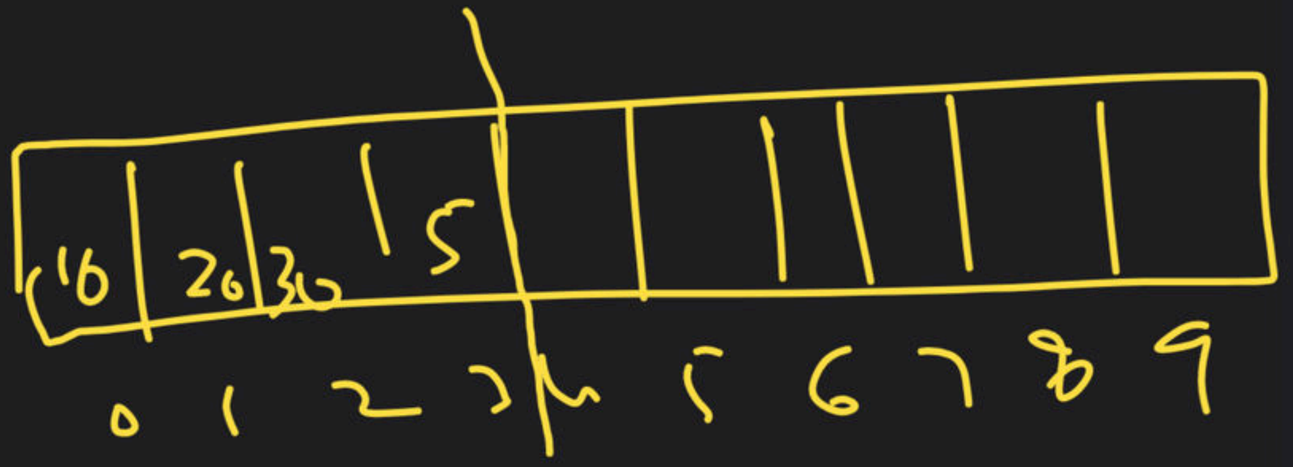


1. Linked List

unacademy

# Array

```
#define SIZE 10
void main(){
    int A[SIZE];
    ==
}
```




n = 4

n : no. of ele. in the array

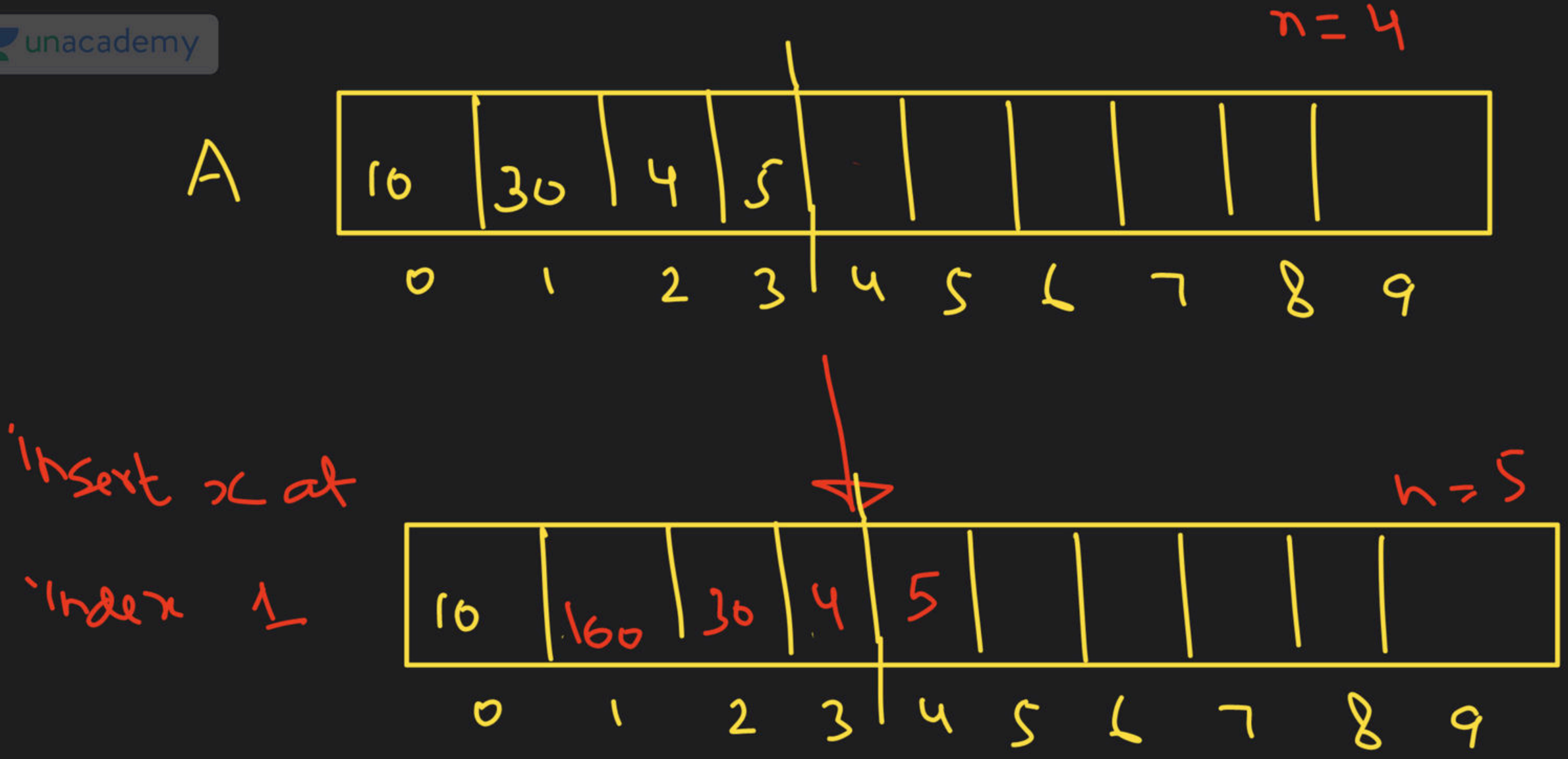
n = 4 5

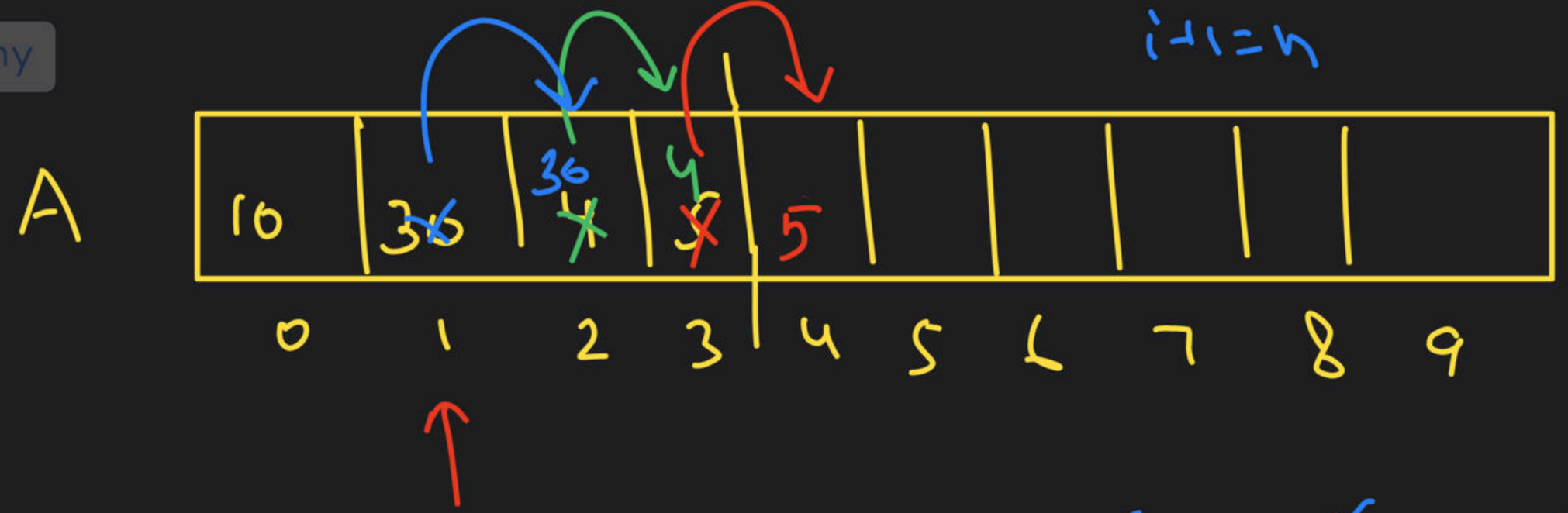

(1) insert : at the end

```
