Name: محمود عبد الغني درويش عدس

Id: 21011275

Lab 1 report

Part 1:

Problem statement: You have to implement 4 bits operations, so your program might allow user to choose one of the following operations.

1. getBit(int number, int position): This function returns the bit value (an integer, 0 or 1) in the number at position position, according to its binary representation. The least significant bit in a number is position 0.

2. setBit(int number, int position):This function set the bit value (to be 1) in the number at position position, according to its binary representation. The least significant bit in a number is position 0 and return number after setting the bit.

3. clearBit(int number, int position): This function cleat the bit value (to be 0) in the number at position position, according to its binary representation. The least significant bit in a number is position 0 and return number after clearing the bit.

4. updateBit(int number, int position, boolean value): This function set the bit value according to value parameter which is false (0) or true (1) in the number at positionposition, according to its binary representation. The least significant bit in a number is position 0 and return number after update

---

Used data structures : None

Pseudo code:

```
getBit(number, position):
    number = number >> position
    Number= number AND 1

setBit(number, position):
    temp = 1 << position
    number = number OR temp
```

```
clearBit(number, position):
  temp = 1 << position
  temp = Not temp
  number = number AND temp
updateBit(number, position, value):
  if value is true:
    number = setBit(number, position)
  else:
    number = clearBit(number, position)
  return number

main():
  x = new part1()
  while true:
    print("Choose an operation:")
    print("1 - getBit")
    print("2 - setBit")
    print("3 - clearBit")
    print("4 - updateBit")

    num = input("Enter the operation number: ")
    if num == 1:
      number = input("Enter a number: ")
      position = input("Enter a position: ")
      result = x.getBit(number, position)
      print("Result: " + result)
    else if num == 2:
      number = input("Enter a number: ")
      position = input("Enter a position: ")
```

```
        result = x.setBit(number, position)
        print("Result: " + result)
    else if num == 3:
        number = input("Enter a number: ")
        position = input("Enter a position: ")
        result = x.clearBit(number, position)
        print("Result: " + result)
    else if num == 4:
        number = input("Enter a number: ")
        position = input("Enter a position: ")
        value = input("Enter a logic value (true/false): ")
        result = x.updateBit(number, position, value)
        print("Result: " + result)
    else:
        print("NOT SUPPORTED")
```

Code snippets:

```java
package Discrete_lab1;

import java.util.Scanner;

public class part1 {

    long getbit(long number,long position)
    {
        number=number>>position;
        return number&1;              //done;
    }
    long setbit(long number,long position)
    {
        long temp=1<<position;
        number=number|temp;
        return number;    //done
    }
    long clearbit(long number,long position)
    {
        long temp=1<<position;
        temp=~temp;
        number=number&temp;
        return number;            //done
    }
    long updatebit(long number,long position,boolean value)
    {
        if(value==true)
        {
            number=setbit(number,position);
        }
        else
        {
            number=clearbit(number,position);
        }
        return number;
    }
```

```
public static void main(String[] args) {
          part1 x=new part1();
        while(true)
        {
            System.out.println("choose an operation");
            System.out.println("1-getbit");
            System.out.println("2-setbit");
            System.out.println("3-clearbit");
            System.out.println("4-updatebit");
            long num;
            Scanner in=new Scanner(System.in);
            num=in.nextLong();
            if(num==1)
            {
               System.out.println("enter a number and a position respectively");
               long number,position;
               number=in.nextLong();
               position=in.nextLong();
               System.out.println(x.getbit(number, position));
          }
            else if(num==2)
            {
               System.out.println("enter a number and a position respectively");
               long number,position;
               number=in.nextLong();
               position=in.nextLong();
               System.out.println(x.setbit(number, position));
            }
            else if(num==3)
            {
               System.out.println("enter a number and a position respectively");
               long number,position;
               number=in.nextLong();
               position=in.nextLong();
               System.out.println(x.clearbit(number, position));
            }
```

```
            else if(num==4)
            {
               System.out.println("enter a number and a position and a logic value respectively");
               long number,position;
               boolean value;
               number=in.nextLong();
               position=in.nextLong();
               value=in.nextBoolean();
               System.out.println(x.updatebit(number, position,value));
            }
            else
            {

               System.out.println("NOT SUPPORTED");
            }
```

Assumptions:

1) All numbers are in the domain of integers
2) Position won't exceed number of bits of the integer
3) If the number is negative then the operations will be done on its 2's complement form.

