1 → 2 → 3 → 4 → NULL

4/2 = 2

5/2 = 2

~~≠~~ 1 → 2 → 3 → 4 → 5 → NULL

n/2 − 1 n/2

TC = O(n)

AS = O(1)

1 → 2 → 3 → 4 → 5 → NULL

Runner Technique

2 pointers → slow speed = 1 hop / iteration

→ fast speed = 2 hop/ iteration

Odd.

1 → 2 → 3 → 4 → 5 → NULL

s f

fast→next == NULL ①

Even.

1 → 2 → 3 → 4 → NULL

fast == NULL ②

1 → 2 → 3 → NULL

1 → 2 → 3 → 4 → 5 → 6 → N

---

1 → 2 → 3 → 2 → 1 → NULL True

1 → 2 → 3 → 1 → NULL False

1 → NULL True

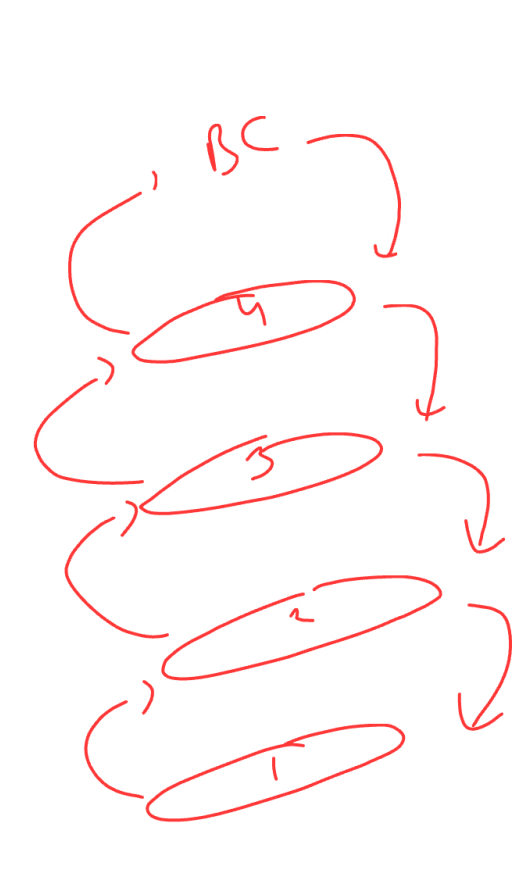
Duplicate reversed linked list : -based approach

TC : O(n)

AS : O(n)

$1_a \to 2_a \to 3 \to 2_b \to 1_b \to$ NULL

rev.



(return head)

NULL) 1_b (return 1_a → next)

1_b) 2_a (return 2_a → next)

2_b) 3 (return 3 → next)

3) 2_b (return 2_b → next)

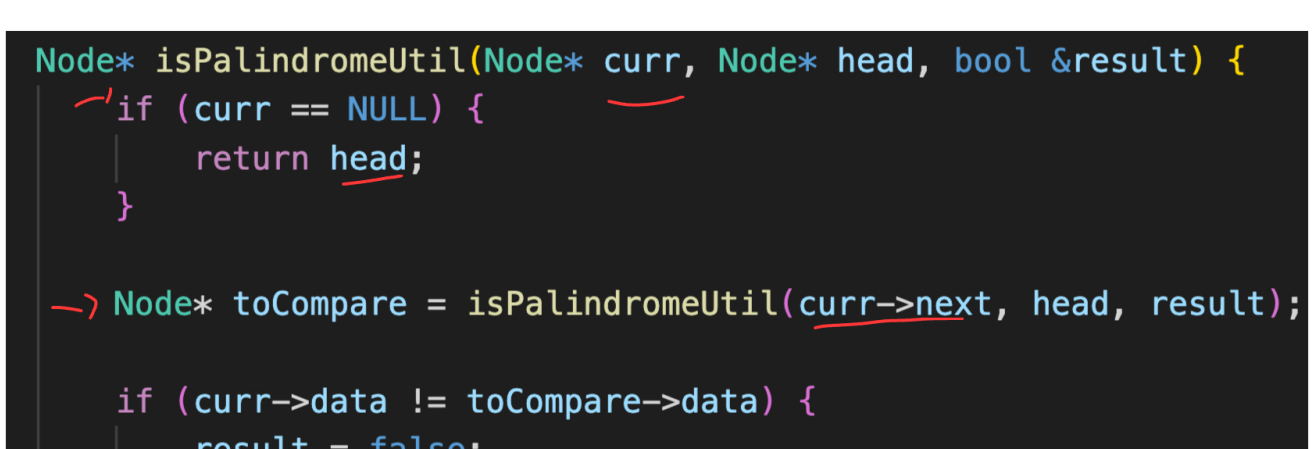
2_a) 1_b (return 1_b → next)

1_a

Orig.

TC: O(n)

AS: O(n)



```
Node* isPalindromeUtil(Node* curr, Node* head, bool &result) {
    if (curr == NULL) {
        return head;
    }

    Node* toCompare = isPalindromeUtil(curr->next, head, result);

    if (curr->data != toCompare->data) {
        result = false;
    }

    return toCompare->next;
}
```

curr = NULL) toC = 1_a

curr = 1_b) toC = 2_a

curr = 2_b) toC = 3

curr = 3) toC = 2_b

curr = 2_a) toC = 1_b

curr = 1_a) NULL

$1_a \to 2_a \to 3 \to 2_b \to 1_b \to$ NULL

1 → 2 → 3 ⁞ → 3 → 2 → 1 → NULL

1 → 2 → 3 ⁞ → 2 → 1 → NULL

A palindrome mirrors around its centre.

①

M1 NULL

1 → 2 → 3 → 3 → 2 → 1 → NULL

M2

reverse

1 → 2 → 3 → NULL

①

M2

1 → 2 → 3 → 2 → 1

1 → 2 → 4 → 2 → 1

1 → 2 → 5 → 2 → 1

M1 ① NULL

1 → 2 → ③ → 2 → 1 → NULL

M2

reverse

M2

1 → 2 → NULL

①

Head

1 → 2 → 3 → 2 → 1 → NULL

mid M2

Head

1 → 2 → ③ → 3 → 2 → 1 → NULL

Head

1 → 2 → 3_a → NULL 3_b → 2 → 1 → NULL

M2 = mid → next

mid → next = NULL

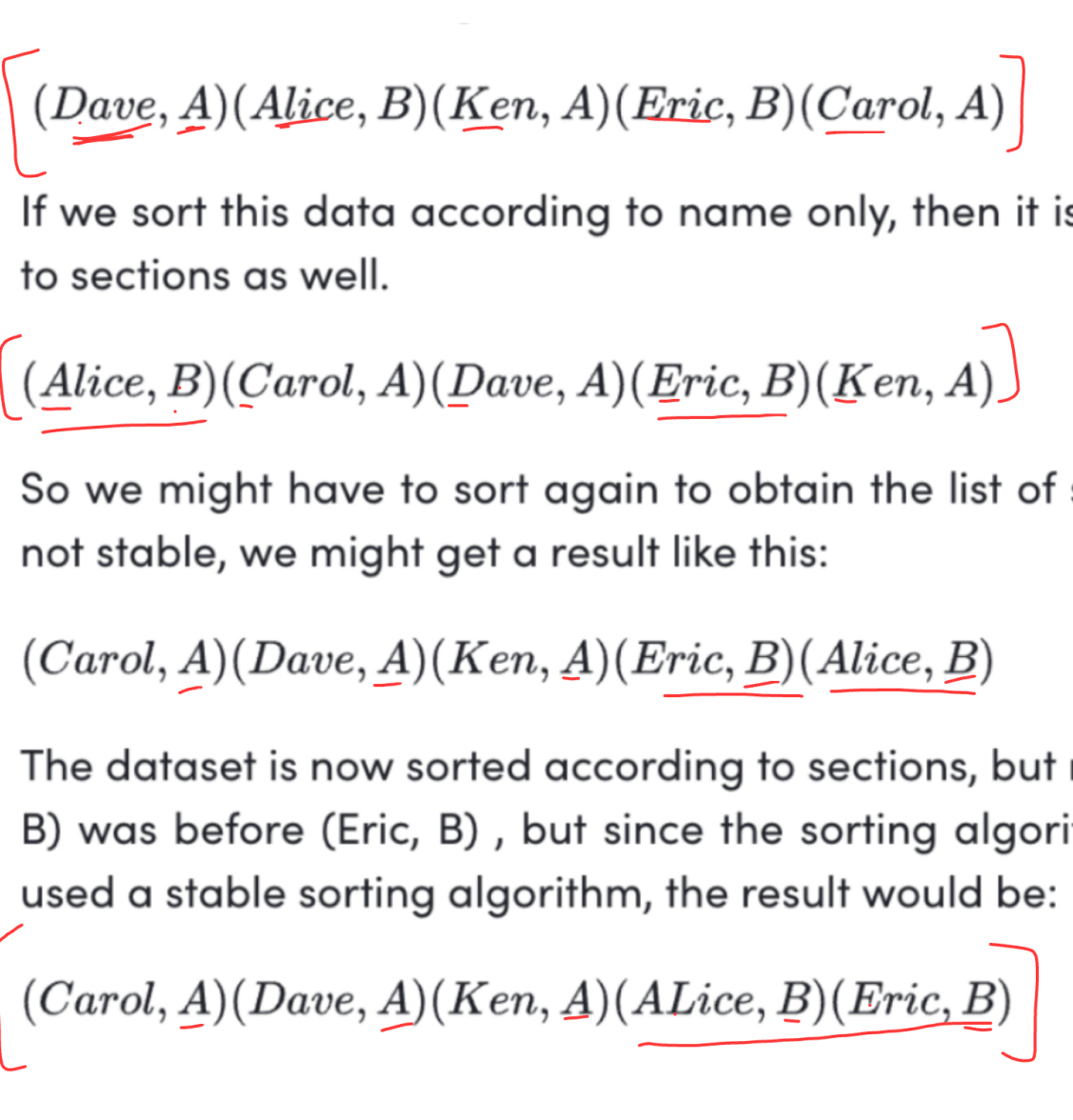
TC: $O\left(n + \frac{n}{2} + \frac{n}{2}\right) = O(n)$.

AS: O(1)

I/P

sort based on name

sort on section using stable sort



(Dave, A)(Alice, B)(Ken, A)(Eric, B)(Carol, A)

If we sort this data according to name only, then it is h
to sections as well.

(Alice, B)(Carol, A)(Dave, A)(Eric, B)(Ken, A)

So we might have to sort again to obtain the list of st
not stable, we might get a result like this:

(Carol, A)(Dave, A)(Ken, A)(Eric, B)(Alice, B)

The dataset is now sorted according to sections, but nc
B) was before (Eric, B) , but since the sorting algorith
used a stable sorting algorithm, the result would be:

(Carol, A)(Dave, A)(Ken, A)(ALice, B)(Eric, B)

sort on section using unstable sort.

[abdef - - - -

defghi - - - -]

20GB File.

[def abc]

[abc def]

20GB 100 → 200 MB

20GB → 200 MB → sort

→ 200 MB → sort

→ 200 MB → sort

→ 200 MB → sort

100

[abc bca]

[bac def]

abc bac

50 → 25 → 12 → 6 → 3 → 1

MS ([4, 3, 10, 7, 9, 1, 11, 8])

MS([4, 3, 10, 7]) MS([9, 1, 11, 8])

[3, 4, 7, 10] ① [1, 8, 9, 11]

Merge

[1, 3, 4, 7, 8, 9, 10, 11]

mergeSort (arr, l, r)

⇓

→ mergeSort(arr, l, mid)

→ mergeSort (arr, mid+1, r)

merge the 2 halves.

n/2

n/2

~~T(n) = 2T(n/2) + c~~

$$\boxed{T(n) = 2T(n/2) + n}$$

TC : O(n log n).

l = 0 r = 4 (n = 5)

mid = (0+4)/2 = 2

case-1 case-2

(0, 1) (2, 4) (0, 2) (3, 4)

(l = 0 r = 1) (n = 2)

mid = (0+1)/2 = 0

case-1 ✗ case-2

(0, −1) (0, 1) (0, 0), (1, 1)

(0, 1)

(0, 1)

