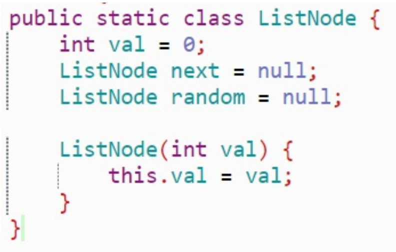
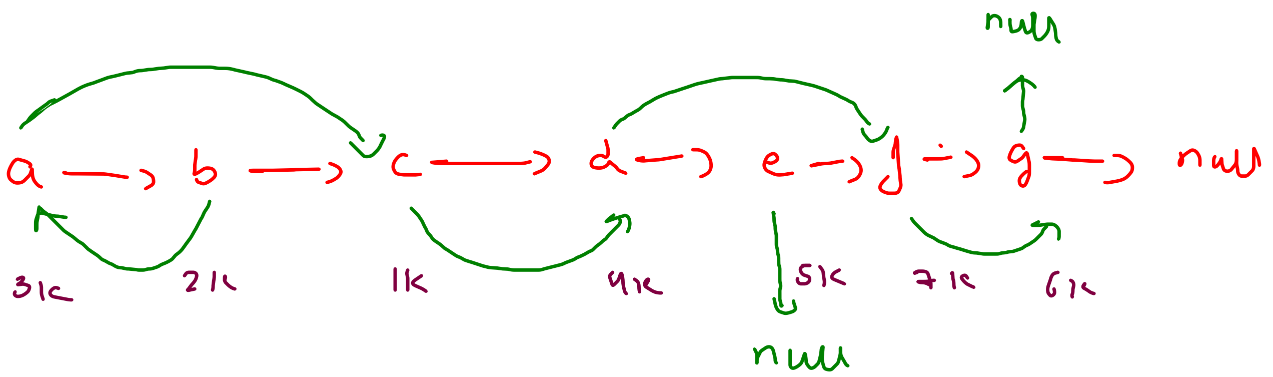
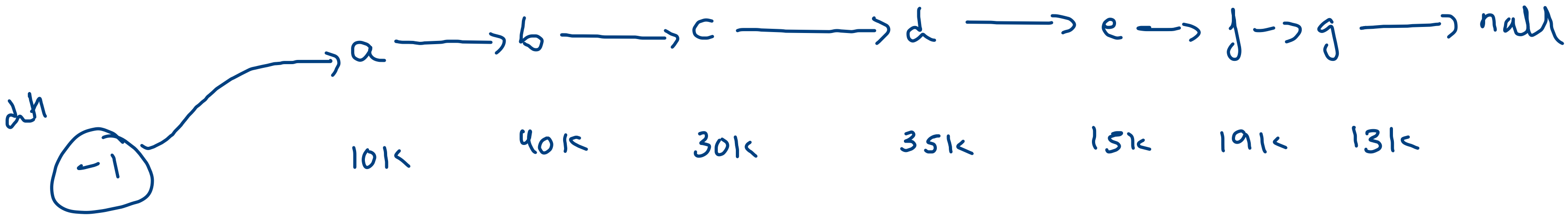


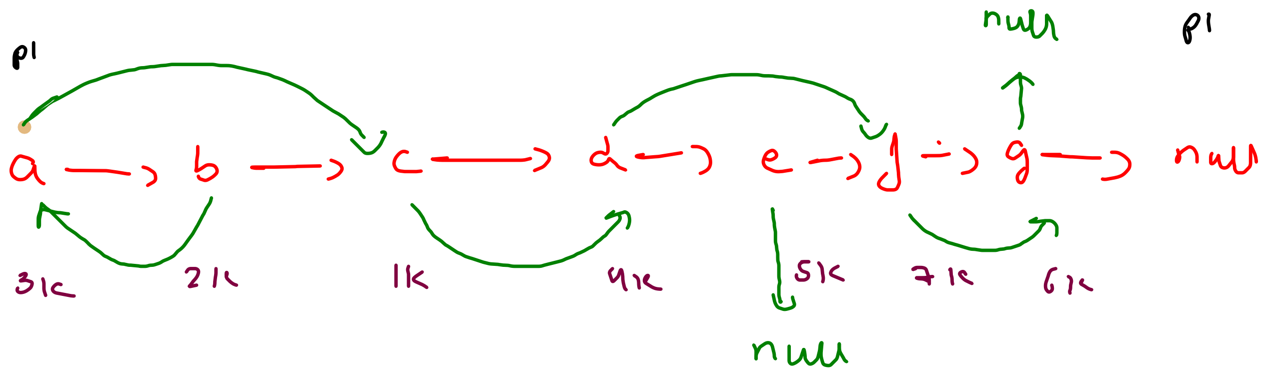
```
public static class ListNode {
    int val = 0;
    ListNode next = null;
    ListNode random = null;

    ListNode(int val) {
        this.val = val;
    }
}
```