Test cases:

```
choose an operation
1-getbit
2-setbit
3-clearbit
4-updatebit
1
enter a number and a position respectively
1000
6
1
```

```
choose an operation
1-getbit
2-setbit
3-clearbit
4-updatebit
1
enter a number and a position respectively
-1000
5
0
```

```
choose an operation
1-getbit
2-setbit
3-clearbit
4-updatebit
1
enter a number and a position respectively
15
3
1
```

part1 [Java Application] C:\Users\20100\.p2\pool\plugins\org.eclipse.ju

```
choose an operation
1-getbit
2-setbit
3-clearbit
4-updatebit
2
enter a number and a position respectively
734
5
766
```

```
choose an operation
1-getbit
2-setbit
3-clearbit
4-updatebit
2
enter a number and a position respectively
567
9
567
```

```
choose an operation
1-getbit
2-setbit
3-clearbit
4-updatebit

2
enter a number and a position respectively
-120
6
-56
```

```
choose an operation
1-getbit
2-setbit
3-clearbit
4-updatebit
3
enter a number and a position respectively

15
3
7
```

```
choose an operation
1-getbit
2-setbit
3-clearbit
4-updatebit
3
enter a number and a position respectively
16
4
0
```

```
part1 [Java Application] C:\Users\20100\.p2\pool\plugins\org.eclipse.justj.openj
choose an operation
1-getbit
2-setbit
3-clearbit
4-updatebit
3
enter a number and a position respectively
-3
0
-4
```

```
choose an operation
1-getbit
2-setbit
3-clearbit
4-updatebit
4
enter a number and a position and a logic value respectively
20
2
false
16
```

```
choose an operation
1-getbit
2-setbit
3-clearbit
4-updatebit
4
enter a number and a position and a logic value respectively
32
0
true
33
```

Part 2:

1)set data structure

Problem statement: Implement a Set data structure that takes in the constructor a list of strings as a Universe (U). The elements in the Set are subset of U. You must use bits to represent the set. The Set data structure should include the main operations:

1) Add string to the set

2) Union with another set

3) Intersection with another set

4) Complement of the set

5) Difference from another set

6) Cardinality of the set

7) Get elements of the set

Pseudo code:

```
Private array[]          // attributes
Private long S

Set(universal[]):    // constructor
   array = universal

AddString(String x):
   For i = 0 to array.length-1:
      If x equals array[i]:
         S = setBit(S, array.length - i - 1)  // implemented in part 1
         Break

Union(Set q):
   Union = new Set(universal)       // creating a set to hold the union
   Union.S = S OR q.S               // using OR will get the union

Intersection(Set q):
   Intersection = new Set(universal)       // creating a set to hold the intersection
   Intersection.S = S AND q.S              // using AND will get the intersection

Complement():
   Z = 2^(array.length) - 1    // this mask represents the universe, consecutive bits of 1s
   Complement = Z AND (NOT S)    // positions of 1s in the universe and not in this set

Difference(Set q):
   Difference = new Set(universal)    // this set will hold the difference
   Difference.S = q.S AND (NOT S)
```

Cardinality():

Cardinality = bits(S) // this function is from part 3

Get_elements():

ll = new LinkedList()

For i = array.length - 1 to 0:

If getBit(S, array.length - i - 1) = 1:

ll.add(array[i])

Return ll

Code snippets:

```
1  package Discrete_lab1;
2
3  import java.util.LinkedList;
4
5  public class set {
6
7      private static String[] array;
8      private long s;
9
10     public set(String []universal)
11     {
12         this.array=universal;
13     }
14     public void addstring(String in)
15     {
16         int l=array.length;
17         for(int i=l-1;i>=0;i--)
18         {
19             if(array[i].equals(in))
20             {
21                 part1 r=new part1();
22                 s=r.setbit(s, l-i-1);
23                 break;
24             }
25         }
26     }
27     public set union(set q)
28     {
29         set union=new set(array);
30         union.s=this.s|q.s;
31         return union;
32
33     }
```

```
public set intersection(set q)
{
    set intersection=new set(array);
    intersection.s=this.s&q.s;
    return intersection;
}
public set complement()
{
    set complement=new set(array);
    complement.s=((int)Math.pow(2, array.length)-1)&(~(this.s));
    return complement;
}
public set difference(set q)
{
    set difference=new set(array);
    difference.s=this.s&(~(q.s));
    return difference;
}
public long cardinality()
{
    part3 r=new part3();
    return r.bits(this.s);
}
public LinkedList<String> get_elements()
{
    LinkedList<String> ll = new LinkedList<String>();
    int l=array.length;
    part1 w=new part1();
    for(int i=l-1;i>=0;i--)
    {
        if(w.getbit(this.s, l-i-1)==1)
        {
            ll.add(array[i]);
        }
    }
    return ll;

}
```

