1 → 2 → 3 → 2 → 1 → N. ✓

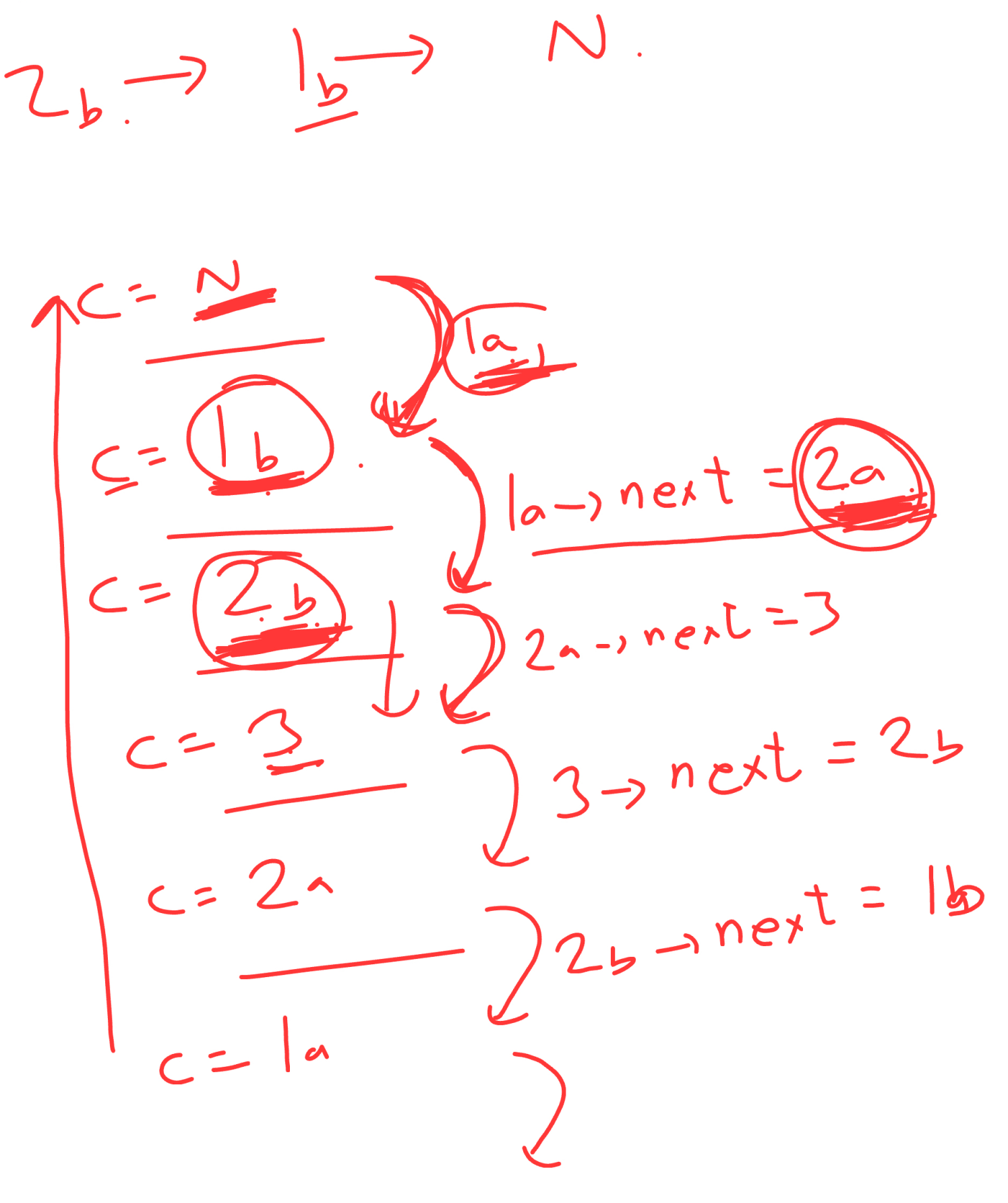
1 → 2 → 3 → 1 → N. ×

1 → 2 → 3 → 2 → 1 → N.

1 → 2 → 3 → 2 → 1 → N

TC = O(n)
AS = O(n).

$1_a \to 2_a \to 3 \to 2_b \to 1_b \to N.$


c = N
c = 1b
c = 2b
c = 3
c = 2a
c = 1a
1a
1a → next = 2a
2a → next = 3
3 → next = 2b
2b → next = 1b

TC = O(n)
AS = O(n)

1 → 2 → 3 → 3 → 2 → 1 → N.

M1
1 → 2 → 3 → N.

M2
3 → 2 → 1 → N. ⇒ M2. 1 → 2 → 3 → N.

1 → 2 → 3 → 2 → 1 → N.

M1
1 → 2 → 3 → N.

M2
2 → 1 → N. ⇒ M2 1 → 2 → N.

TC = O(n).
AS = O(1).

mid.

$1_a \to 2_a \to 3_a \to 3_b \to 2_b \to 1_b \to N.$

M1
1a → 2a → 3a → N.