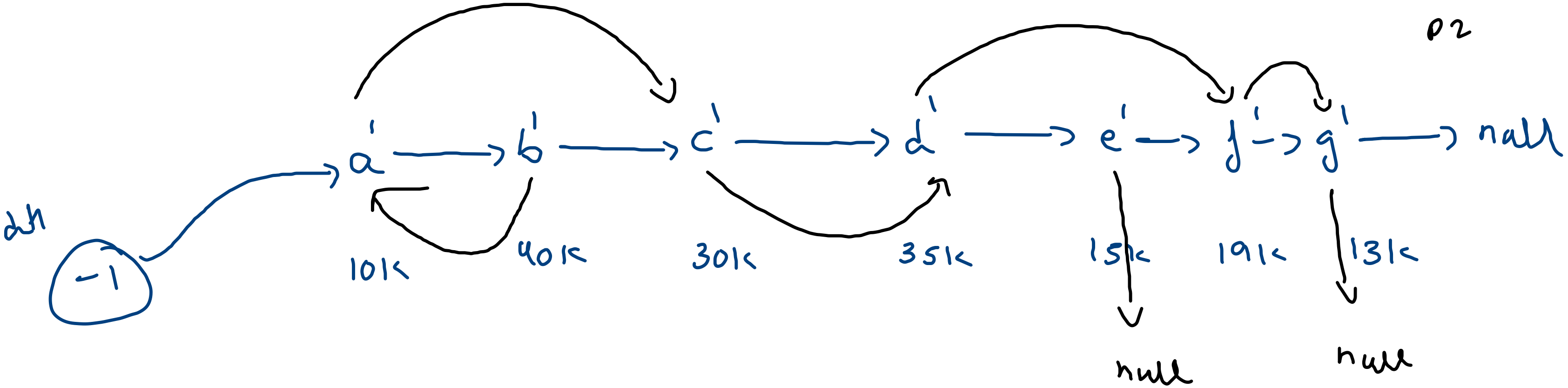
⟶ next

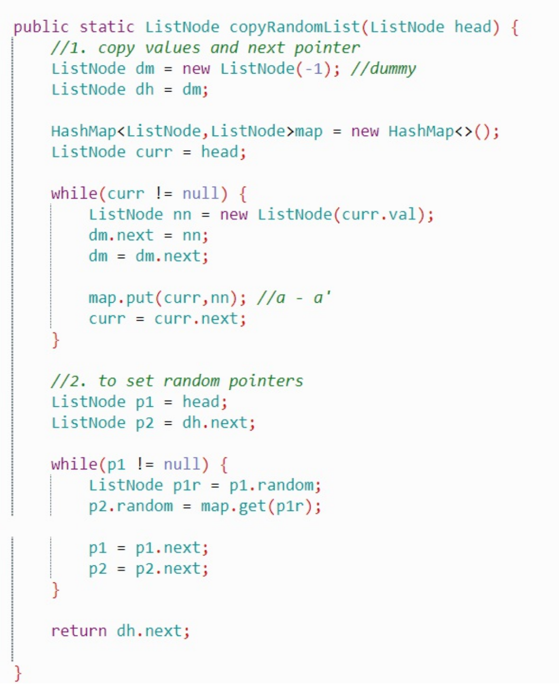
⟶ random



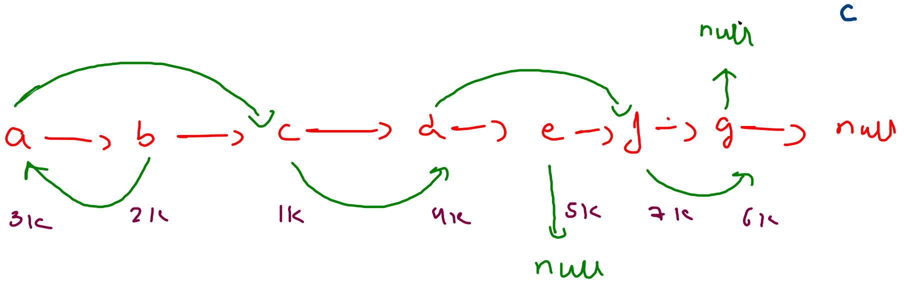
a - a'	3k - 10k
b - b'	2k - 40k
c - c'	1k - 30k
d - d'	4k - 35k
e - e'	5k - 15k
f - f'	7k - 19k
g - g'	6k - 13k

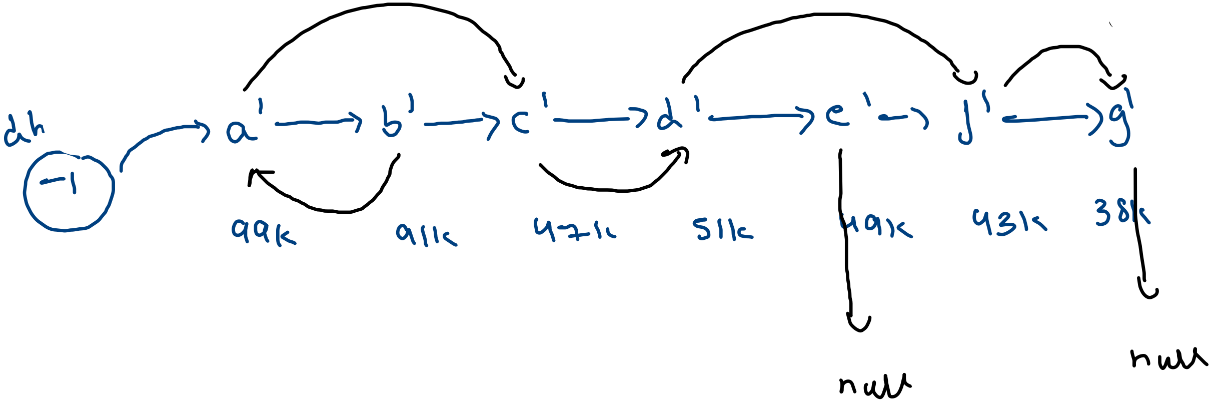
```
p1r = p1.random;
p2.random = map.get(p1r);
```





```
public static ListNode copyRandomList(ListNode head) {
    //1. copy values and next pointer
    ListNode dm = new ListNode(-1); //dummy
    ListNode dh = dm;

    HashMap<ListNode,ListNode>map = new HashMap<>();
    ListNode curr = head;

    while(curr != null) {
        ListNode nn = new ListNode(curr.val);
        dm.next = nn;
        dm = dm.next;

        map.put(curr,nn); //a - a'
        curr = curr.next;
    }

    //2. to set random pointers
    ListNode p1 = head;
    ListNode p2 = dh.next;

    while(p1 != null) {
        ListNode p1r = p1.random;
        p2.random = map.get(p1r);

        p1 = p1.next;
        p2 = p2.next;
    }

    return dh.next;

}
```





3k —> 99k

2k -> 91k

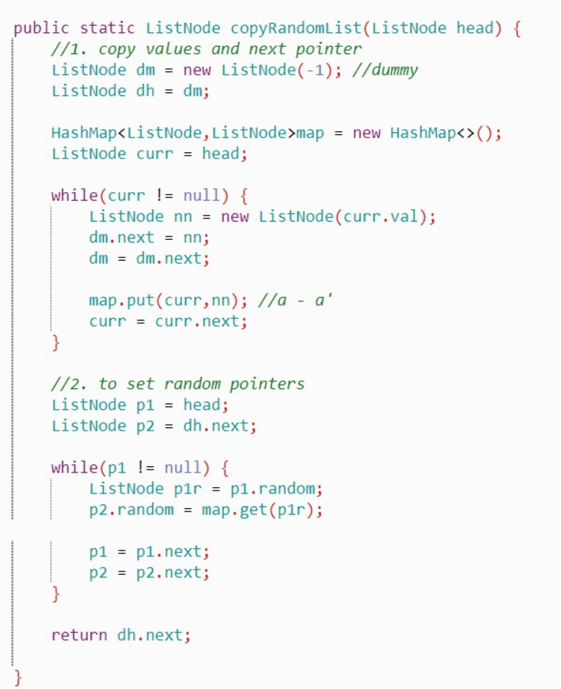
1k -> 47k

4k -> 51k

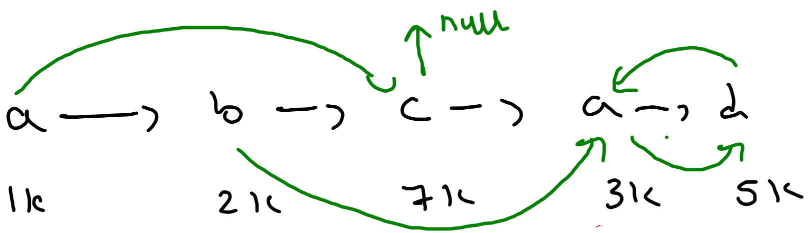
5k -> 49k

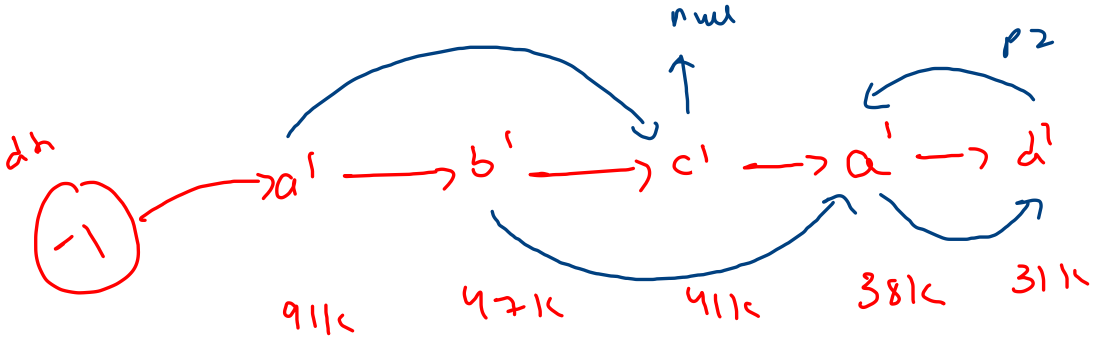
7k -> 43

6k -> 38



```
public static ListNode copyRandomList(ListNode head) {
    //1. copy values and next pointer
    ListNode dm = new ListNode(-1); //dummy
    ListNode dh = dm;

    HashMap<ListNode,ListNode>map = new HashMap<>();
    ListNode curr = head;

    while(curr != null) {
        ListNode nn = new ListNode(curr.val);
        dm.next = nn;
        dm = dm.next;

        map.put(curr,nn); //a - a'
        curr = curr.next;
    }

    //2. to set random pointers
    ListNode p1 = head;
    ListNode p2 = dh.next;

    while(p1 != null) {
        ListNode p1r = p1.random;
        p2.random = map.get(p1r);

        p1 = p1.next;
        p2 = p2.next;
    }

    return dh.next;

}
```





1k->91k

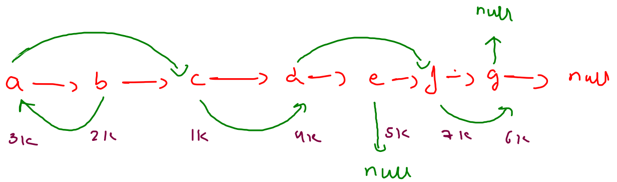
2k->47k

7k->41k

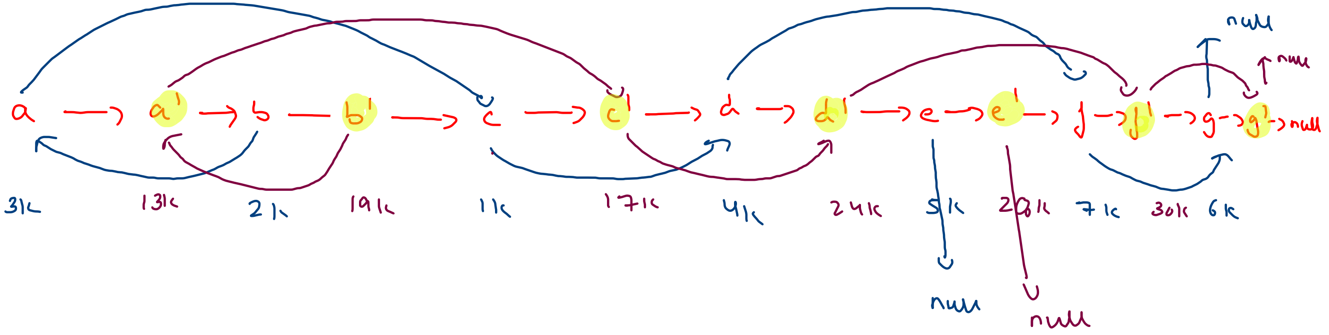
3k->38k

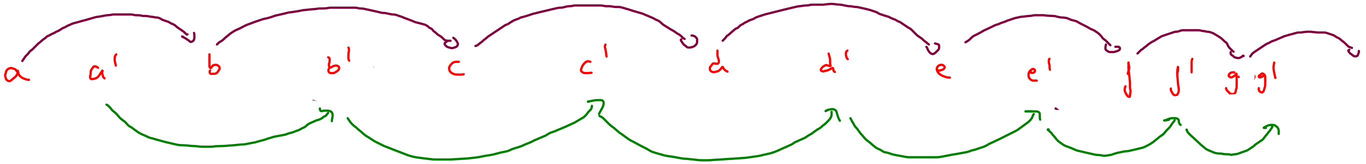
5k->31k

without space



c2.random = c1.random.next

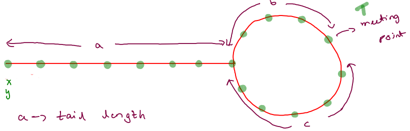




×

c1 c2 n1 n2

```
c1.next = n1;
c2.next = n2;
c1 = n1;
c2 = n2;
```



a → tail length

cycle distance → b + c

m → no. of rounds taken by x before meeting.

n → no. of rounds take by y before meeting.