A[n] = x;
n = n+1;
```

n = 4
x = 100

A[4] = 100


n = 4
A
10
30
4
5
0 1 2 3 4 5 6 7 8 9
insert x at
index 1
n = 5
10
160
30
4
5
0 1 2 3 4 5 6 7 8 9



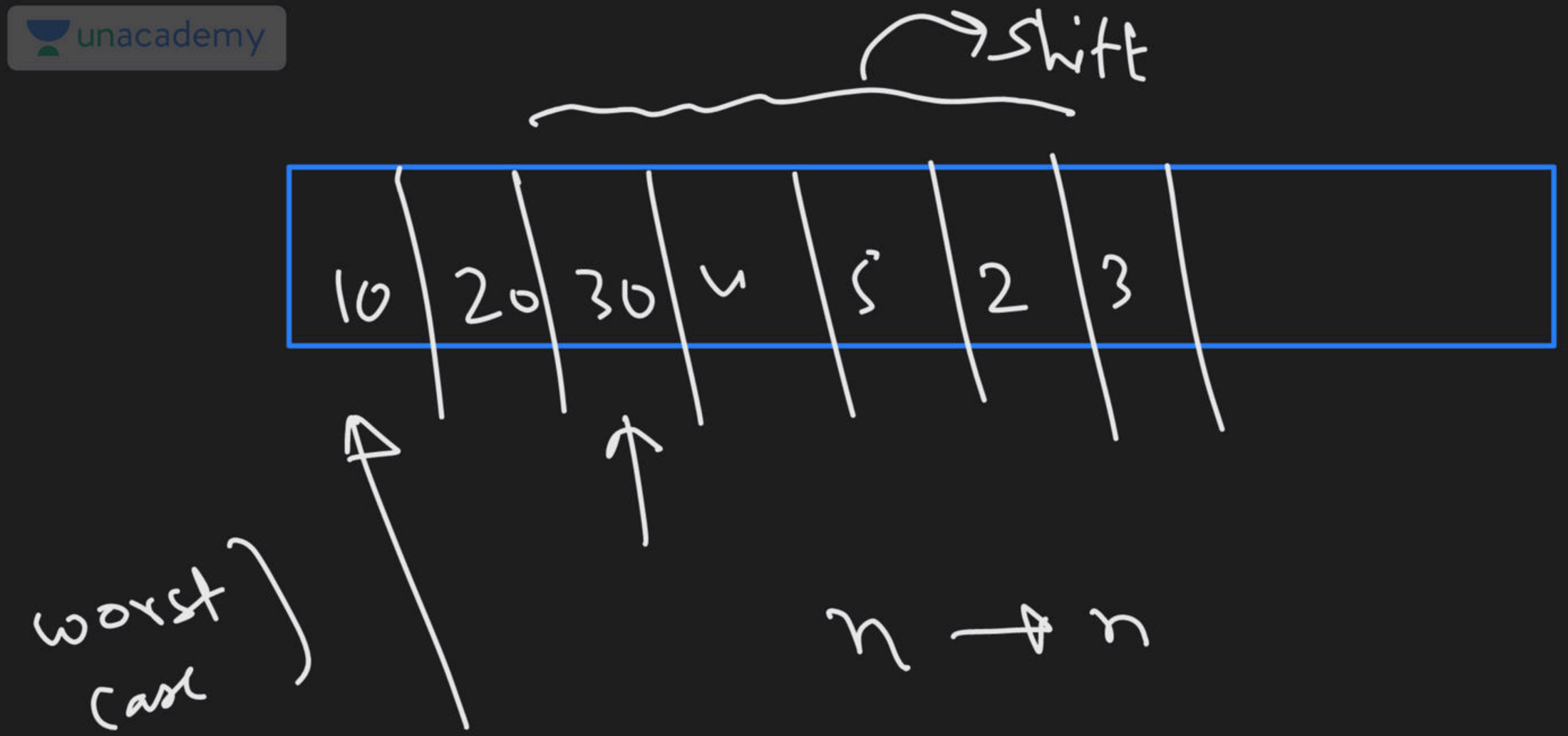

```
for(i=n-1; i>=index; i--)
    A[i+1]=A[i];
```

```
for(i=n; i>=index+1; i--)
    A[i]=A[i-1];
```

A[4]=A[3]