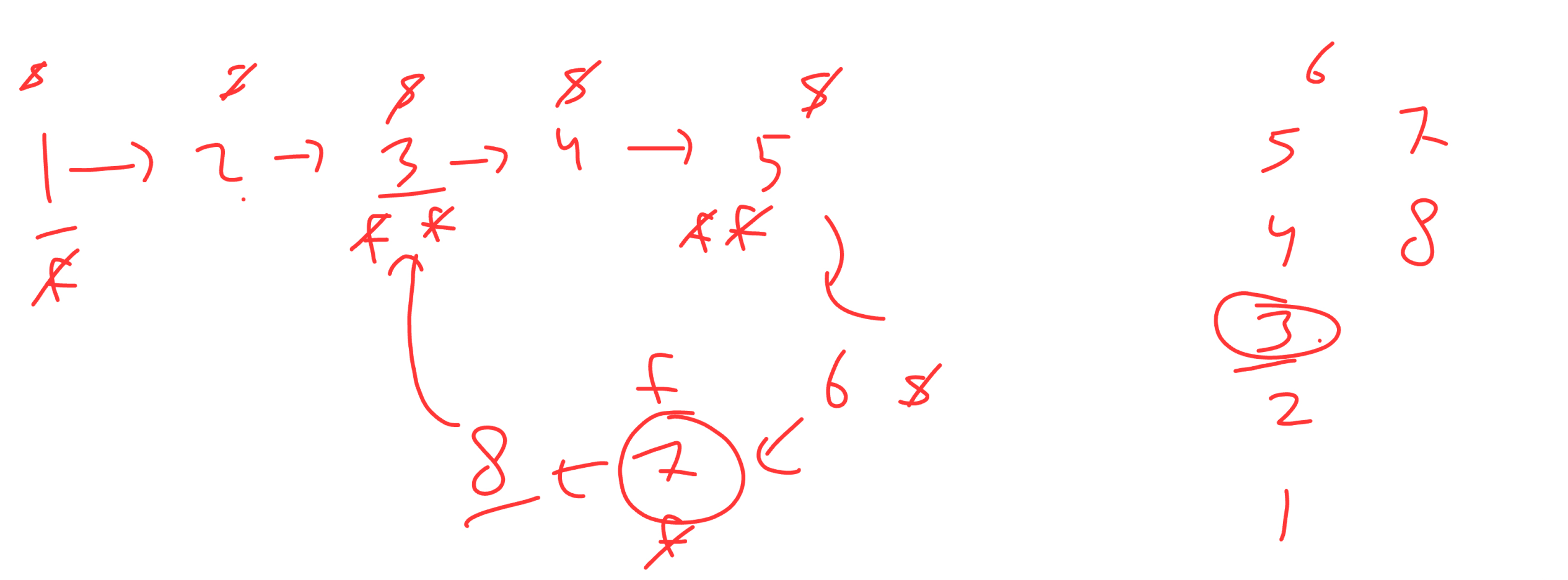
M2.
3b → 2b → 1b → N.

M1 = H.
M2 = mid → next
mid → next = N.

M. M2.
H' 1 → N.
mid.

H1
1 → N.

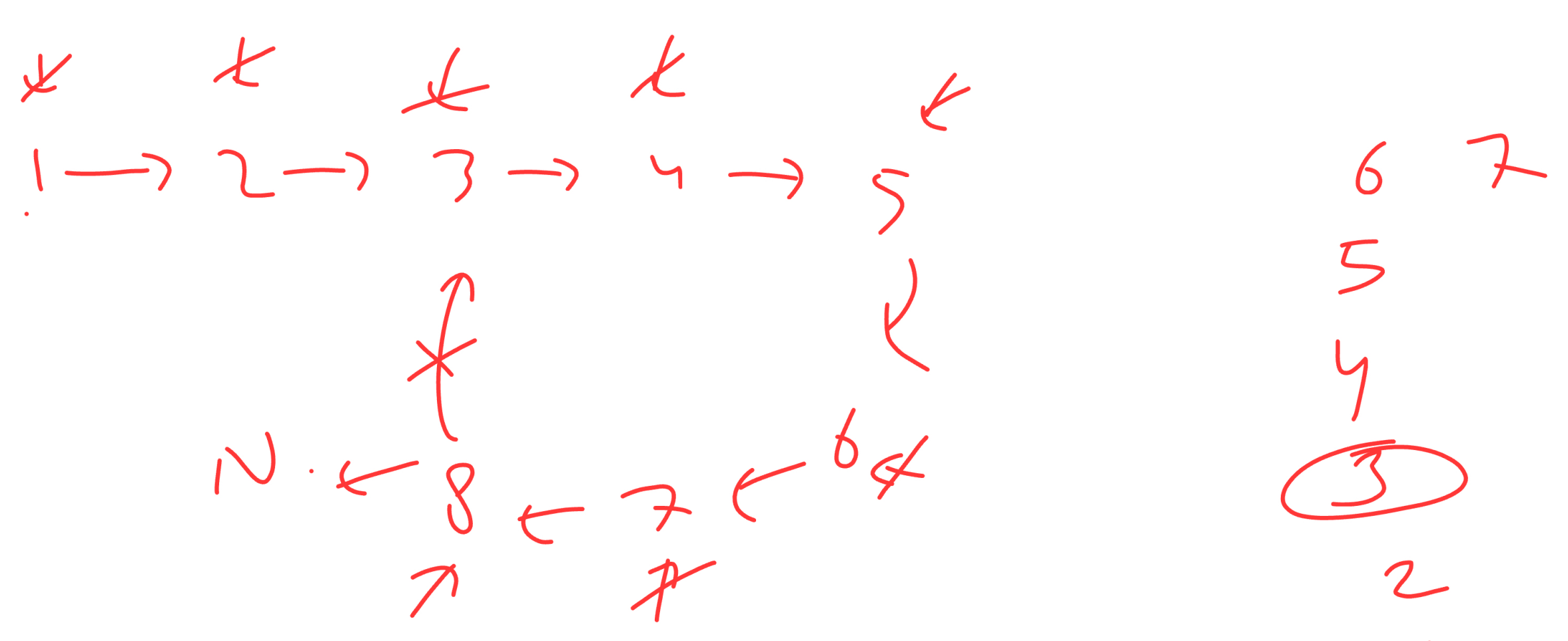
M2
N.


1 → 2 → 3 → 4 → 5
6
7
8
f
s
6 7
5 8
4
3
2
1

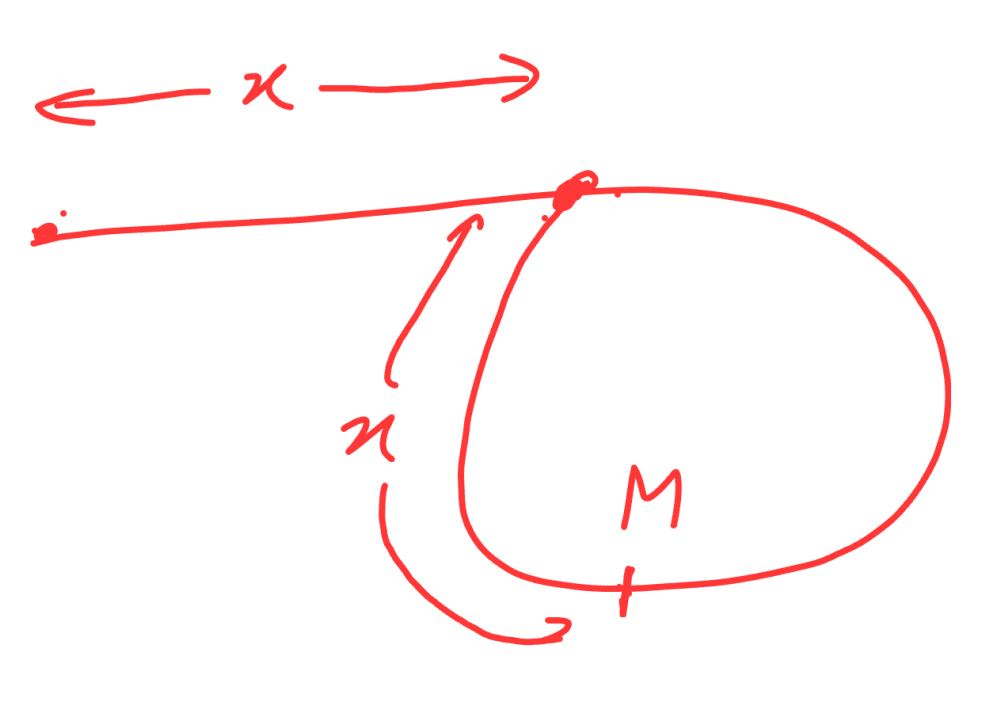
TC = O(n)
AS = O(n)

