7th August Morning

1. Multiplication → Addition
2. Merge k Sorted Divide and Conq.
3. Reverse of linked list using addFirst
4. Remove duplicates from sorted list → Add Last / Normal
5. Remove all duplicates
6. Clone a linkedlist with Random pointer
   ↳ O(1) → without using space

7th August Evening

1. Mathematical proof of cyclic linkedlist
2. Clone a linkedlist using Hash Map
3. Segregate odd-Even
4. segregate 01 → swap data / Nodes arrangement
5. segregate 012 → swap data / Nodes arrangement

8th August (Morning)

1. Segregate node of Linkedlist over last Index
2. Segregate Node of Linked list over pivot Index
3. quick sort in linkedlist

8th August (Evening) Doubly linkedlist

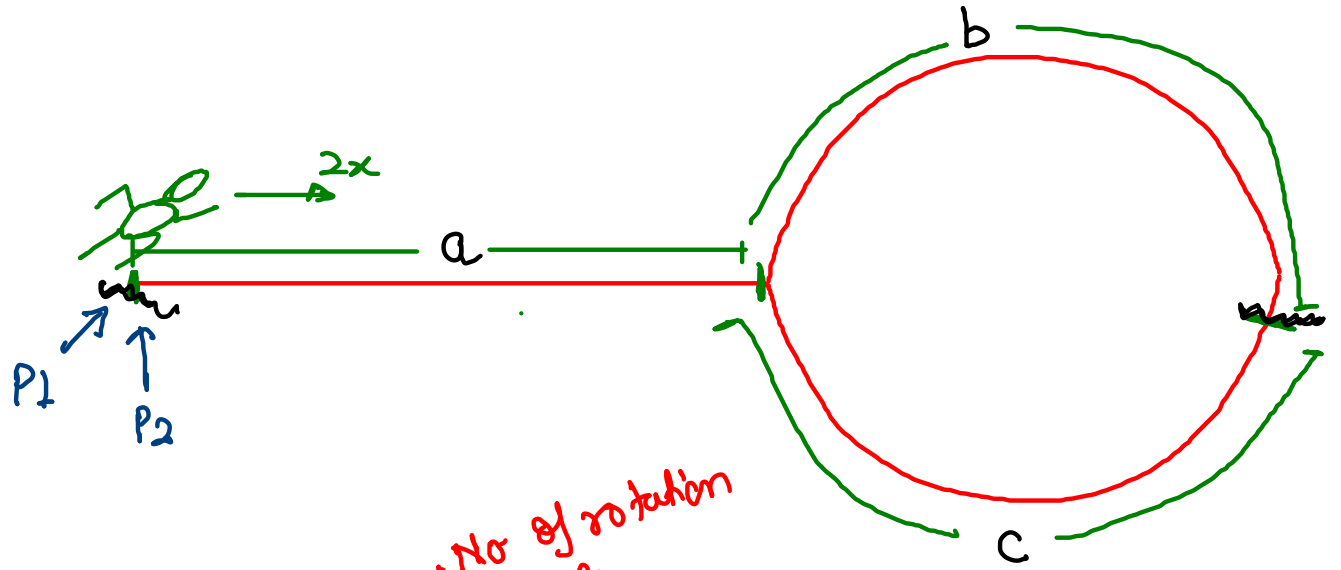
1. Get First + Get Last + Get At
2. Add At
3. Remove At
4. Add Before + Add After
5. Remove Before + Remove After
6. Remove Node in DLL.

Creation of doubly linked list

10th August

1. Display Forward and Backward
2. LRU cache → Application of DLL
3. Miscellaneous.

Proof of cyclic linkedlist :



Speed of P_1 pointer : x

Speed of P_2 pointer : y

No of rotation by P_1

distance travel by P_1 pointer = $a + n(b+c) + b$

distance travel by P_2 pointer = $a + m(b+c) + b$

m → No. of Rotation by P_2

time taken by P_1 = T

time taken by P_2 = T

→ time is same for both pointer

$$\text{Speed} = \frac{\text{distance}}{\text{Time}}$$

$$\Rightarrow \boxed{\text{Time} = \frac{\text{distance}}{\text{Speed}}}$$

Time from P_1 ptr = $\frac{a + n(b+c) + b}{x}$, Time from P_2 ptr = $\frac{a + m(b+c)+b}{y}$.