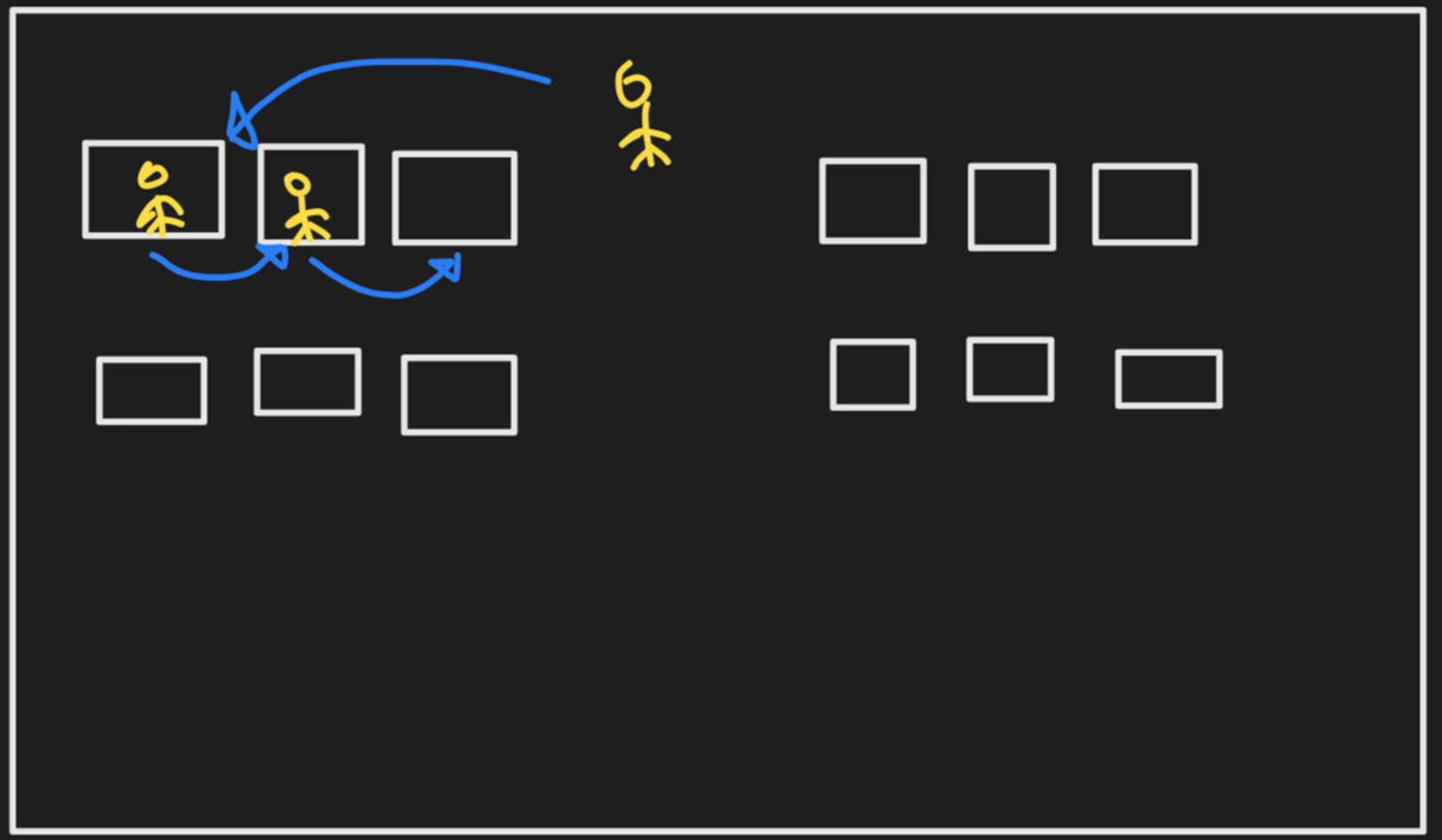
A[3]=A[2]

A[2]=A[1]


shift
10
20
30
4
5
2
3
worst case
n → n

unacademy

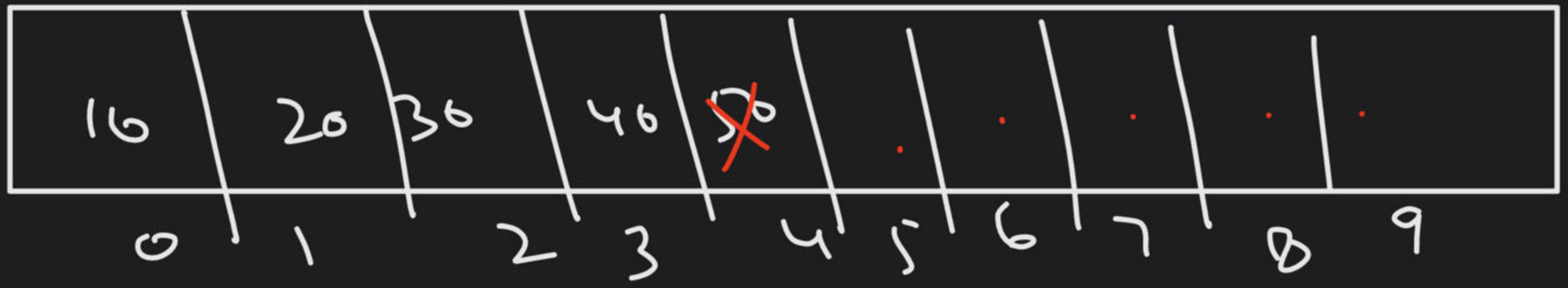
Array insert → O(n)



