**LINKED LISTS**

DESCRIPTION OF THE PROBLEM AND FLOW

Falguni Dable

A **Linked list** is another common data structure that complements the array data structure. Similar to the array, it is also a linear data structure and stores elements in a linear fashion. However, unlike the **array**, it doesn’t store them in contiguous locations; instead, they are scattered everywhere in memory, which is connected to each other using nodes. The last node is referenced to NULL. The first is the data and the next is the pointer (\*) which points to the memory location of another node. For the first one 4 is the data and ( \* ) points to the node which has data 2.

|  |  |
| --- | --- |
| 4 | \* |

|  |  |
| --- | --- |
| 2 | \* |

- An Array is a collection of elements of a similar data type. Whereas a Linked List is an ordered collection of elements of the same type in which each element is connected to the next using pointers. Array elements can be accessed randomly using the array index. Random accessing is not possible in linked lists, it follows uni-direction.

- It follows that linked lists should be used for large lists of data where the total number of items in the list is changing. Arrays, on the other hand, are better suited to small lists, where the maximum number of items that could be on the list is known.

The **problem identified is,** how to check if a given linked list contains a cycle or not.

Description**:** Given a linked list, determine if it has a cycle in it. To represent a cycle in a given linked list, we use a counter integer pos which represents the position (0-indexed) in the linked list where tail connects to. If pos is -1, then there is no cycle in the linked list.

Flow: We go through each node one by one and record each node's reference (or memory address) in a table. If the current node is null, we have reached the end of the list and it must not be cyclic. If current node’s reference is in the hash table, then return true. For this, the last node is taken as the first and its pointer points to the memory location of the assigned position and if the memory location exists, then a cycle exists and if no memory location is found then no cycle exists.

For example: Input: head = [3,2,0,-4], pos = 1 -> Output: true

Explanation**:** There is a cycle in the linked list, where tail connects to the second node. As, the tail, which is at position 3 is said to link to the node which is at position 1, which has data 2, so the linked node exists and cycle is seen. Therefore the output is true.

Another example**:** Input: head = [1], pos = -1 -> Output: false

Explanation: There is no cycle in the linked list. Position is said -1 an when we try and link the tail, there is no node to link. So no cycle is seen and the output is false.

By going through this, the cycle which we have seen above can be seen in a stack of plates. Like when we go to attend any functions and where food is been served, the plates which are kept are taken from the top, which will be the end and below will be the top and the way the plates are taken will be the cycle and as soon as the plates are finished, the cycle ends and a new cycle is generated.