(i) cycle detection

(ii) cycle start point

(iii) cycle break

$x = d_x / T$ — ①

$y = d_y / T$ — ②

$$\frac{d_x}{x} = \frac{d_y}{y}$$

$d_x = a + m(b+c) + b$

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

$y\,d_x = x\,d_y$

$\frac{y}{x} = r$

$\frac{y}{x} d_x = d_y$

$r\,d_x = d_y$

$ar + mr(b+c) + br = a + n(b+c) + b$

$a(r-1) + b(r-1) = (b+c)(n-mr)$

$(r-1)(a+b) = (b+c)(n-mr)$

$$\boxed{a+b = \frac{(b+c)(n-mr)}{r-1}}$$

$x = 3$
$y = 5$

$r > 1$

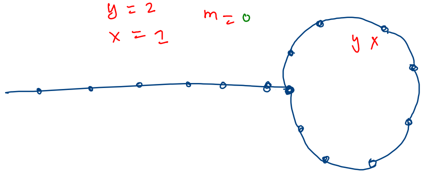
$\frac{y}{x} > 1$

optimise

$\boxed{r = 2}$

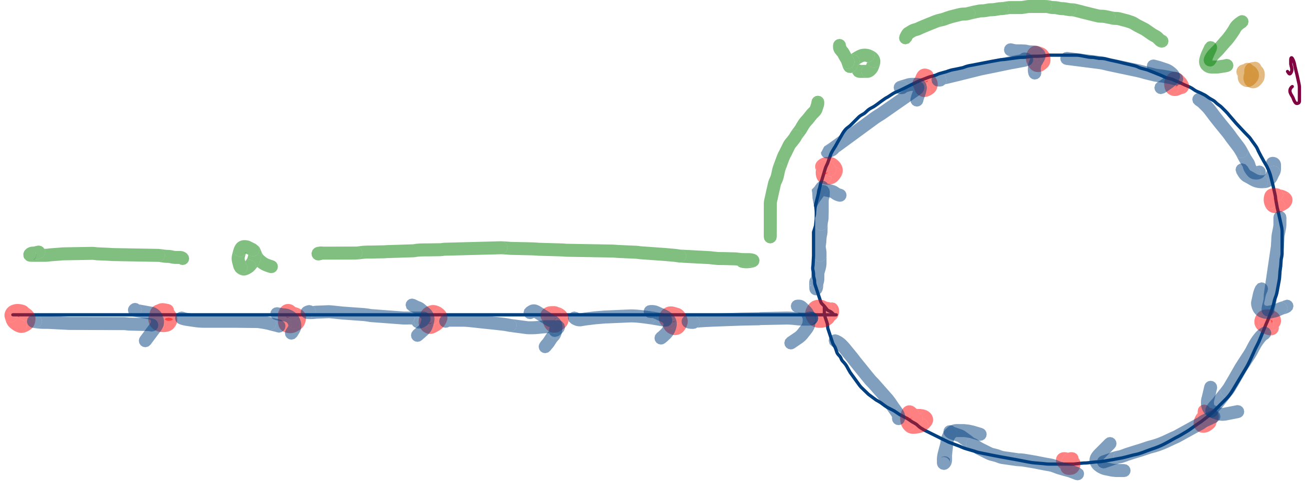
$y = 2$
$x = 1$

$m = 0$



$\frac{y}{x} = 2$

$\boxed{y = 2x}$



$n = 1$

$r = 2$

$m = 0$

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

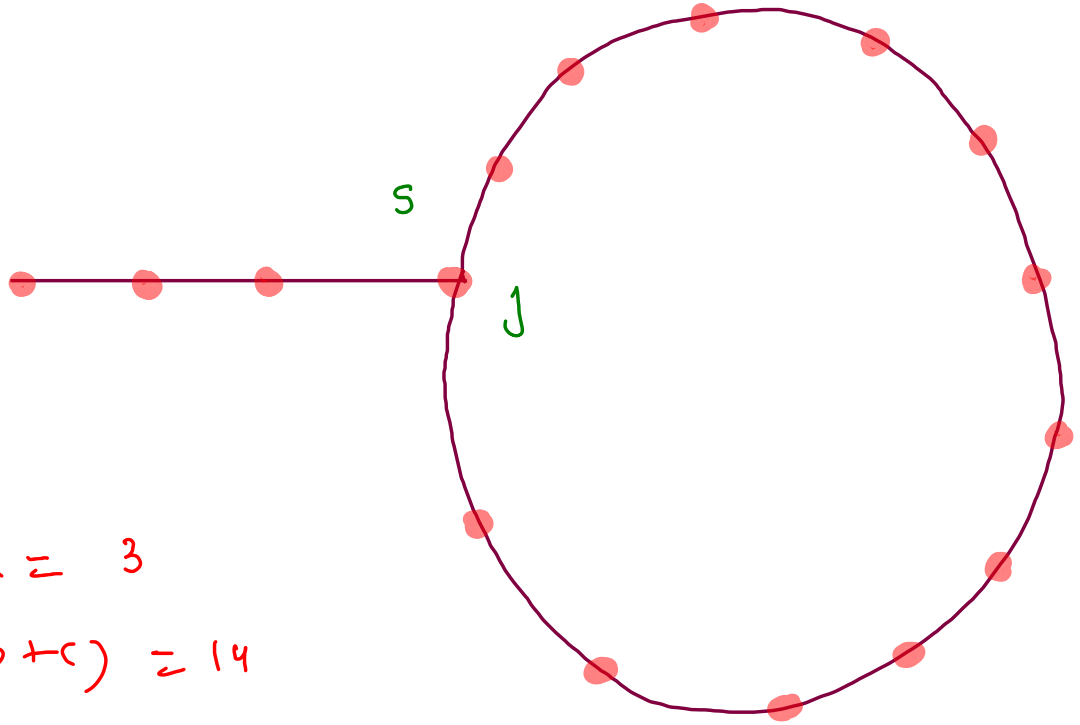
$$= (b+c)n - (b+c) + c$$

$$\boxed{a = (b+c)(n-1) + c}$$

$$(a+b) = \frac{(b+c)(n-mr)}{(r-1)}$$

$$\boxed{a+b = (b+c)(n)}$$

$$a + \cancel{b} = \cancel{b} + c$$



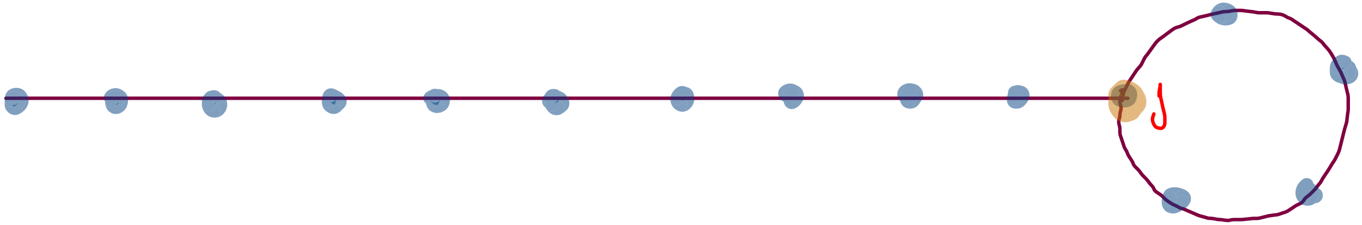
$a = 3$