Both time are equal

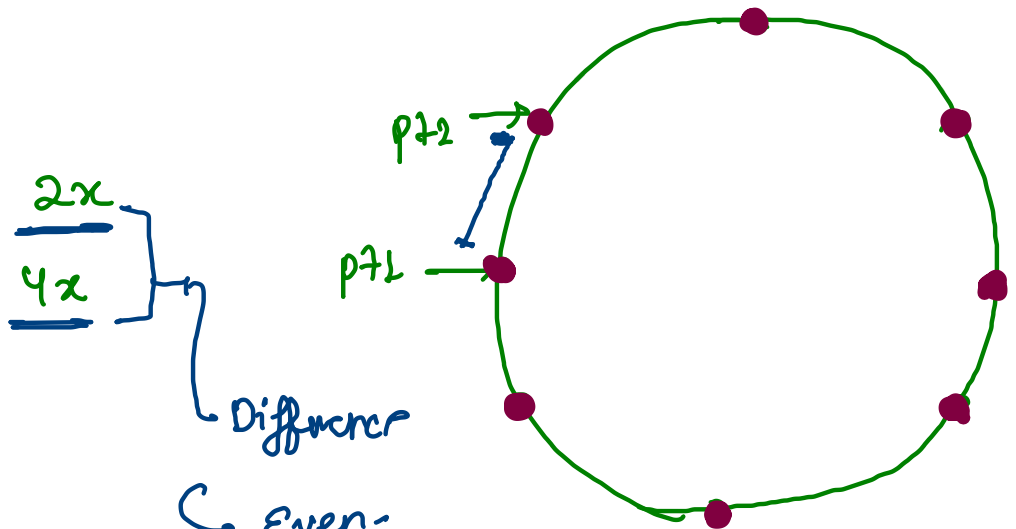
$$\Rightarrow \frac{a + n(b+c) + b}{x} = \frac{a + m(b+c) + b}{y}$$

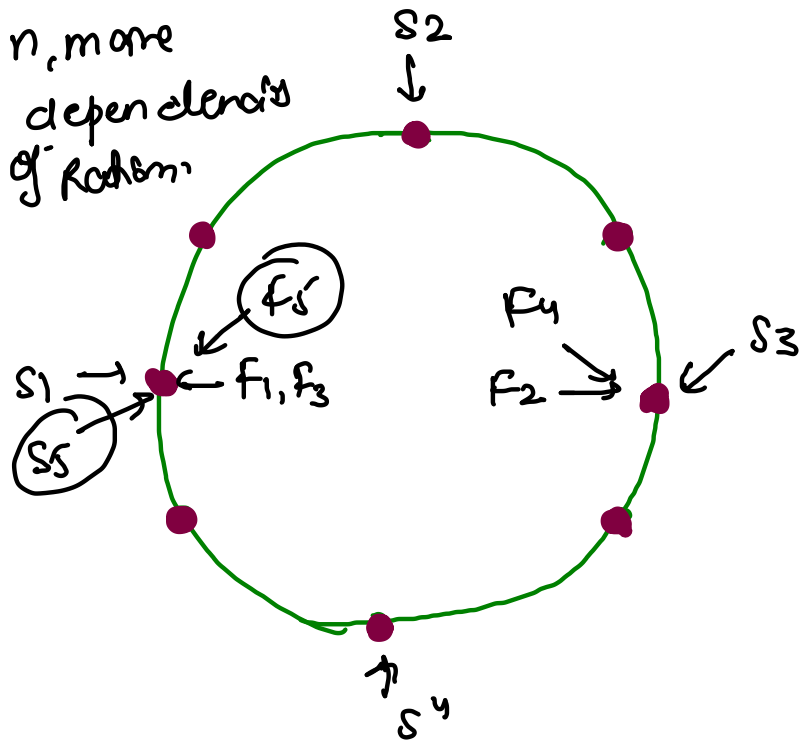
$$\Rightarrow \frac{x}{y} = \frac{a + n(b+c) + b}{a + m(b+c) + b}$$