TestCases : no test cases for this section all of them are applied on set opearion in question 2 of this part

---

Assumption:

1) It is allowed to use objects from other parts from the lab and use their functions

For example:

1)Cardinality Is nothing but number of 1s in the integer which implemented in part 3

2)To add the element in the exact bit we want it is useful to use setbit from part 1 and other things like getting element in a set it is easy to use get bit from part 1

---

2)set operations:

Problem statement : Write a program that.

(a) Asks the user to enter a list of strings as a Universe (U)

(b) Then asks for a number of sets (that are subsets of U).

The user will enter the elements in each set

(c) Then asks the user about the operations they want to perform:

1) Union of two sets

2) Intersection of two sets

3) Complement of a set

4) Difference between two sets

5) Cardinality of a set

6) Print a set

Used Data Structure:

An array to store universe and another one to store only the distinct elements of the universe and an intermediate one between them and an array of sets to store subsets

---

Pseudocode:

```
Main():
l= input("enter universe size")
Universe[] , univ[],uni[]                // this logic is to remove dublicates from the universe
For I=0 to l-1 :
 Universe[I]=input("enter universe element")
c=0,f=0
For I=0 to l-1:
      For j=0 to c :
           If(universal[I] equals univ[j]):
            f=1
```

```
            Break
        If (f = 0):
         Univ[c++]=universal[I]
    f=0
   For I=0 to c-1:
   uni[i]=univ[i];

            // starting progam
            S=input(“enter number of subsets”)
            Subsets[]   // array of sets
            For I=0 to S-1:
            ss= input(“enter size of subset”)
            For j=0 to ss-1:
            Subsets[I].addstring(“enter element”)

while true:
    print("Choose an operation:")
    print("1) Union of two sets")
    print("2) Intersection of two sets")
    print("3) Complement of a set")
    print("4) Difference between two sets")
    print("5) Cardinality of a set")
    print("6) Print a set")

    choice = input()

    if choice == 1:
       print("Enter the number of the sets")
```

```
    x = input()
    y = input()
    u = subsets[x - 1].union(subsets[y - 1])
    ll = u.get_elements()                                   //ll is a linked list
    a = ll.toArray()
    reverse(a)
    print(a)

else if choice == 2:
    print("Enter the number of the sets")
    x = input()
    y = input()
    u = subsets[x - 1].intersection(subsets[y - 1])
    ll = u.get_elements()
    a = ll.toArray()
    reverse(a)
    print(a)

else if choice == 3:
    print("Enter the number of the set")
    x = input()
    u = subsets[x - 1].complement()
    ll = u.get_elements()
    a = ll.toArray()
    reverse(a)
    print(a)

else if choice == 4:
    print("Enter the number of the sets")
```

```
    x = input()
    y = input()
    u = subsets[x - 1].difference(subsets[y - 1])
    ll = u.get_elements()
    a = ll.toArray()
    reverse(a)
    print(a)

else if choice == 5:
    print("Enter the number of the set")
    x = input()
    u = subsets[x - 1].cardinality()
    print("The cardinality is " + u)

else if choice == 6:
    print("Enter the number of the set")
    x = input()
    ll = subsets[x - 1].get_elements()
    a = ll.toArray()
    reverse(a)
    print(a)
```