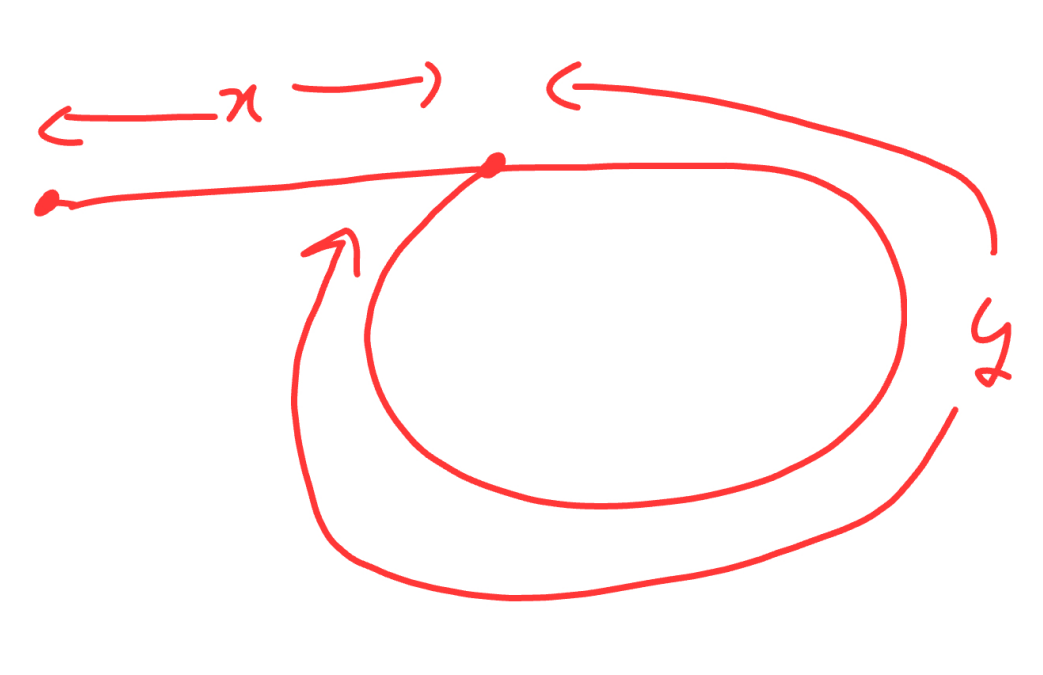
t: s: x
f: x + 1

t-1: s: x - 1
f: x + 1 - 2 = x - 1

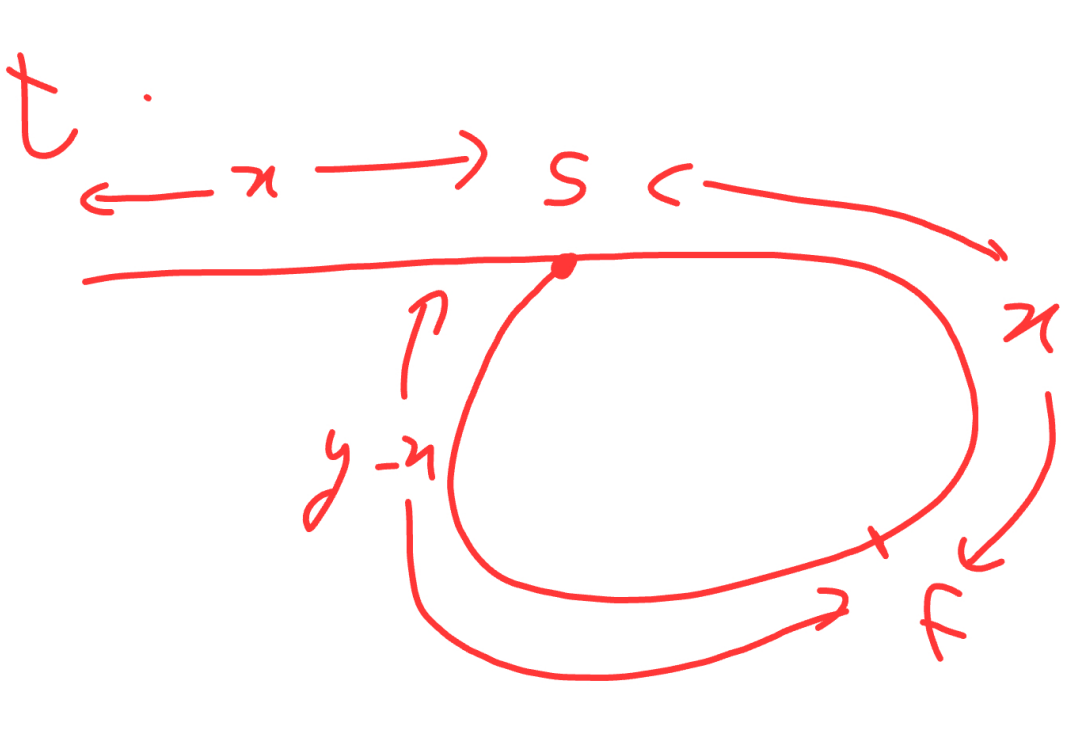

1 → 2 → 3 → 4 → 5
6
7
8
N.
6 7
5
4
3
2
1

TC = O(n)
AS = O(n)





At time t



slow is x steps behind fast.
fast is (y - x) steps behind slow.