unacademy

# Deletion

n = 5




end

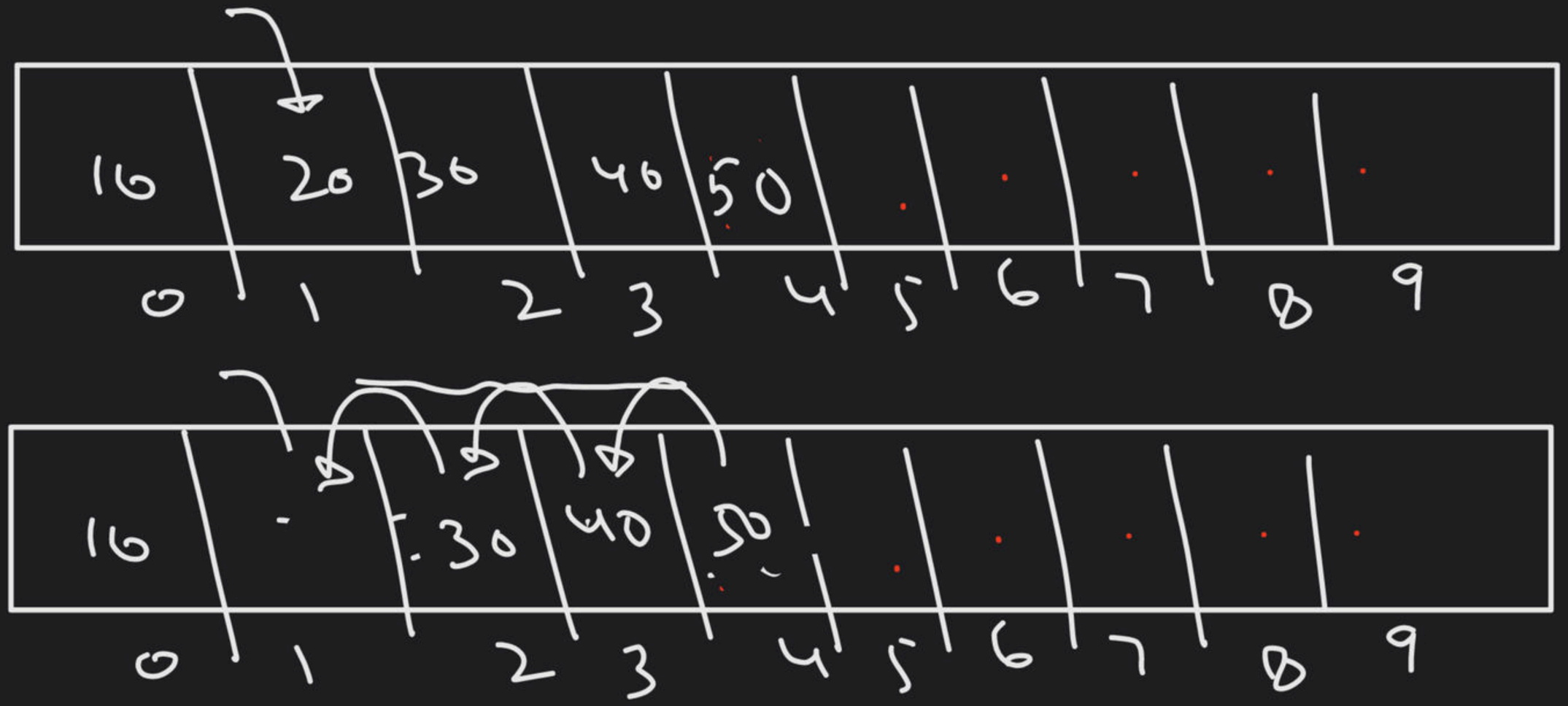
↳ No Tension

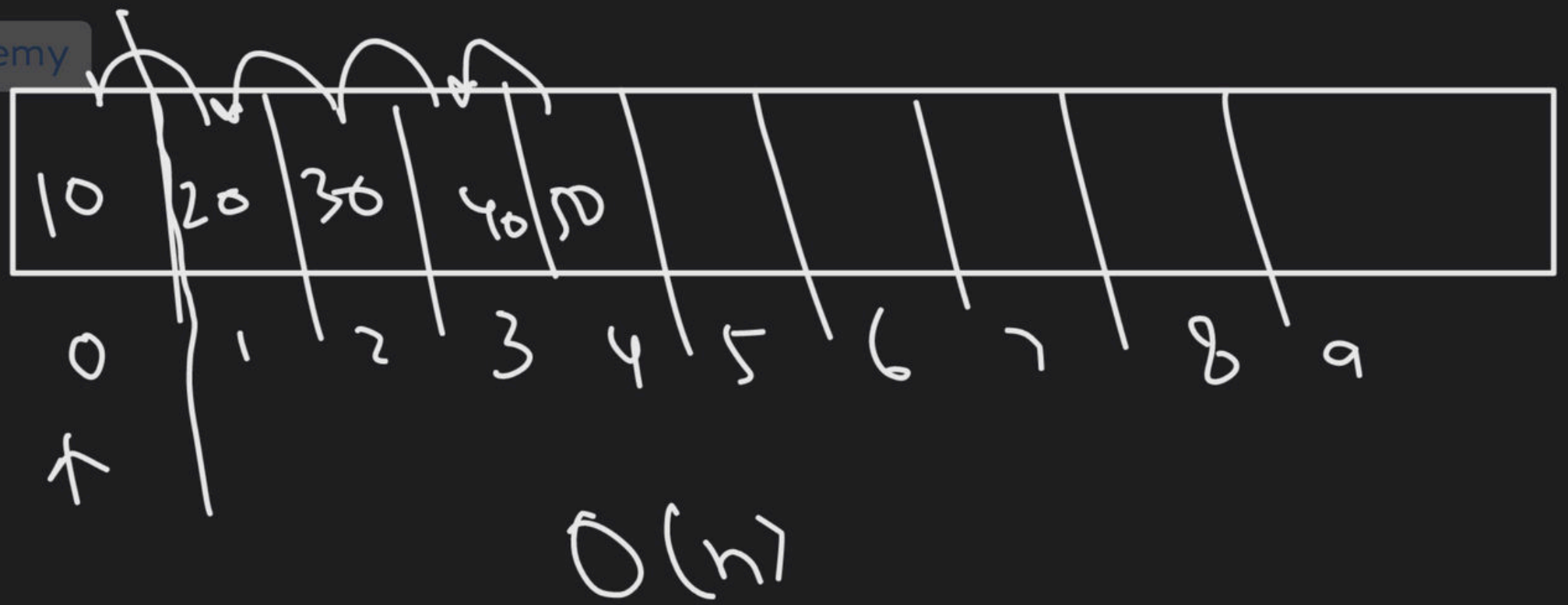
h = h-1;

unacademy

# Deletion

$h = 5$






O(n)

unacademy

# Arrays

① size → fix

Static

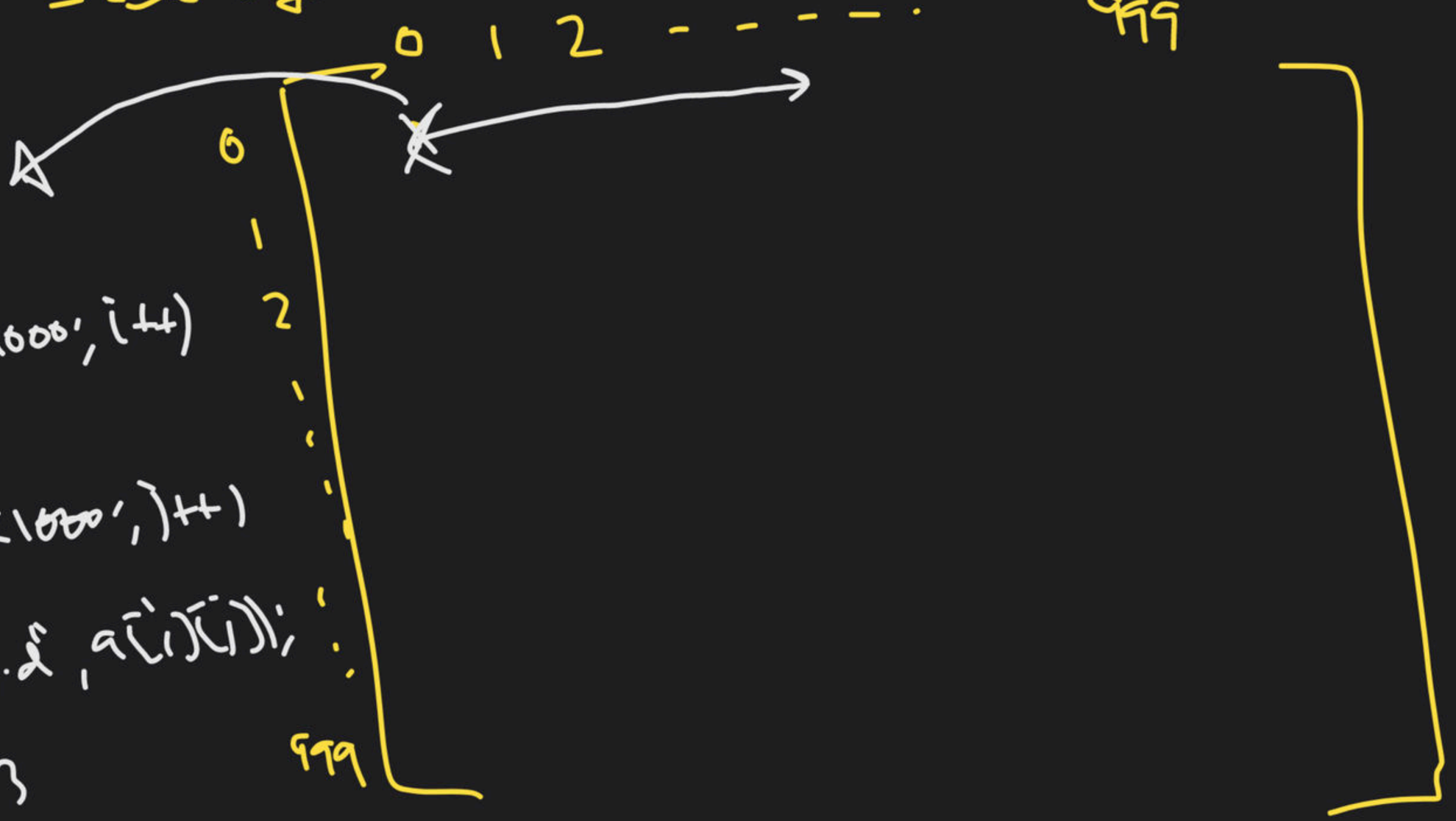
② Insert/Delete → time consuming

Adv.

(i) Random Access : Constant time ✓

(ii) Cache Friendliness

1KB = 1024 B = 256 integer
4



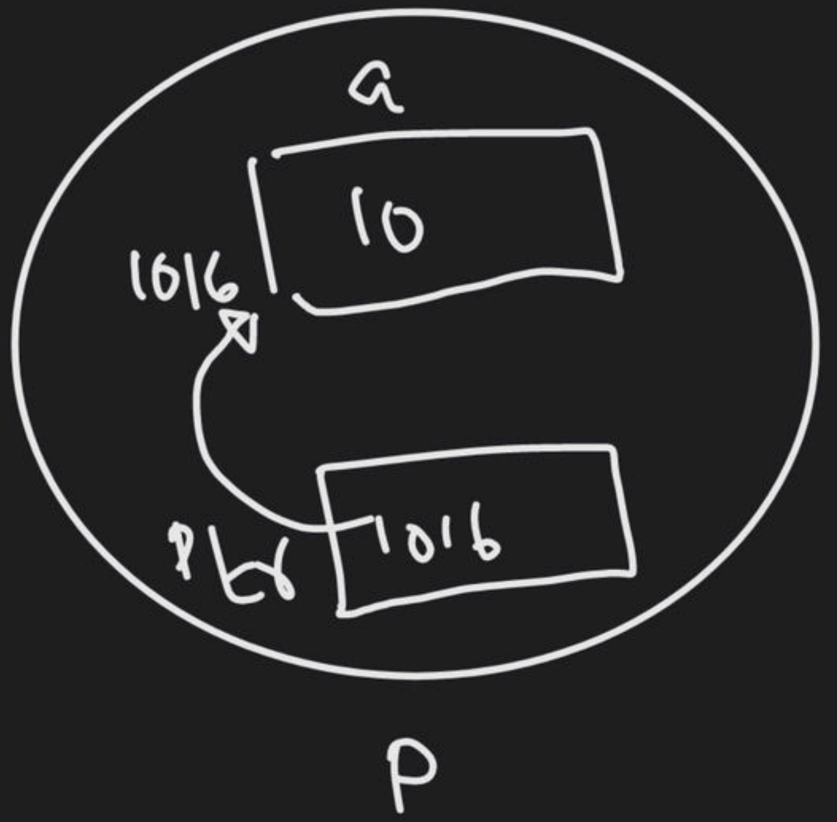

```
for(i=0; i<1000; i++)
{
for(j=0; j<1000; j++)
scanf("%d", &a[i][j]);
}
A[0][0]
```