Code snippets:

```
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.println("Enter the size of the universal");
        int l = in.nextInt();
        in.nextLine(); // Consume the newline character

        String[] universal = new String[l];
        System.out.println("Enter the elements of the universal");
         String sin = in.nextLine().replaceAll("\\[|\\]","");
             universal = sin.split(",");
            String []univ=new String[l];
            int c=0,f=0;
            for(int i=0;i<l;i++)
            {
                for(int j=0;j<c;j++)
                {
                    if(universal[i].equals(univ[j]))
                    {
                        f=1;
                        break;
                    }


                }
                if(f==0)
                {
                    univ[c++]=universal[i];
                }
                f=0;

            }
       String []uni=new String[c];
             for(int i=0;i<c;i++)
             {
                 uni[i]=univ[i];
             }
```

```
System.out.println("Enter the number of subsets");
int s = in.nextInt();
in.nextLine(); // Consume the newline character

set[] subsets = new set[s];
for (int i = 0; i < s; i++) {
    subsets[i] = new set(uni);
}

for (int i = 0; i < s; i++) {
    System.out.println("Enter the size of subset number " + (i + 1));
    int ss = in.nextInt();
    in.nextLine(); // Consume the newline character

    System.out.println("Enter the elements of the subset");

         String sin1 = in.nextLine().replaceAll("\\[|\\]","");
         String[] h = sin1.split(",");
         for(int j=0;j<ss;j++)
         {
             subsets[i].addstring(h[j]);
         }
```

```
while(true)
{
    System.out.println("choose an operation");
    System.out.println("1) Union of two sets");
    System.out.println("2) Intersection of two sets");
    System.out.println("3) Complement of a set");
    System.out.println("4)Difference between two sets");
    System.out.println("5)Cardinality of a set");
    System.out.println("6)Print a set");
    int choice=in.nextInt();
    if(choice==1)
    {
        System.out.println("enter the number of the sets");
        int x=in.nextInt();
        int y=in.nextInt();
        set u=subsets[x-1].union(subsets[y-1]);
        LinkedList<String> ll=u.get_elements();
        Object[] a=ll.toArray();
          Collections.reverse(Arrays.asList(a));
            System.out.println(Arrays.asList(a));

    }
    else if(choice==2)
    {
        System.out.println("enter the number of the sets");
        int x=in.nextInt();
        int y=in.nextInt();
        set u=subsets[x-1].intersection(subsets[y-1]);
        LinkedList<String> ll=u.get_elements();
        Object[] a=ll.toArray();
          Collections.reverse(Arrays.asList(a));
            System.out.println(Arrays.asList(a));
    }



else if(choice==3)
{
    System.out.println("enter the number of the set");
    int x=in.nextInt();
    set u=subsets[x-1].complement();
    LinkedList<String> ll=u.get_elements();
    Object[] a=ll.toArray();
      Collections.reverse(Arrays.asList(a));
        System.out.println(Arrays.asList(a));
}
else if(choice==4)
{
    System.out.println("enter the number of the sets");
    int x=in.nextInt();
    int y=in.nextInt();
    set u=subsets[x-1].difference(subsets[y-1]);
    LinkedList<String> ll=u.get_elements();
    Object[] a=ll.toArray();
      Collections.reverse(Arrays.asList(a));
        System.out.println(Arrays.asList(a));
}
else if(choice==5)
{
    System.out.println("enter the number of the set");
    int x=in.nextInt();
    long u=subsets[x-1].cardinality();
    System.out.println("The cardinality is "+u);
}
else
{
    System.out.println("enter the number of the set");
    int x=in.nextInt();
    LinkedList<String> ll=subsets[x-1].get_elements();
    Object[] a=ll.toArray();
      Collections.reverse(Arrays.asList(a));
        System.out.println(Arrays.asList(a));
}
```

Test cases :

Using one universe :

```
Enter the size of the universal
5
Enter the elements of the universal
[a,b,a,c,d]
Enter the number of subsets
2
Enter the size of subset number 1
3
Enter the elements of the subset
[a,b,a]
Enter the size of subset number 2
2
Enter the elements of the subset
[c,d]
choose an operation
1) Union of two sets
2) Intersection of two sets
3) Complement of a set
4)Difference between two sets
5)Cardinality of a set
6)Print a set
1
enter the number of the sets
1
2
[a, b, c, d]
```

Same universe and subsets:

```
choose an operation
1) Union of two sets
2) Intersection of two sets
3) Complement of a set
4)Difference between two sets
5)Cardinality of a set
6)Print a set
2
enter the number of the sets
1
2
[]
choose an operation
1) Union of two sets
2) Intersection of two sets
3) Complement of a set
4)Difference between two sets
5)Cardinality of a set
6)Print a set
3
enter the number of the set
1
[c, d]
choose an operation
1) Union of two sets
2) Intersection of two sets
3) Complement of a set
4)Difference between two sets
5)Cardinality of a set
6)Print a set
5
enter the number of the set
2
The cardinality is 2
```