We can match slow and fast pointer with any ratio of radius.

n, m → significance ??

n, m are dependent of Radius.





$$\frac{x}{y} = \frac{a+n(b+c)+b}{a+m(b+c)+b}$$

Assum $\frac{x}{y} < \frac{1}{2}$

$y = 2x$

$ax^2+bx+c \geq 0$

Assumption speed of P_1 is x

speed of P_2 is $2x$

Rotation of $P_1 = 1$ (slow) Minimum of $n = 1$

$$\frac{x}{2x} = \frac{a+n(b+c)+b}{a+m(b+c)+b}$$

$m = 2n$

$\frac{m}{n} > 2$

Rotation of P_2

if increase n then to maintain the ratio we have to increase m

$m-2n \neq 0$

$a+m(b+c)+b = 2a+2n(b+c)+2b$

$m(b+c) = a+2n(b+c)+b$

$a+n(b+c)+b = 2[a+m(b+c)]$

$n(b+c) = a+b+2m(b+c)$

$m(b+c) - 2n(b+c) = a+b$

$(m-2n)(b+c) = (a+b)$

$m-2n = \left[\frac{a+b}{b+c}\right] \Rightarrow k$ (constant)

① $m \neq 2n$

② k is distance so always +ve that means $m-2n > 0$

$m > 2n$

$\frac{m}{n} > 2$

$\frac{b+c}{a+b} = \frac{1}{m-2n}$

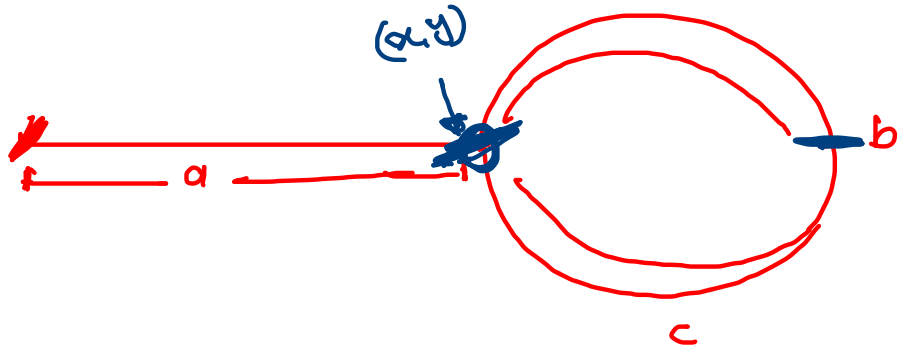
$n = 5$

$m \geq 10$

→ Minimum n should be '1' long not 0, $\left(\frac{m}{n} = \infty\right)$

Limit $\Rightarrow$ $m/n = 2$

$a + n(b+c) + b$



$m = 2n$

$a + 2n(b+c) + b$.

Speed of $P_1 = x$

Rotation of $P_1 = n$

Speed of $P_2 = 2x$

Rotation of $P_2 = m$

$= 2n$.

distance = speed x Time.

Speed x time =

distan. $= x \times T$

$x \times T =$ distan.

$$x = a + n(b+c) + b$$

$a + b = 0$

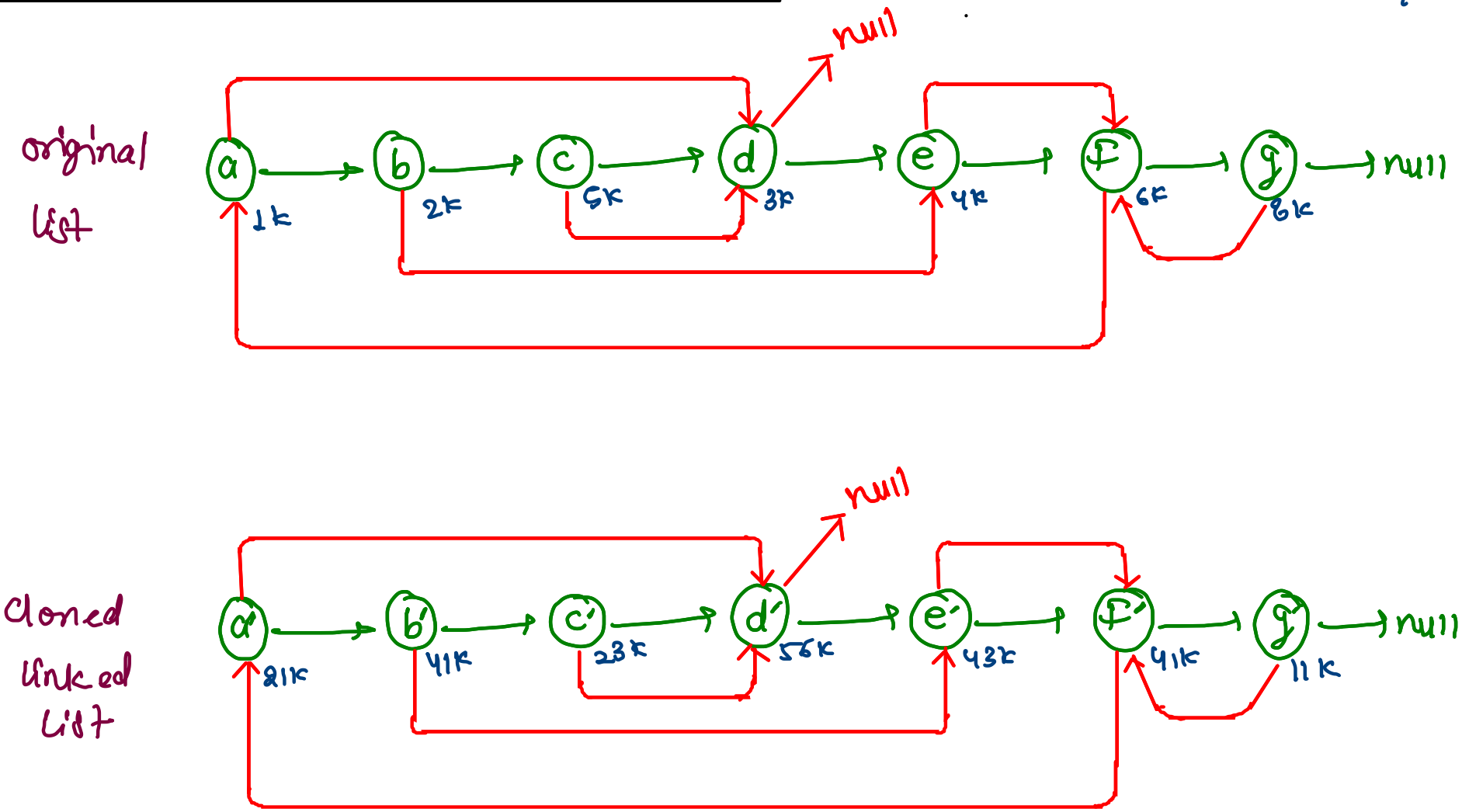
dist $= 2x \times T$