# Performance improvement

unacademy

# Structure

```
struct Pankaj {
    int a;
    int* ptr;
};
void main() {
    struct Pankaj p;
    p.a = 10;
```

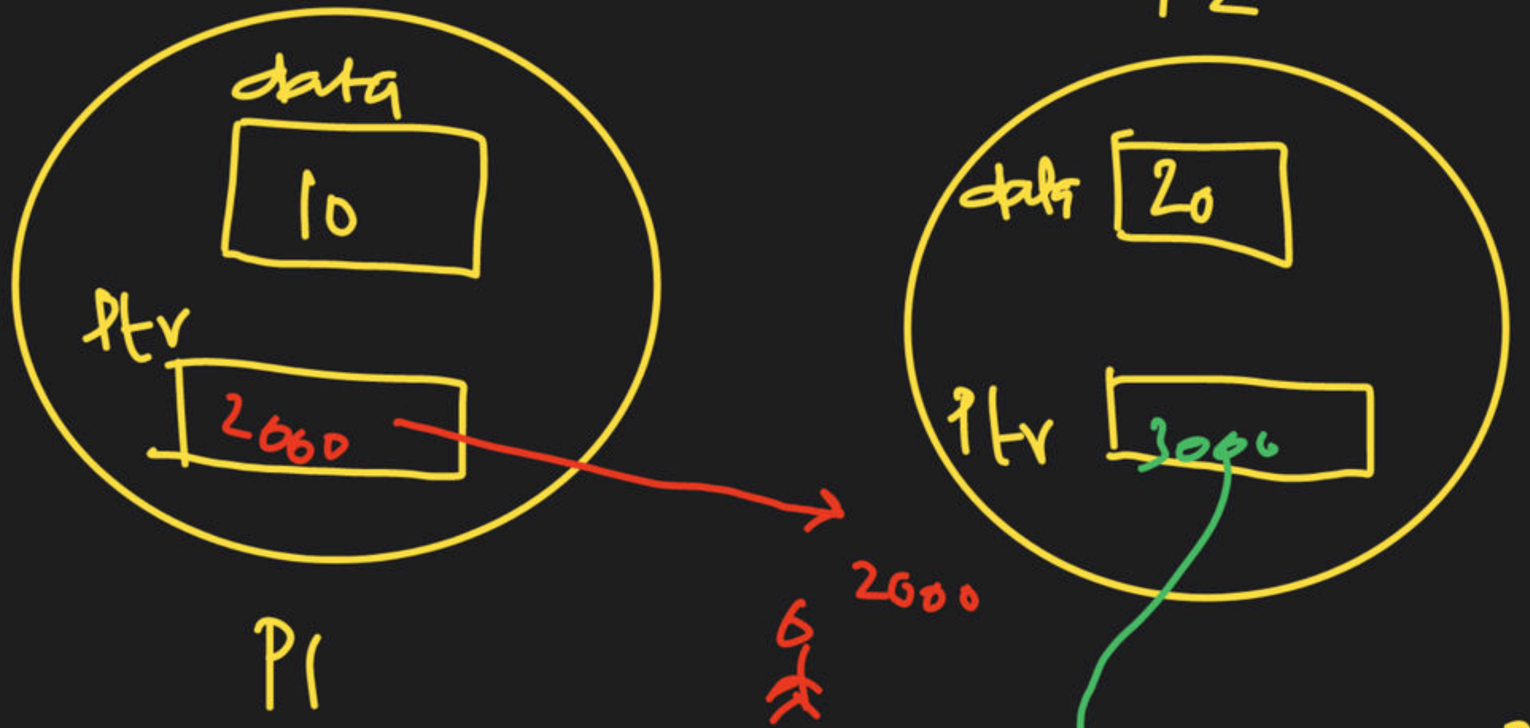



```
p.ptr = &p.a;
```

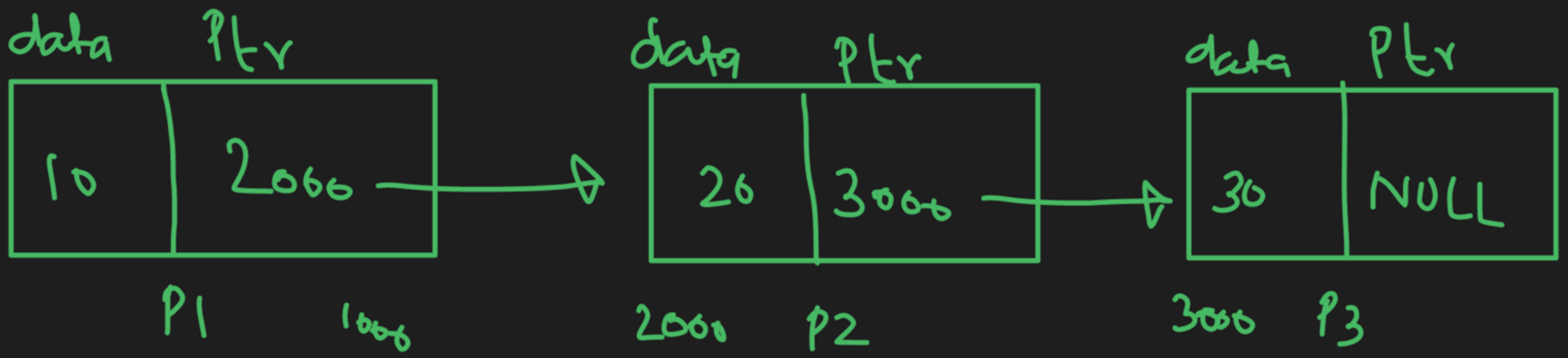
unacademy

# Self-Refrencial structure

```
struct Pankaj {
int data;
struct Pankaj *ptr;
};
void main(){
struct Pankaj P1, P2, P3;
P1.data = 10; P2.data = 20; P3.data = 30
P1.ptr = &P2; P2.ptr = &P3; P3.ptr = NULL;
```



Self - Referencial structure

unacademy

# Linked List

A linked list is a linear data structure, which is collection of elements called nodes, in which every node is divided into 2 parts

(i) data

(ii) Contains address of next node in linked list
(Pointer to next node)