Using another universe

```
Enter the size of the universal
12
Enter the elements of the universal
[1,2,3,5,6,7,8,10,6,6,11,12]
Enter the number of subsets
2
Enter the size of subset number 1
10
Enter the elements of the subset
[1,2,2,2,3,3,11,12,6,7]
Enter the size of subset number 2
5
Enter the elements of the subset
[8,10,6,5,3]
choose an operation
1) Union of two sets
2) Intersection of two sets
3) Complement of a set
4)Difference between two sets
5)Cardinality of a set
6)Print a set
1
enter the number of the sets
1
2
[1, 2, 3, 5, 6, 7, 8, 10, 11, 12]
```

```
choose an operation
1) Union of two sets
2) Intersection of two sets
3) Complement of a set
4)Difference between two sets
5)Cardinality of a set
6)Print a set
2
enter the number of the sets
1
2
[3, 6]
choose an operation
1) Union of two sets
2) Intersection of two sets
3) Complement of a set
4)Difference between two sets
5)Cardinality of a set
6)Print a set
3
enter the number of the set
2
[1, 2, 7, 11, 12]
choose an operation
1) Union of two sets
2) Intersection of two sets
3) Complement of a set
4)Difference between two sets
5)Cardinality of a set
6)Print a set
4
enter the number of the sets
1
2
[1, 2, 7, 11, 12]
```

```
[1, 2, 7, 11, 12]
choose an operation
1) Union of two sets
2) Intersection of two sets
3) Complement of a set
4)Difference between two sets
5)Cardinality of a set
6)Print a set
5
enter the number of the set
2
The cardinality is 5
choose an operation
1) Union of two sets
2) Intersection of two sets
3) Complement of a set
4)Difference between two sets
5)Cardinality of a set
6)Print a set
6
enter the number of the set
1
[1, 2, 3, 6, 7, 11, 12]
choose an operation
1) Union of two sets
2) Intersection of two sets
3) Complement of a set
4)Difference between two sets
5)Cardinality of a set
6)Print a set
```

Assumptions:

1) Number of distinct elements in the universe will not exceed the integer bits
2) Elements of any subset will be from the universe. No elements from any other universes
3) When printing a set we remove all duplicates in it

Part 3:

1)non repeated element :

Problem statement:

Write a function that takes a non-empty array of integers nums, where every element appears twice except for one integer, and returns the unique integer. You must implement a solution with a linear runtime complexity and use only constant extra space. you must think for your solution using bits manipulation operation

Used data structure:

An array to store the elements of the input

Pseudocode :

X=0

For I=0 to array.length-1:

X= X xor array[I]

Return X

Code snippets:

```
5  public class part3 {
6      int nonrepeated_num(int []nums)
7      {
8          int l=nums.length;
9          int number=0;
10         for(int i=0;i<l;i++)
11         {
12             number=number^nums[i];
13         }
14
15        return number;
16
17     }
```

Test cases : will be provided in end of part 3

---

2)number of 1s in an unsigned number

Problem statement :

Write a function that takes an unsigned integer and returns the number of ’1’ bits it

Used data structure:

None

Pseudocode:

```
count = 0
    i = 0
    h = new Part1()

    while number != 0:
        count = count + h.getBit(number, 0)
        number = number >> 1

    return count
```

Code snippets:

```
    long bits(long number)
    {
        long count=0;
        int i=0;
        part1 h=new part1();
        while(number!=0)
        {
            count=count+h.getbit(number, 0);
            number=number>>1;

        }
        return count;
    }
```

Test cases : will be put in end of part 3

Driver code for part 3:

```java
public static void main(String[] args) {

    part3 sol=new part3();
    while(true)
    {
        System.out.println("1-find the unique element in array");
        System.out.println("2-find the number of 1 bits in a number");
        Scanner in=new Scanner(System.in);
        int choose=in.nextInt();
        if(choose==1)
        {
            System.out.println("enter the array length");
            int l=in.nextInt();
            int[] nums=new int[l];
            System.out.println("enter the array elements");
            for(int i=0;i<l;i++)
            {
                nums[i]=in.nextInt();
            }

            System.out.println("the unique number is= " );
            System.out.print(sol.nonrepeated_num(nums));
            System.out.println("");

        }
        else if(choose==2)
        {
            System.out.println("enter an unsigned int");
            int num;
            num=in.nextInt();
            System.out.println("number of 1 bits = ");
            System.out.println(sol.bits(num));

        }

    }

}
```

Its pseudo code:

main():

sol = new Part3()

while true:

print("1 - Find the unique element in an array")

print("2 - Find the number of 1 bits in a number")

choose = