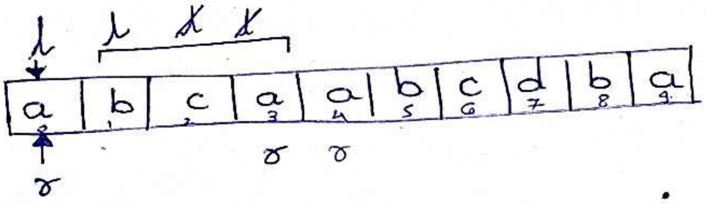
Longest Substring without Repeating Characters

① Naive =
- Generate all substrings (kadane)
- TC = $O(n^3)$ →

  TC = $O(n^2)$

  SC = $O(n)$

② Optimized = hashset.



len = ~~0~~ ~~1~~ ~~2~~ 3
a b c

hashset = d, c, b, a, ~~a~~, ~~c~~, ~~b~~, ~~a~~

$l - r = 7 - 4 + 1 = 4$

$l - r = 3 - 0 = 3$

TC = $O(n) + O(n) = O(2n)$
l pointer, r pointer

SC = $O(n)$

③ best approach =

a b c d e e f g
l ↓ , r ↑

0	1	2	3	4					
a	b	c	a	a	b	c	d	b	a

(d, 7) ---- 4 − 7
(c, ~~2~~) 6 — 4 − 6
(b, ~~1~~) 5
(a, ~~0~~) — move l to 1
~~3~~ 4 — move l to 4

∴ 6 to 9

TC = $O(n)$

SC = $O(1)$ ...... unordered_map.

Count No. of subarrays with XOR as K.
contiguous

arr[] =
[4, 2, 2, 6, 4]

① [4^2] = 6
② [6] = 6
③ [2, 2, 6] = 6
④ [4, 4, 2, 2, 6] = 6

∴ Ans = 4 subarrays

---

① Naive =
- Generate all subarrays =
- TC = $O(n^3)$

```
for (i = 0 to n-1) {
    for (j = i to n-1)
    {
        for (k = i -> j)
            XOR = XOR ^ arr[k];

        if (XOR >= k)
            cnt++;
    }
}
```

---

② better =
- TC = $O(n^2)$

```
for (i = 0 to n-1) {
    XOR = 0
    for (j = i to n-1)
        XOR = XOR ^ arr[j];

        if (XOR >= k) cnt++;
}
```

---

③ Optimal =
- start, end of subarray.

$\boxed{X = XR \wedge K}$ 0....6

- Is there a subarray ending at 6 and having XOR of k.

4 2 2 6 4

∴ 4^2^2^6 = 2

2^6 = 4 ∴ 4 is present at front.

$$\begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \\ \end{array}$$
$$\begin{bmatrix} 4 & 2 & 2 & 6 & 4 \end{bmatrix}$$

		Pre/front XOR	x = XR^k
0	XR = Ø4.	4 = 4	
1	XR = 2^4 = 6	FinalCnt = 1	0 = 6^6
.		∴ 0 is present in map. ∴ we found one subarray.	
2	6^2 = 4	2 = 4^6 ∴ we are looking for 2. increase cnt(4) = 1+1 = 2	
3	4^6 = 2 insert in map.	2^6 = 4 (4,2,2) (6,4,2,2) cnt(4) = 2 ∴ FinalCnt = 3	
4	2^4 = 6	∴ 6^6 = 0 ∴ FinalCnt = 4	

(2,1)
(6,1)
(4,~~1~~) 2
(0,1)

hashmap
(PreXOR, Cnt)

$TC = O(n) \times \underbrace{n \log n}_{\text{for map}}$

$SC = O(n)$.

- Flattening of LinkedList =

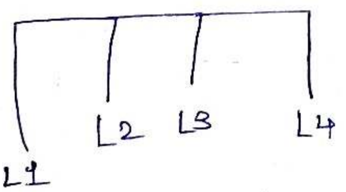
```
19 ——— 28
|       |
22  ,  35
|       |
50     40
        |
       45
```

- merge 2 linkedlists into 1 sorted linkedlist.

```
(0) tmp, res
 |   * 2 pointers at 19 & 28.
 |     19 < 28
(19) tmp
 |     22 < 28
(22)   28 < 50
 |
(28)   35 < 50
 |
(35)   50 > 40
 |
(40)   45 < 50
 |
(45)
 |
(50) tmp
 |
null
```

19.next = null.

-



- Flatten (L3, L4)
- Flatten (L2, L3)
- Flatten (L1, L2).

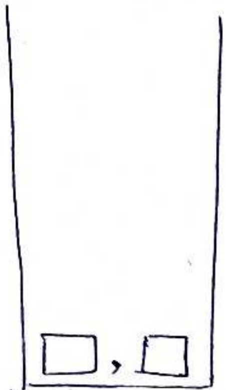
```
     -> f (L1)
L7 (
     -> f (L2)
L6 (
     -> f (L3)
L5 ( ........ merge (L3, L4)
     -> f (L4)
```

TC = O(sum all nodes)

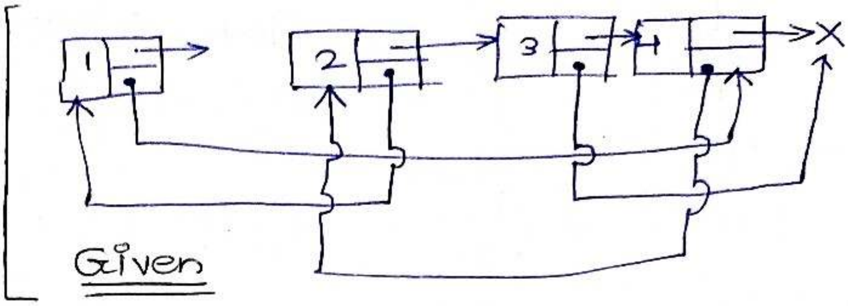
SC = O(1)

Clone LinkedList with next and random pointer =

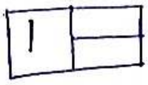
① Naive =



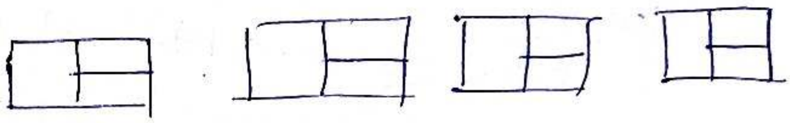
hashmap
<Node, Node>
↑
deep Copy of node



Given



New



- these are the deep copies of given linkedlist.