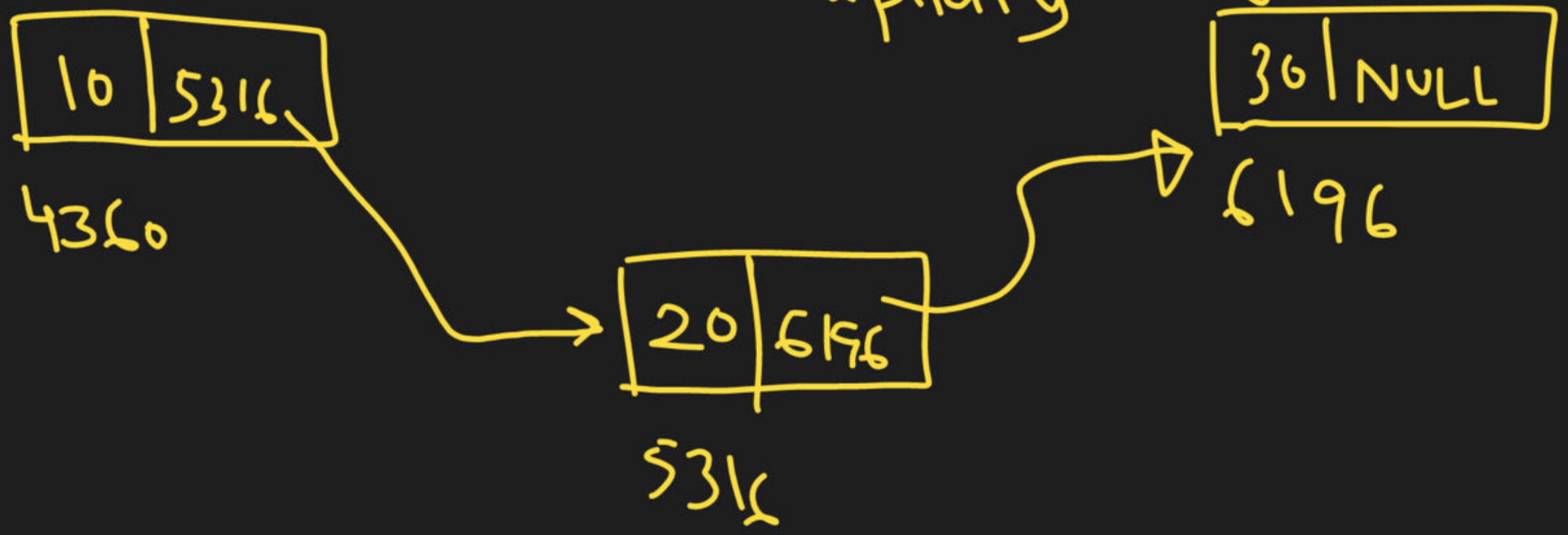
unacademy

Array → Linear d.s

linear

implicit

Linear order is maintained explicitly using pointers



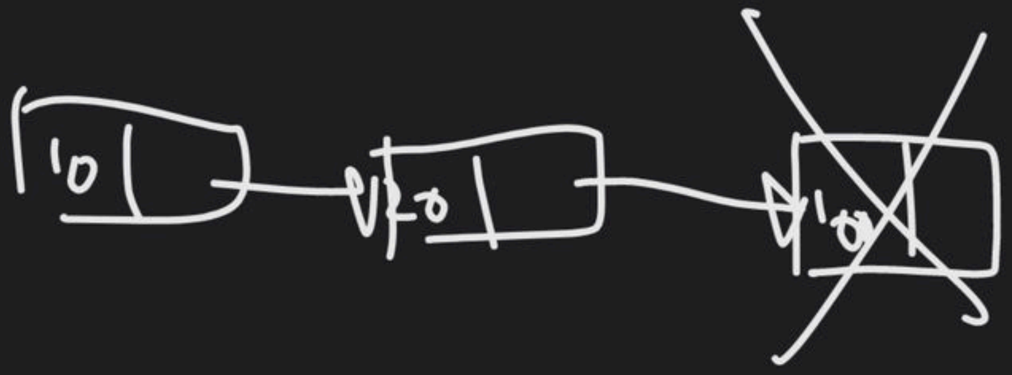
```
struct Node {
    int data;
struct Node * Next;
};
```

① `struct Node s;` ✗

② `struct Node *ptr;` ✓

```
Ptr = malloc(sizeof(struct Node));
```

```
void main(){
    ≡
    ≡
    Insert(100);
    ≡
    ≡
}
```

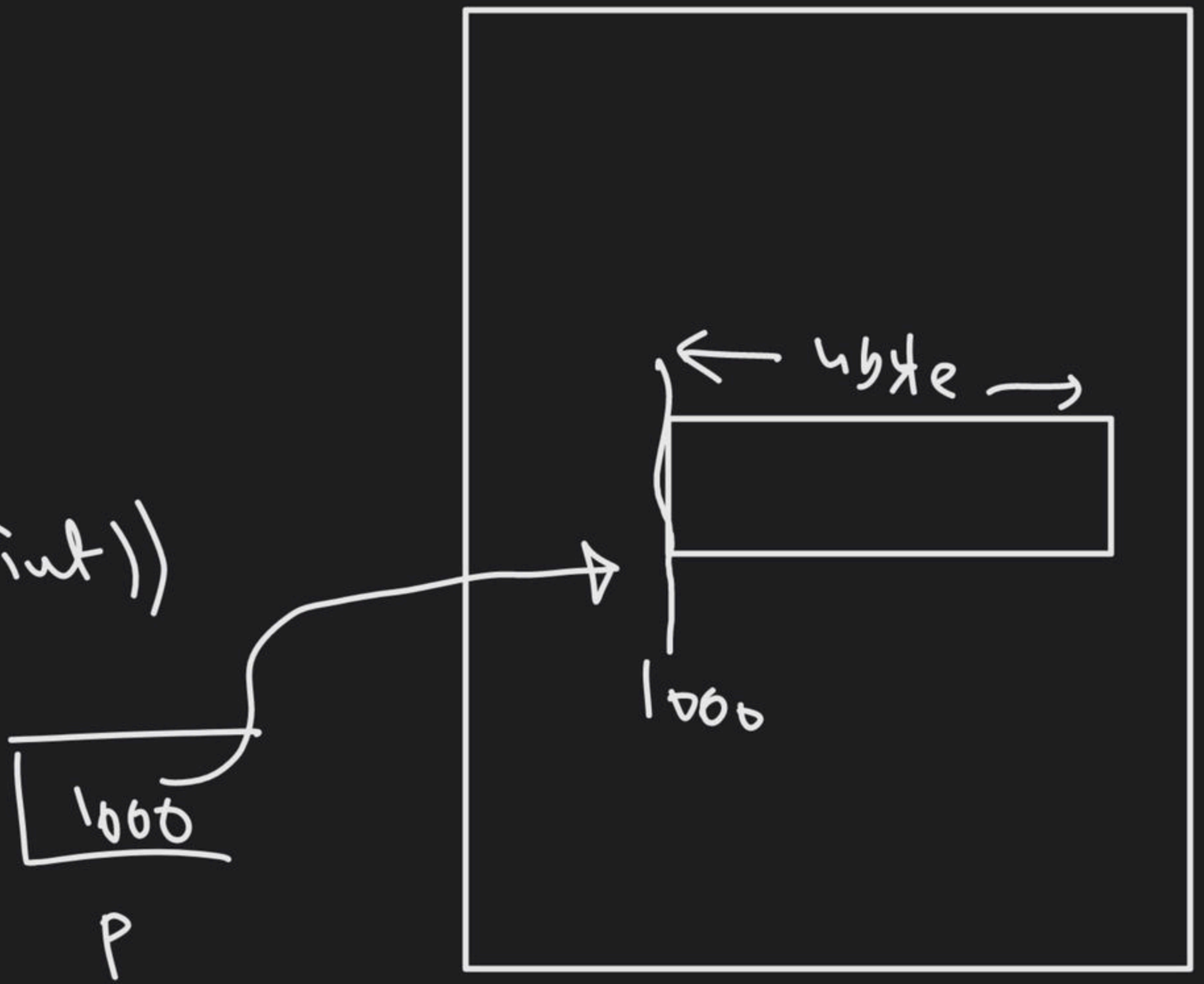



```
void Insert(int n){
    struct Node y;
    ≡
    ≡
}
```