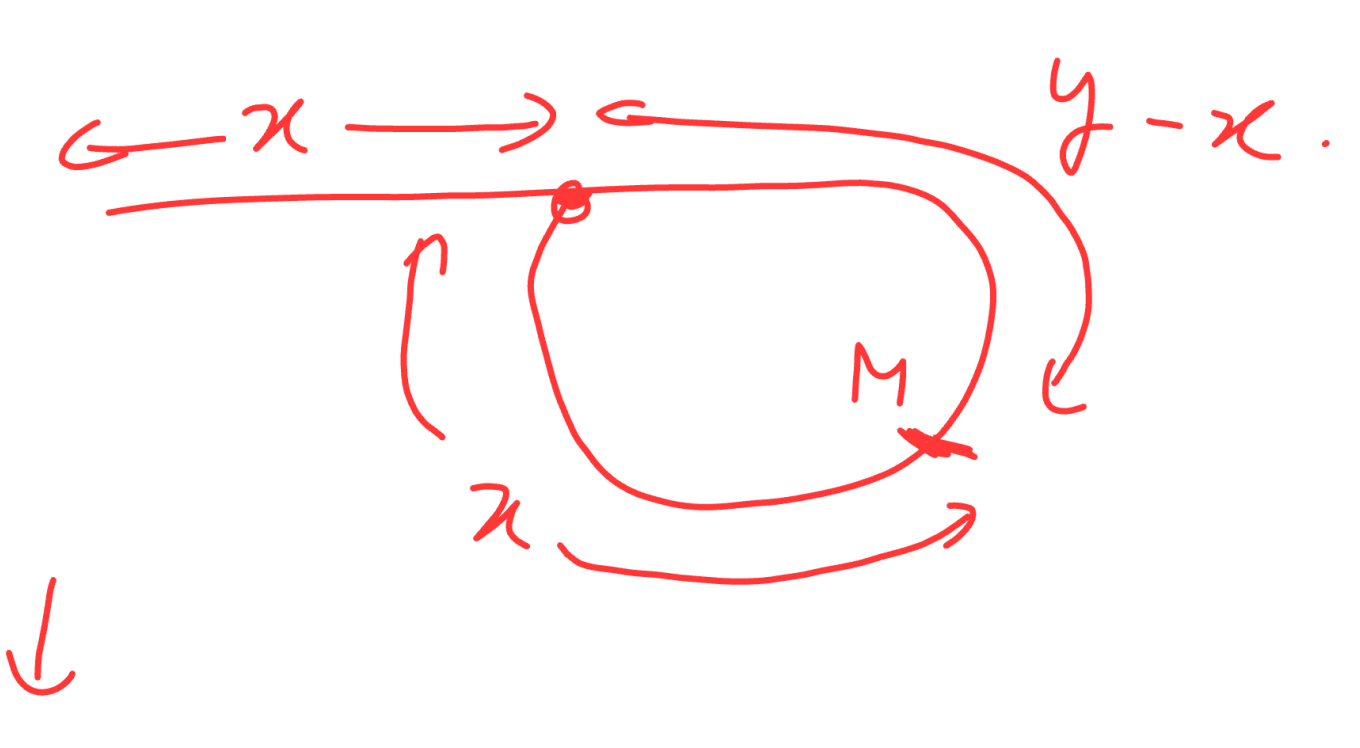
fast catches up slow (2 - 1) steps per iteration.

