June11

10 June 2020 23:47

Hashing and Hash Table

Hash tables are a type of data structure in which the address or the index value of the data element is generated from a hash function. That makes accessing the data faster as the index value behaves as a key for the data value. In other words Hash table stores key-value pairs but the key is generated through a hashing function.

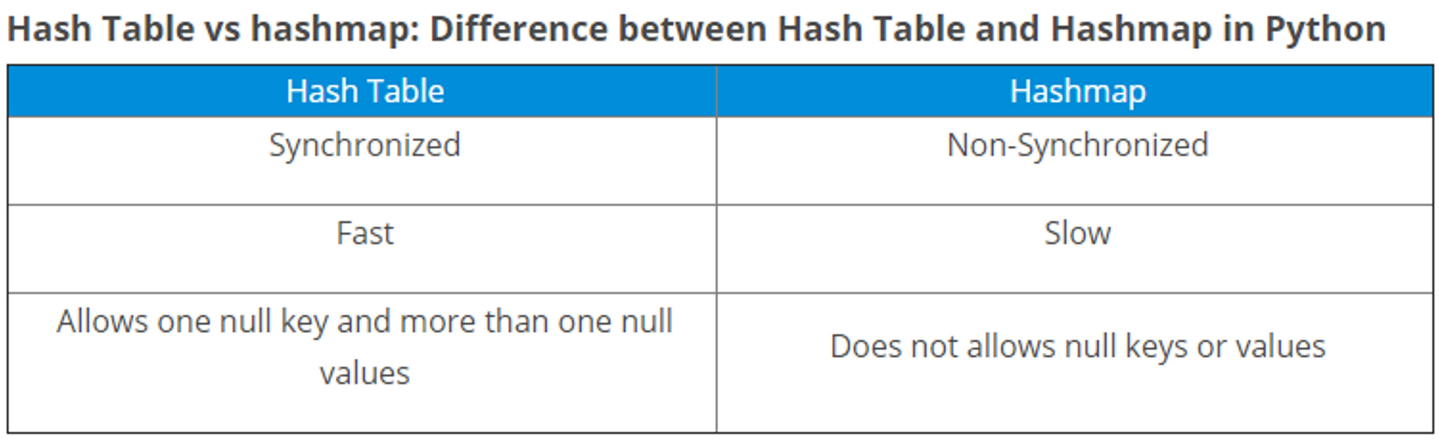
So the search and insertion function of a data element becomes much faster as the key values themselves become the index of the array which stores the data.

In Python, the Dictionary data types represent the implementation of hash tables. The Keys in the dictionary satisfy the following requirements.

- The keys of the dictionary are hashable i.e. the are generated by hashing function which generates unique result for each unique value supplied to the hash function.
- The order of data elements in a dictionary is not fixed.

Hash Tables and Hashmaps in Python: What are they and How to implement?

A Hash table or a Hashmap is a type of data structure that maps keys to its value pairs (implement abstract array data types). It basically makes use of a function that computes an index value that in turn holds the elements to be searched, inserted, removed, etc. This makes it easy and fast to access data. In general, hash tables store key-value pairs and the key is generated using a hash function.

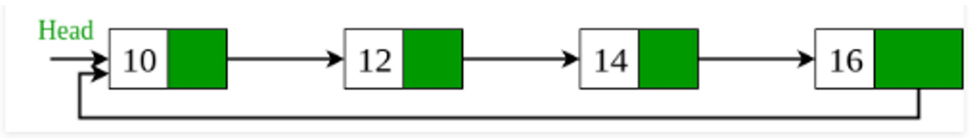


Hash Table vs hashmap: Difference between Hash Table and Hashmap in Python

Hash Table	Hashmap
Synchronized	Non-Synchronized
Fast	Slow
Allows one null key and more than one null values	Does not allows null keys or values

Check if a linked list is Circular Linked List

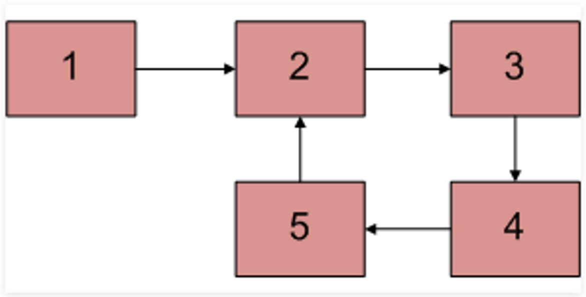
Given a singly linked list, find if the linked list is circular or not. A linked list is called circular if it is not NULL-terminated and all nodes are connected in the form of a cycle. Below is an example of a circular linked list.



An empty linked list is considered as circular.

Detect loop in a linked list using hash table

Given a linked list, check if the linked list has loop or not. Below diagram shows a linked list with a loop.



Algorithm:

Traverse the list one by one and keep putting the node addresses in a Hash Table. At any point, if NULL is reached then return false and if next of current node points to any of the previously stored nodes in Hash then return true.

Questions to implement

1. **Describe python hash() method with example.**
2. **Check whether the given linked list is either NULL-terminated or ends in a cycle (cyclic) using hash table.**
   Note :Time Complexity Should be O(n).
3. **Rajiv and Nitish had a fight because Rajiv was annoying Nitish with his question. Rajiv being a genius in arrays gave Nitish an array of natural numbers A of length N with elements A1, A2, . . ., AN. Nitish has to find the total amount of perfect pairs in the array.**

   A perfect pair (Ai, Aj) is a pair where (Ai + Aj) is a perfect square or a perfect cube and i ≠ j.

   Since Rajiv and Nitish are not talking with each other after the fight you have been given the question to solve and in turn make both of them a perfect pair again.

   NOTE :- A pair (Ai, Aj) and (Aj, Ai) are same and not to be counted twice

   Input

   The first line on the input contains the a single integer T denoting the number of test cases. The first line of each test case
   contains a single integer N. The second line contains N space-separated integers A1,

A2, . . ., AN.

Output
For each test case, print a single line containing a single integer denoting the total number of perfect pairs.

Constraints

1 ≤ T ≤ 10

1 ≤ N ≤ 105

1 ≤ Ai ≤ 103

SAMPLE INPUT

2

5

1 2 3 4 5

4

1 4 5 8

SAMPLE OUTPUT

3

2

Explanation

In first testcase pairs (1, 3), (3, 5) and (4,5) give values 4, 8, 9 and form perfect pairs.

In the second testcase pairs (1, 8), (4, 5) give values 9, 9 and form perfect pairs.