Lecture 3 : Hashmaps

Deadline

Oct 5, 2020, 11:59 PM

Lecture

100.0%

Score 200.00/200

- Introduction to Hashmaps
- Inbuilt Hashmap
- Remove Duplicates
- Code : Maximum Frequency Number 40.0/40
- Code : Print Intersection 80.0/80
- Code : Pair Sum to 0 80.0/80
- Iterators
- Bucket Array and hash function
- Collision Handling
- Hashmap Implementation - Insert
- Hashmap Implementation - Delete a...
- Time complexity and Load factor
- Rehashing

Assignment

0.0%

Score 0/560

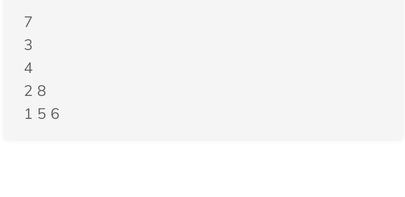
- Longest Consecutive Sequence 0/80
- Pairs with difference K 0/80
- Zero Sum Sub-Array 0/40
- Vertical order 0/80
- Make Strings Anagram 0/80
- Longest Subset 0/120
- Nearest Repetition 0/80

Problem | Result

Vertical order

Send Feedback

Given a binary tree, print that binary tree in vertical order. Vertical order is -



Print the nodes which are at same vertical order in same line separated by space. Print different levels in different lines.

Order of different levels in different lines is not important. But in one level, print the element in pre-order format.

Input Format :

Elements in level order form (separated by space). If any node does not have left or right child, take -1 in its place

Output Format :

Required output in given format

Sample Input :

```
1 2 3 4 5 6 7 -1 8 -1 -1 -1 -1 -1 -1 -1 -1
```

Sample Output :

```
7
3
4
2 8
1 5 6
```

```
void printBinaryTreeVerticalOrder(BinaryTreeNode<int>* root) {

// Following is the structure of the Binary Tree node class
/*
    class BinaryTreeNode {
            public :
                T data;
            BinaryTreeNode<T> *left;
            BinaryTreeNode<T> *right;

        BinaryTreeNode(T data) {
                this -> data = data;
                left = NULL;
                right = NULL;
            }
    };
*/

/* Don't write main().
    * the root of the input binary tree is already passed as function argument.
    * Taking input is handled automatically.
* Print the binary tree in vertical order. Don't return anything.
    */



}
```