**Find the number occurring odd number of times in an array**

**Problem:** Given an array of integers, all numbers occur even number of times except one. You need to find the number which occurs odd number of time.

int ar[] = new int[]{20, 40, 50, 40, 50, 20, 30, 30, 50, 20, 40, 40, 20};

**Solutions:**

**Solution 1**: Use for Loops and compare the element.

Algorithm: 1.Use one for loop to select an element.

2.Use second for loop to compare the element with other elements.

**Solution 2**: Use HashMap and store the elements as Key Value pairs

Algorithm: 1.Declare a HashMap

2.Use a for loop to retrieve the elements from array.

3.Check if the element is already present in the HashMap.

4.If the element is not present then enter the element in the HashMap, with element as the key and 1 as value.

5.Else the element is present then enter the element into the map by increasing the value by 1.

6.Use a EntrySet to iterate over the HashMap

7.Check if the Value of a particular key is not divisible by 2

8.Return the key

int getOddTimesElementHashing(int ar[])

{

int i;

HashMap<Integer,Integer> map=new HashMap<Integer,Integer>();

for (i = 0; i < ar.length; i++)

{

int element=ar[i];

if(map.get(element)==null)

{

map.put(element, 1);

}

else

map.put(element, map.get(element)+1);

}

for (Entry<Integer,Integer> entry:elements.entrySet())

{

if(entry.getValue()%2==1)

{

return entry.getKey();

}

  }

**Solution 3**: Use XOR operation

Properties Of XOR used

1. a ^ a = 0

2. a ^ 0 = a

3. ( a ^ b ) ^ c = a ^ ( b ^ c )

Algorithm: 1. Initialize a integer variable r as 0.

2. Retrieve each element from array and perform XOR operation with r.

3. The value of r at the end of the loop is the number which appears odd number of times

int getOddTimesElement(int ar[])

{

int i;

int result = 0;

for (i = 0; i < ar.length; i++)

{

result = result ^ ar[i];

}

return result;

}