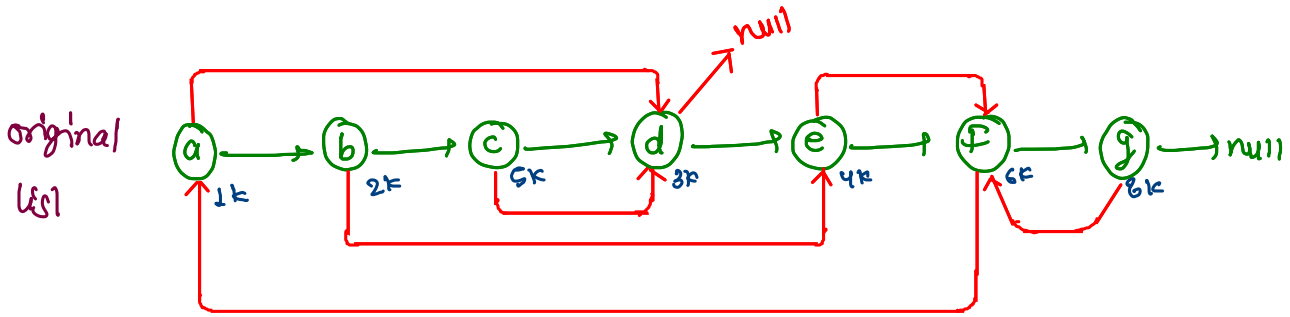
$$xT = \frac{\text{dist}}{2}$$

$$xT = \frac{a + 2n(b+c) + b}{2}$$

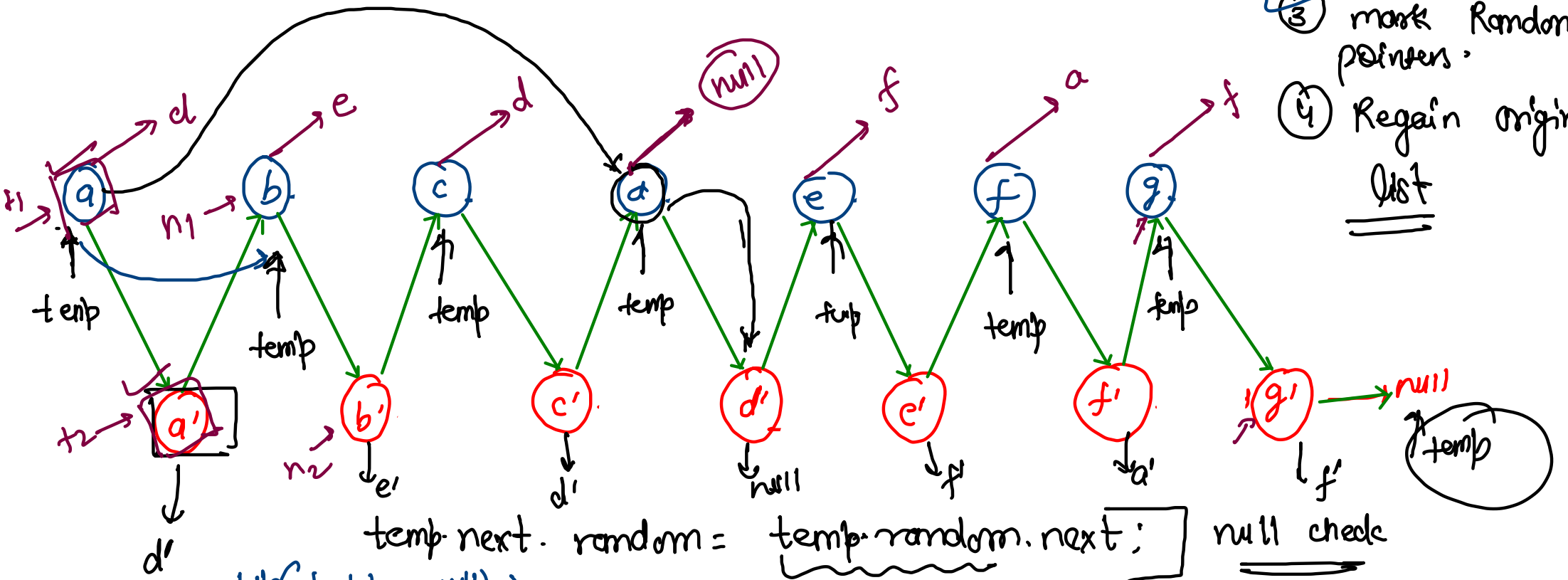
Copy Linked list with Random pointer:

