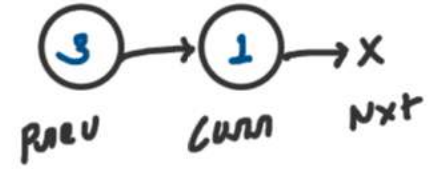
HW 10: Find Minimum and Maximum Number of Nodes Between Critical Points (Leetcode-2048)

Ex 1

Input

Dry Run



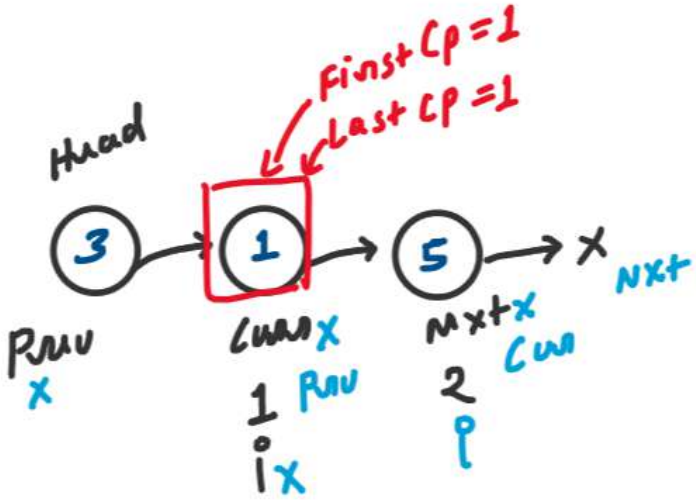
NO Critical point Exist

Output [-1,-1]

```
vector<int> ans = {-1,-1};
Node* prev = head;
if(!prev) return Ans;
Node* curr = head->next;
if(!(curr) returns Ans;
Node* Nxt = curr->next;
if(!Nxt) return Ans;
```

Ex2

Dry Run

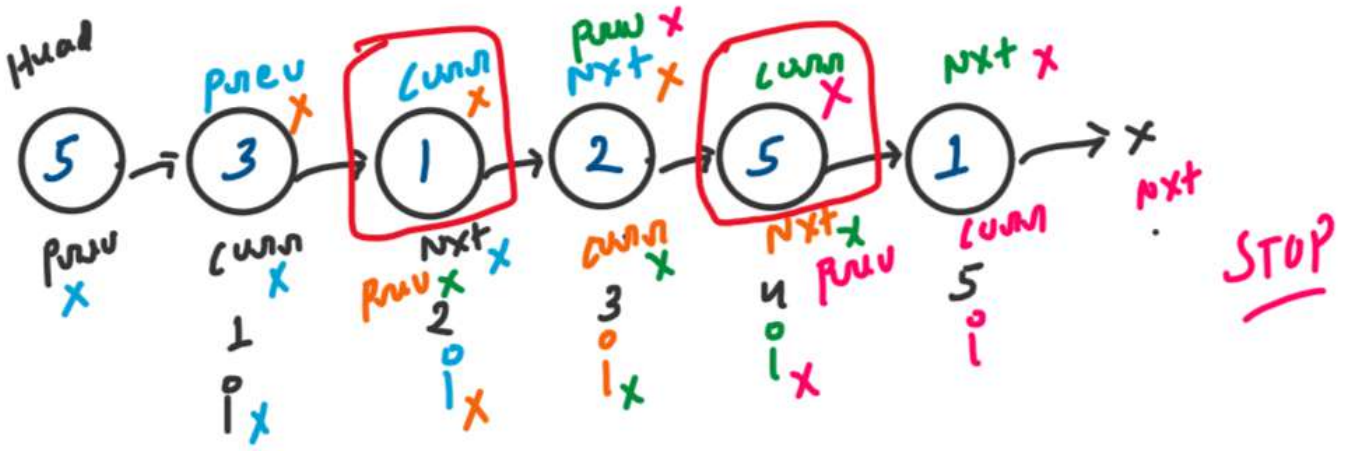


Only Critical point Exist

1 < 3 && 1 < 5

Output = [-1, -1]