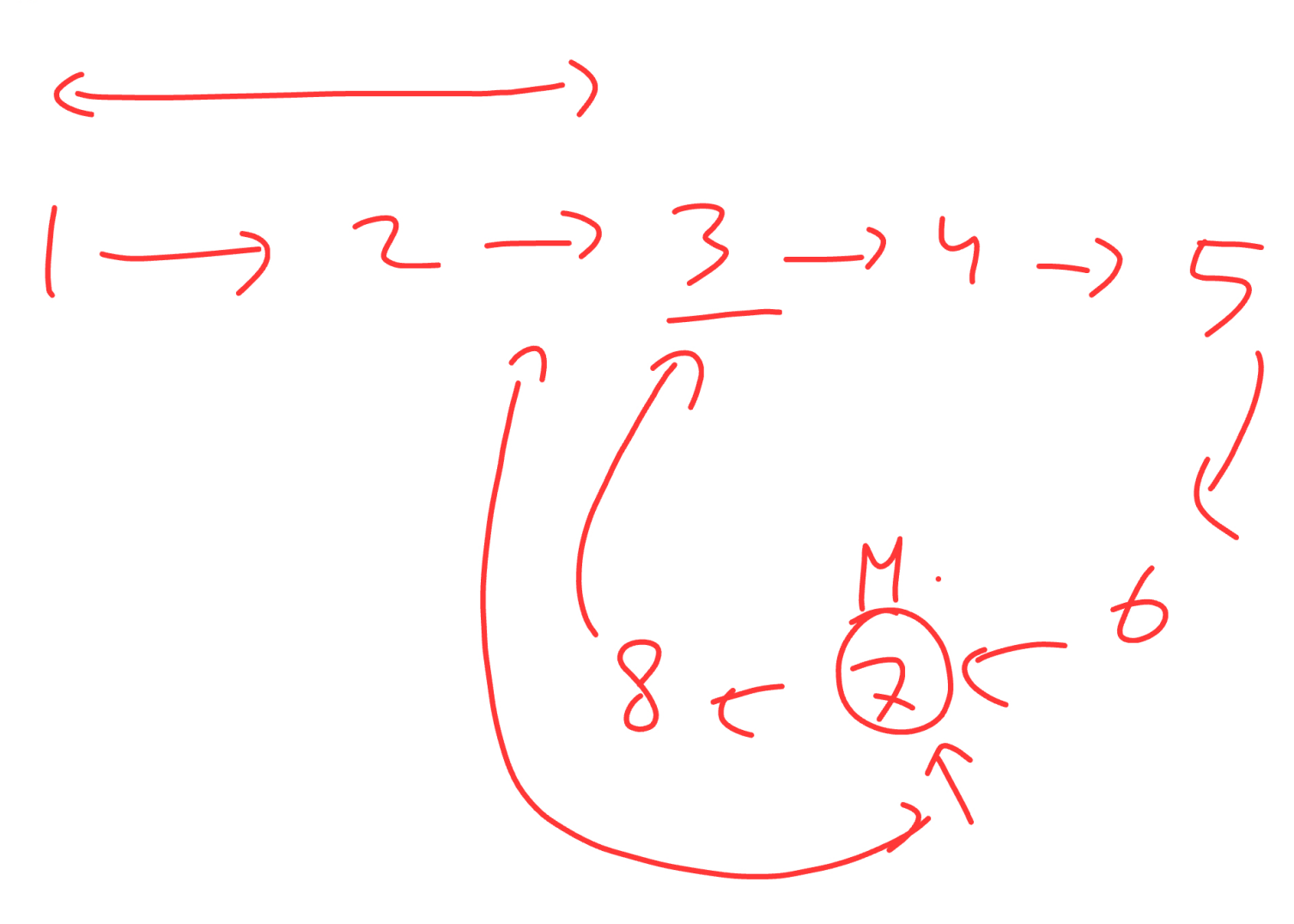
$s = \frac{d}{t}$

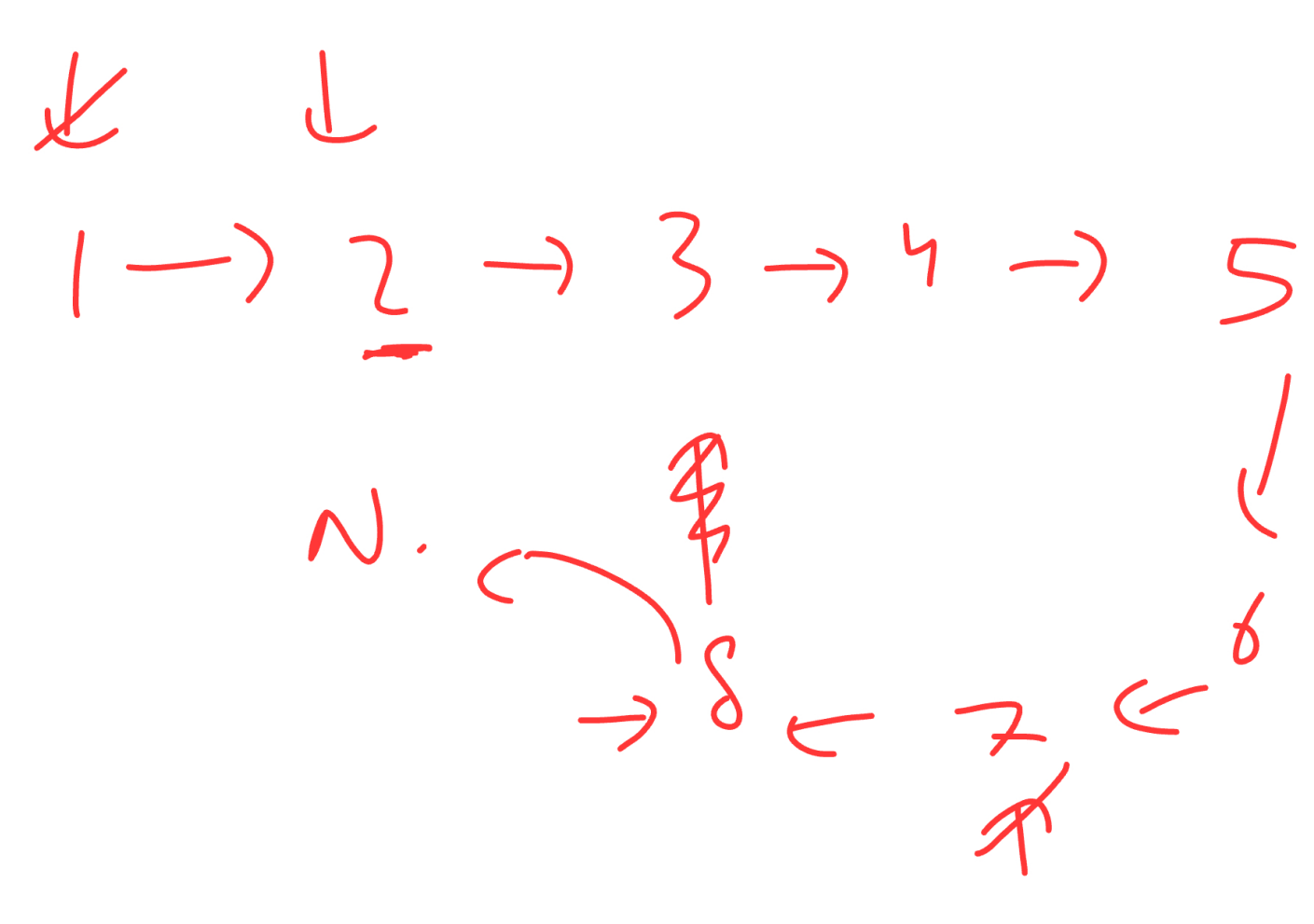
Time that fast will take to catch slow = $\frac{y-x}{2-1}$ = (y - x) iterations