Without HashMap

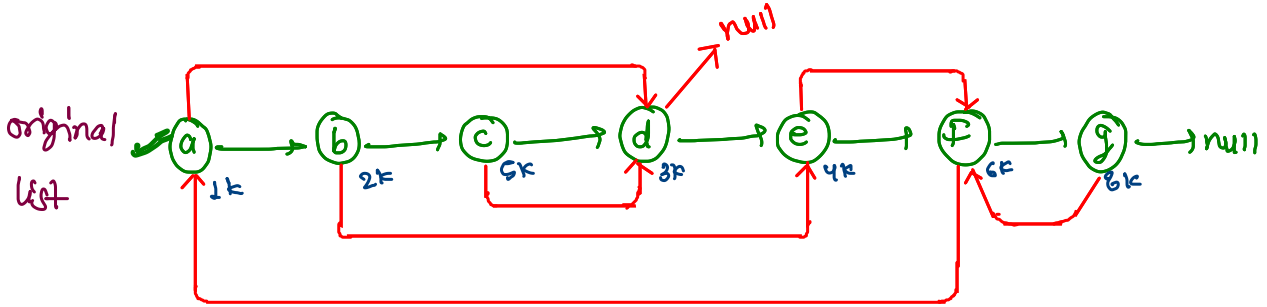




1. Make a copy linked list with next pointer only.
2. Make a zig-zag pointer.
3. mark Random pointers.
4. Regain original list



Copy Linked list with Random pointer:



1. Make copy of Linked list with next pointer only.

$a' \rightarrow b' \rightarrow c' \rightarrow d' \rightarrow e' \rightarrow f' \rightarrow g' \rightarrow null$

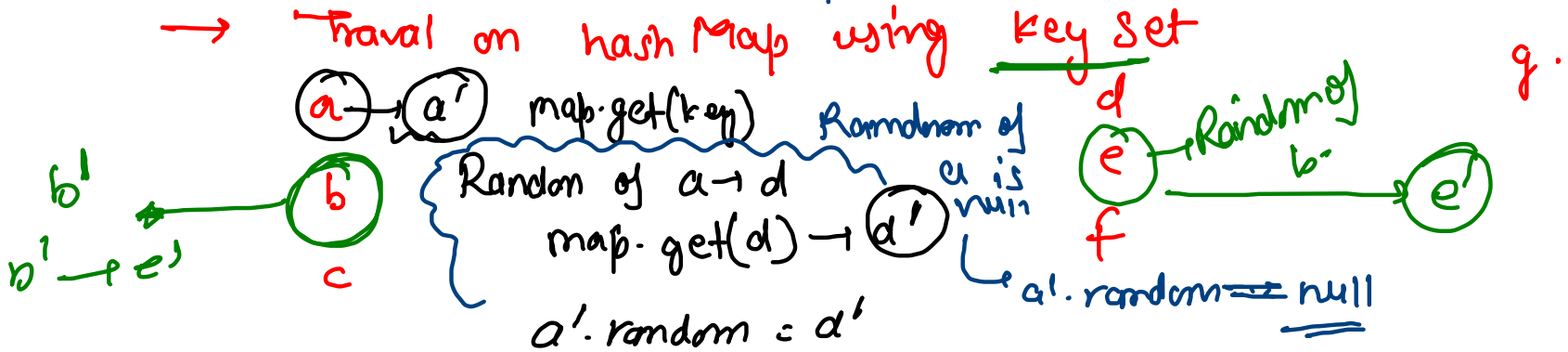
2. Make a hashMap of Node vs. Node and store original vs. copied Node.

HashMap<ListNode, ListNode> map. key = original node, value → copy node.