Ex: 3



minDis = i - LastCP ⇒ 4 - 2 ⇒ 2

FirstCP = -1 2 2

LastCP = -1 2 4

maxDis = LastCP - FirstCP

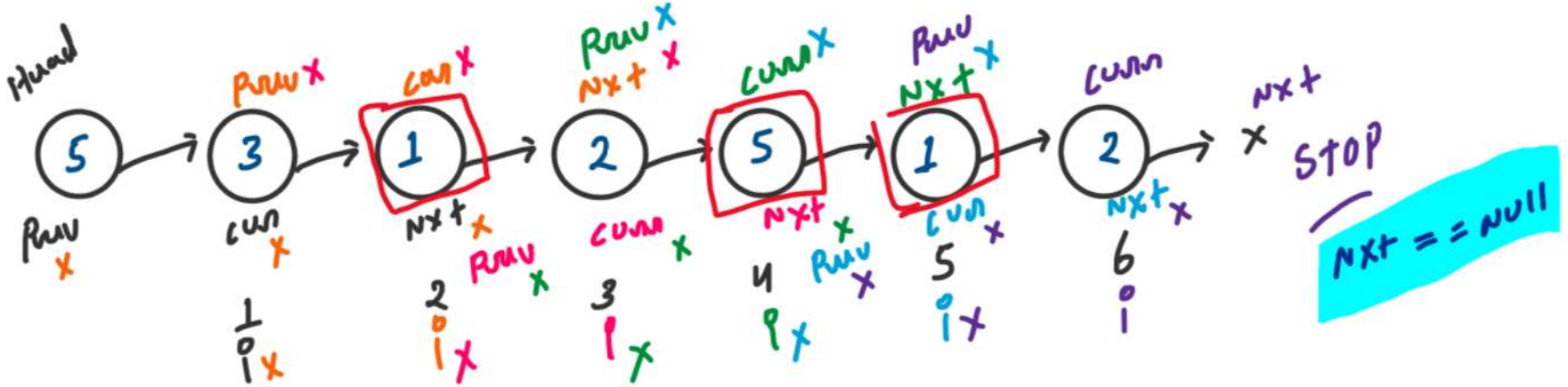
= 4 - 2

= 2

Two Critical points Exist

O/P ⇒ [2,2]

Ex: 4



First CP = ~~1~~ 2

Last CP = ~~1~~ ~~2~~ ~~4~~ 5

minDis = i - lastCP ⇒ 4 - 2 = ~~2~~ ⇒ 5 - 4 = 1

maxDis ⇒ LastCP - firstCP

⇒ 5 - 2 = 3

Three critical points exist

output ⇒ [1, 3]

```cpp
// HW 10: Find Minimum and Maximum Number of Nodes Between
Critical Points (Leetcode-2048)


class Solution {
public:
    vector<int> nodesBetweenCriticalPoints(ListNode* head) {
        // vector initialized with minDis and maxDis
        vector<int> ans = {-1, -1};
        ListNode* prev = head;
        if(!prev) return ans;
        ListNode* curr = prev->next;
        if(!curr) return ans;
        ListNode* nxt = curr->next;
        if(!nxt) return ans;

        int firstCP = -1;
        int lastCP = -1;
        int i = 1;
        int minDis = INT_MAX;

        while(nxt){...}

        // Only one critical poit found condition
        if(lastCP == firstCP){
            return ans;
        }
        else {
            ans [0] = minDis;
            ans [1] = lastCP - firstCP;
        }
        return ans;
    }
};
```

```cpp
while(nxt){
    bool isCP = ((curr->val > prev->val && curr->val > nxt->val)
                || (curr->val < prev->val && curr->val < nxt->val))
                 ? true : false;

    // first critical point condition
    if(isCP && firstCP == -1){
        firstCP = i;
        lastCP = i;
    }
    // at least one critical point ke liye condition
    else if(isCP){
        minDis = min(minDis, i - lastCP);
        lastCP = i;
    }
    i++;
    prev = prev->next;
    curr = curr->next;
    nxt = nxt->next;
}
```

Time complexity: *O(N),*
Where N is number of nodes of the linked list

Space complexity: *O(1),*
Where no extra space used