unacademy

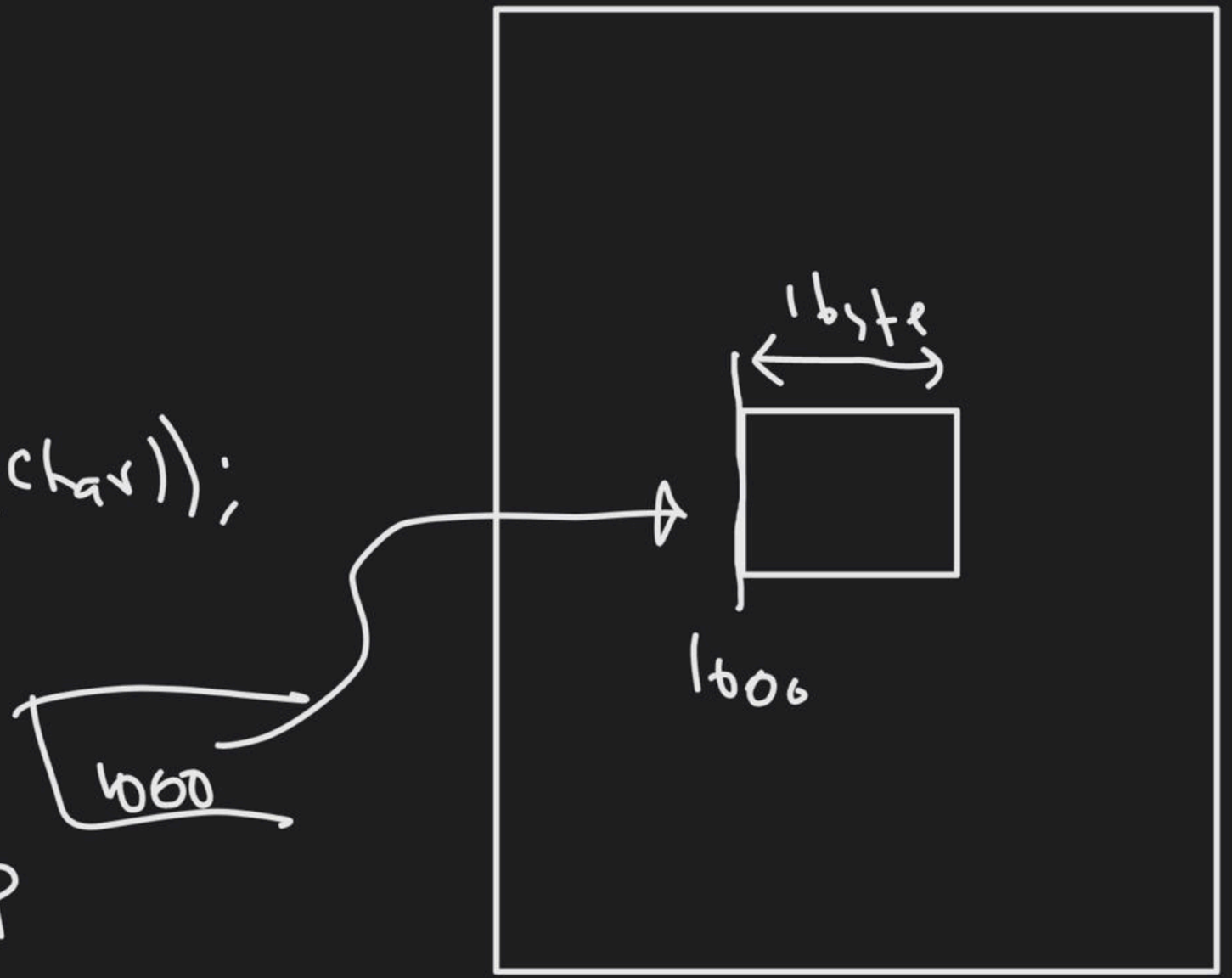
# DMA

```
int *p;
p = malloc (sizeof (int))
```



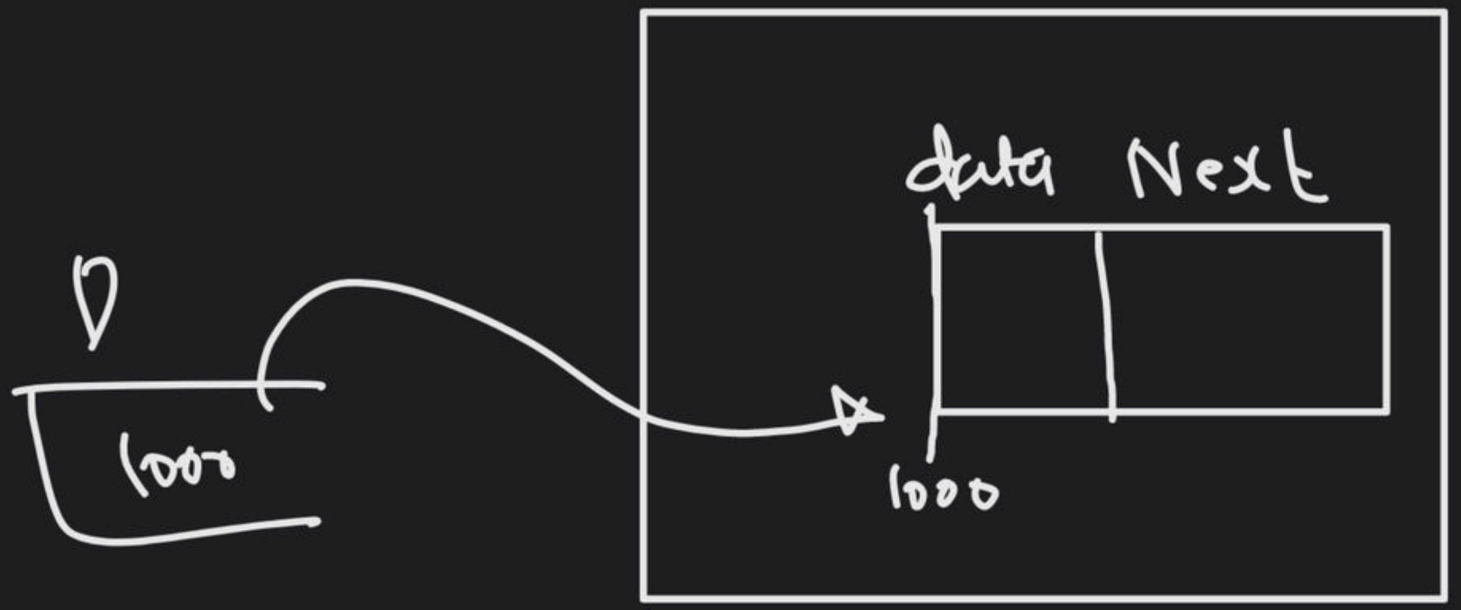
```
char *p;

p = malloc(sizeof(char));
```



```
struct Node {
    int data;
    struct Node *Next;
};
```

```
struct Node *p;
p = malloc(sizeof(struct Node));
```



unacademy

# THANK YOU!

Here's to a cracking journey ahead!