$(b + c) = 14$

$b = 11$

$n = 1$

$n = 2$



$a = 10$

$b+c = 5$

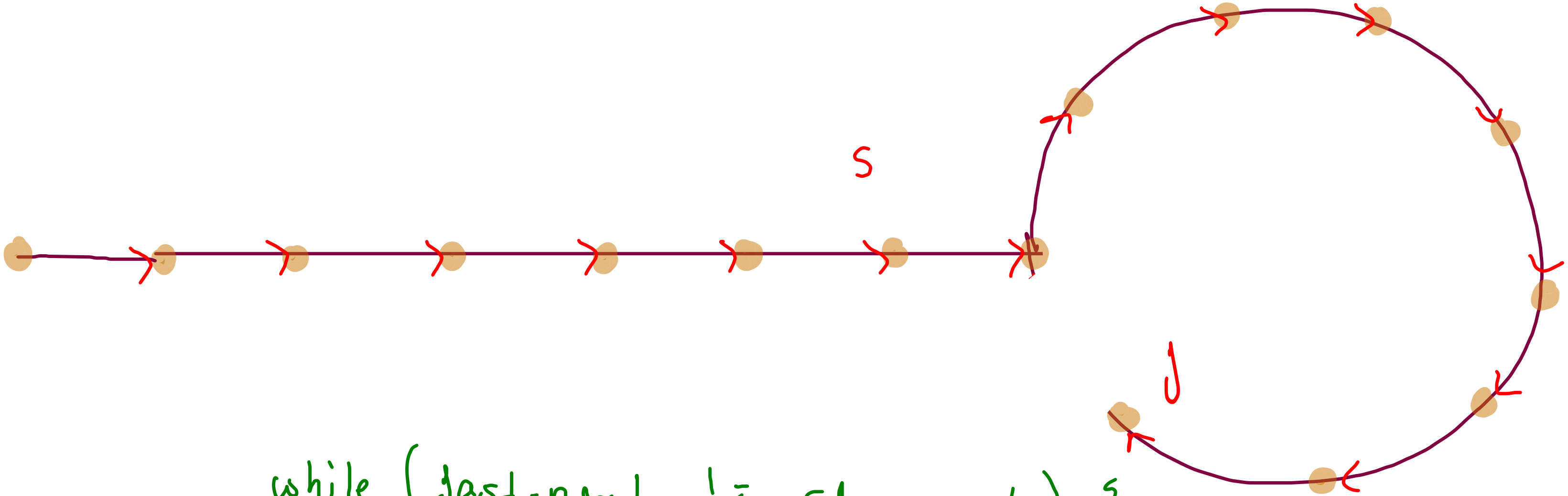
$b = 0$

$c = 5$

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

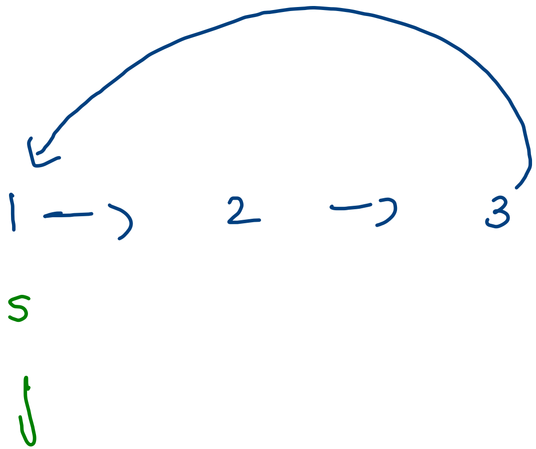
$a = (b+c)n-b$

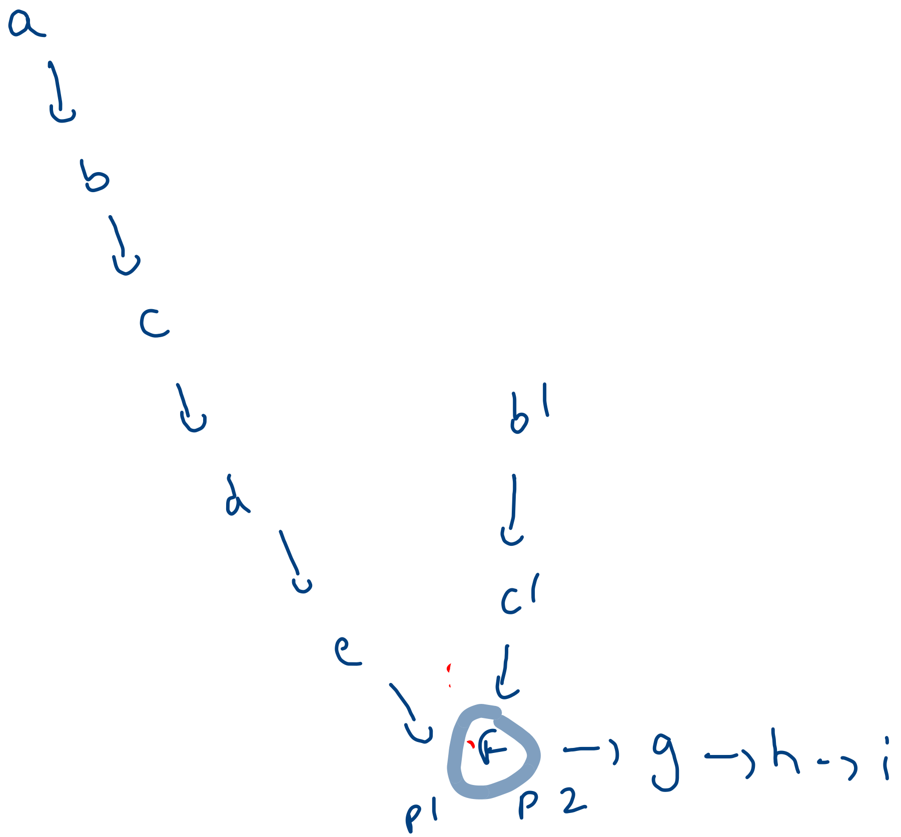
$$\boxed{a = (b+c)(n-1) + c}$$



```
while (fast.next != slow.next) {
        slow = slow.next;
        fast = fast.next;          cycle    break.
}
fast.next = null.
```

Cycle break





s1 = 9

s2 = 6

gap = 3