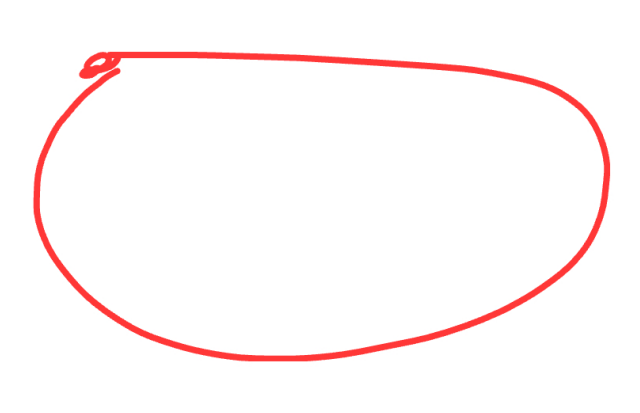
$t = \frac{d}{s}$

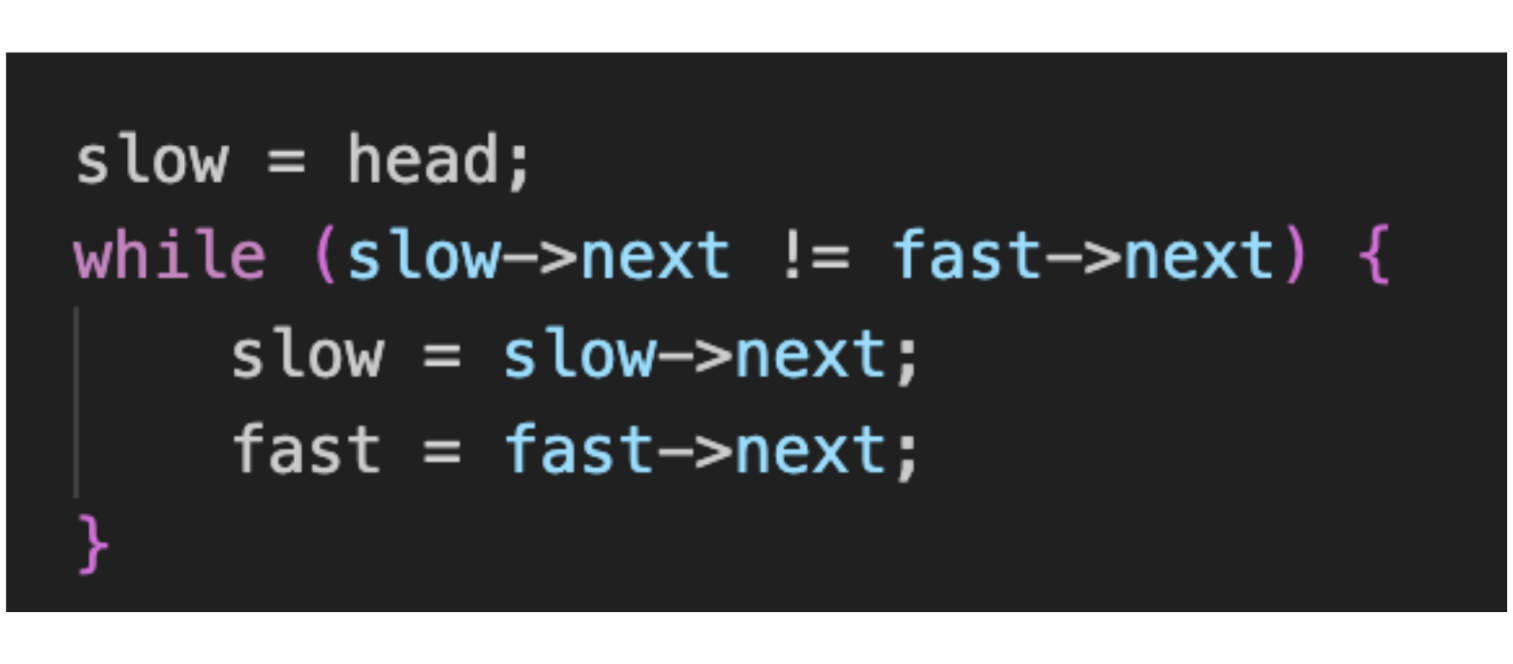
At time t + (y - x).