$a \rightarrow a'$	$c \rightarrow c'$	$e \rightarrow e'$	$g \rightarrow g'$
$b \rightarrow b'$	$d \rightarrow d'$	$f \rightarrow f'$	

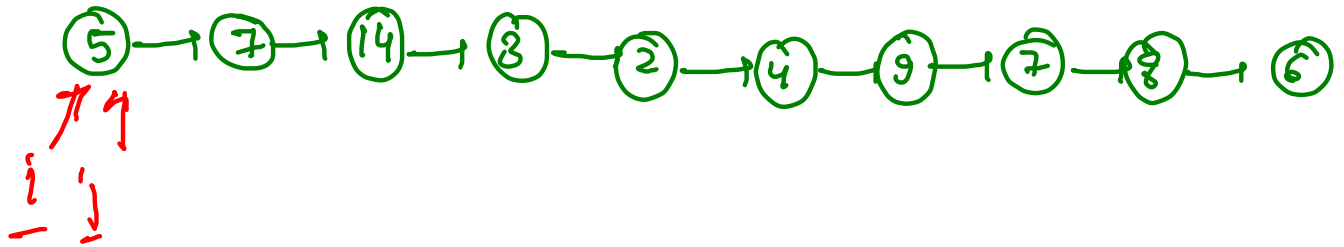
(3) Make connection of random point.

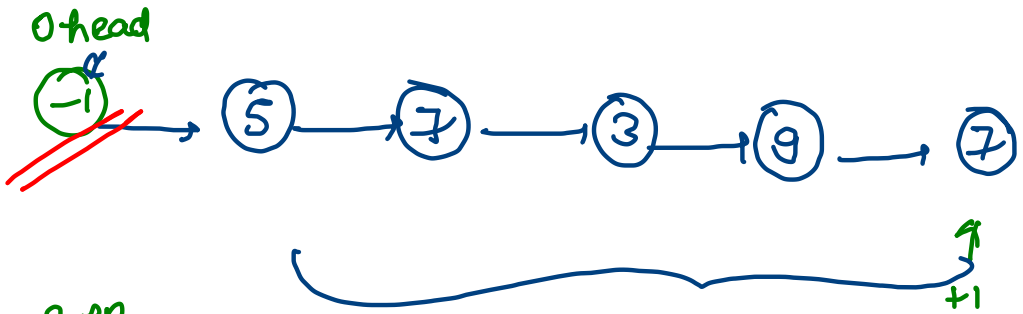
→ Traval on hash Map using key set



Segregate odd Even

(using pointer) In input odd and Even are randomly arranged →





this check is different for segregate 01

```
while(temp1 != null){
  if(temp->val % 2 == 0){
    t2->next = temp;
    t2 = temp;
  }else{
    t1->next = temp;
    t1 = temp;
  }
  temp = temp->next
}
t1->next = null, t2->next = null
```

add in order accordingly

odd → Even ,

Even → odd

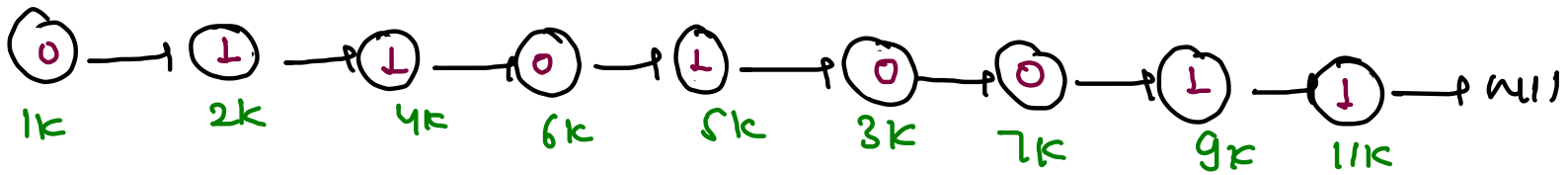
t2->next = Ohead->next

return Ehead->next

Segregate 0 1 (using ~~swapping~~ Node)

(change in pointer)

Input



O/P → Zero ⟶ one.