N^{th} root of an Integer =

① $N=3$ $M=27$ $\sqrt[3]{27}=3$

$1\times1\times1 = \times$
$2\times2\times2 = \times$
$3\times3\times3 = \checkmark$

② $N=4$ $M=69$ $\sqrt[4]{69} = -1$

$1\times1\times1\times1$
$2\times2\times2\times2$
$3\times3\times3\times3$
$4\times4\times4\times4$

```
for( i=1 -> m){
    if ( fun( i, n) == M)
        return i;

    else if( f( i, n) > m)
        break;
}
return -1;
```

$TC = O(M \times \log_2 n)$

② better = Using binary search =

$N = 3$, $M = 27$

low = 1
high = 27

mid = 28/2 = 14
∴ $14 \times 14 \times 14 > 27$
∴ high = mid − 1 = 13

mid = 1 + 13 = 14/2 = 7
$(7 \times 7 \times 7) > 27$
∴ high = 6

mid = 7/2 = 3
$3 \times 3 \times 3 == 27$
∴ return 3.

```
f (n, m)
  low = 1, high = m
  while ( low <= high){
      mid = (l+h)/2 =
      if (f (mid, n) = = m) ret mid;
      else if (f(mid, n) < n)
            low = mid+1;
      else
            high = mid-1;
  }
  return -1;
```

$N = 4$, $M = 69$

low = 1
high = 69

mid = 35
$(35 \times 35 \times 35 \times 35) > 69$
∴ high = 34

mid = 17
$(17 \times 17 \times 17 \times 17) > 69$
∴ high = 16

mid = 8
$8 \times 8 \times 8 \times 8 > 69$
∴ high = 7

mid = 4
$4 \times 4 \times 4 \times 4 > 69$
∴ high = 3

mid = 2
$2 \times 2 \times 2 \times 2 < 69$
∴ low = 3

mid = (3 + 3)/2 = 3
$3 \times 3 \times 3 \times 3 > 69$
low, high crossed
∴ return −1;

$$TC = \log_2 m \times \underbrace{\log_2 n}_{\text{for loop to calculate pow.}}$$

* It will fail.

$n = 10$, $m = 10^9$

low = 1

high = 10^9

mid = $\frac{10^9}{2}$

fun $\left(\frac{10^9}{2}, 10\right) \approx 10^{90}$ Overflow.

$\therefore \frac{10^9}{2} \times \frac{10^9}{2} \times \frac{10^9}{2} \dots\dots$

The moment it crosses 10^9, stop.

∴ return 1 if == n

return 0 if < n

return 2 if > n

```
int fun (mid, n, m){
    long ans=1;
    for(int i=1 ; i<=n ; i++){
        ans = ans *mid;
        if(ans > m) return 2;
    }
    if (ans == m) return 1;
    return 0;
}
```

```
int midN = fun (mid, n, m);
if (midN == 1) ret mid;
else if (midN == 0) low = mid+1;
else high = mid -1;
```

Median of rowwise sorted matrix =

① Naive =
- Use xtra data structure to add ele.
- sort it
- return.

$$TC = \underbrace{(N \times M)}_{\text{put into DS.}} \ \underbrace{(N \times M) \log_2 (N \times M)}_{\text{sort}}$$

$SC = O(N \times M)$.

---

② Using binary search =

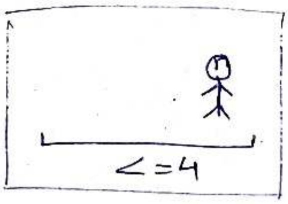
$\begin{bmatrix} 1 & 3 & 6 \\ 2 & 6 & 9 \\ 3 & 6 & 9 \end{bmatrix}$	1 2 3 3 6 6 6 9 9	$\le 1 \to 1$ $\le 2 \to 2$ $\le 3 \to 4$ $\le 4 \to 4$ $\le 6 \to 7$

low = 1
high = 15 i.e. 10^9

mid = 8

≤ 8 rowwise $\longrightarrow 3+2+2 = 7$ ↓↓

∴ low = 1
high = 7
mid = 4
$\le 4 = (2+1+1) \longrightarrow 4$



∴ move right.

low = 5
high = 7
mid = 6

$\le 6 = (3+2+2) = 7$

low = 5
high = 5
mid = 5
$\le 5 \longrightarrow 2+1+1 = 4$

$\cancel{4}$

>4 ✓

∴ low = 6
high = 5

1 2 3 3 $\boxed{6}$ 6 6 9 9

$[1 \longrightarrow 10^9]$

bs

↳ mid → $\boxed{<= mid}$?

$\boxed{<= \frac{n \times m}{2}}$ = low+1

$\boxed{high-1}$

bs
return a[ind]>mid ;

$$TC = \underbrace{\log_2 (2^{32})}_{\substack{10^9 \\ \text{bin-search}}} \times \underbrace{N \times \log_2 M}_{\text{Cnt no-of-ele.}}$$

$$TC = O(32 \times N \times \log_2 M)$$

$$SC = O(1)$$