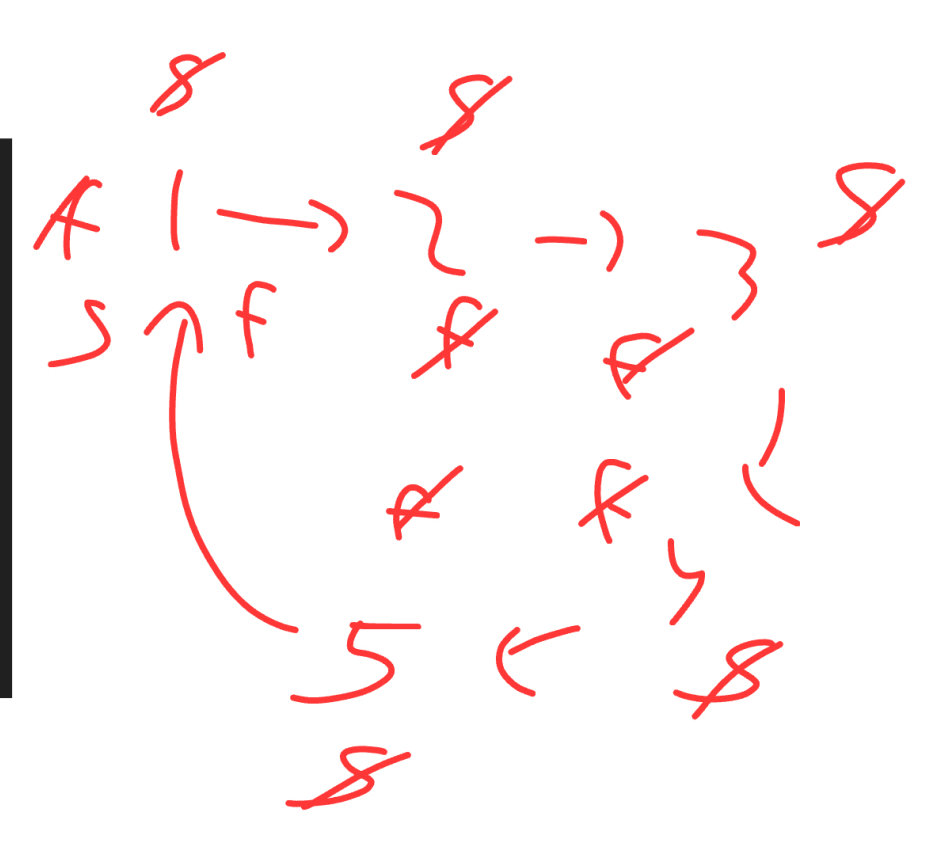


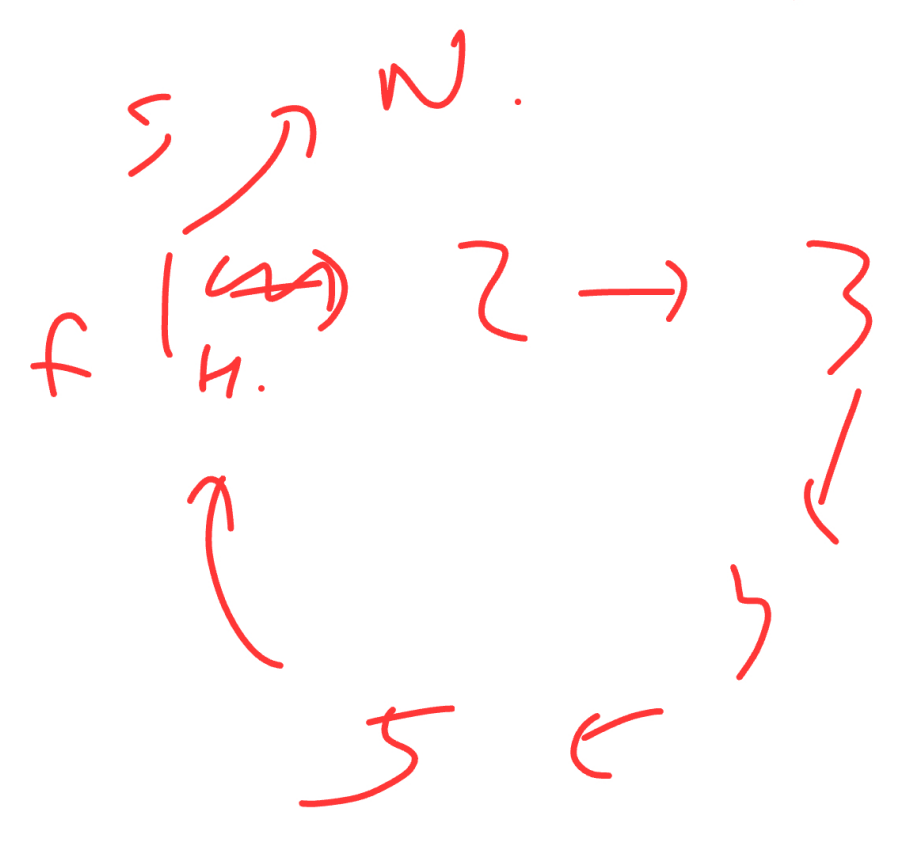






```
slow = head;
while (slow->next != fast->next) {
    slow = slow->next;
    fast = fast->next;
}
```





1 → N.

---

1 → 1 → 2 → 4 → 6 → N.

1 → 3 → 6 → 10 → N.

o/p: 1 → 1 → 1 → 2 → 3 → 4 → 6 → 6 → 10 → N.

1 → 1 → 1 → 2 → 3 → 4 → 6 → 6 → 10 → N.

TC = O(n+m)
AS = O(1).