(0, 1)

---

l = 2 r = 5

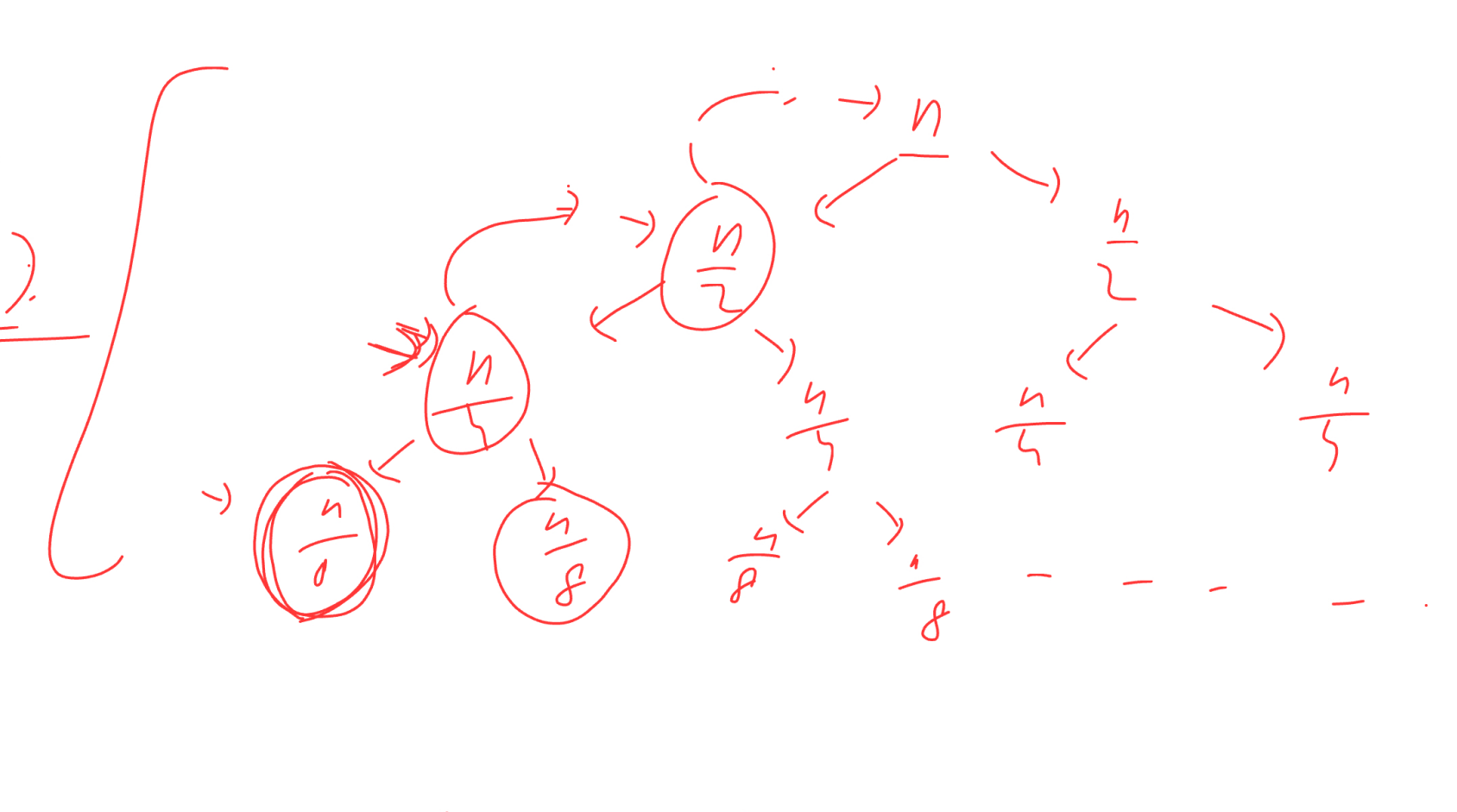
0 1 2 3 4 5 6 7 8

arr: [_ _ 4 7 3 6 _ _ _]

temp: [3, 4, 6, 7]

Ht:

O(log n).



AS = ~~O(n log n)~~

O(n + log n) = O(n)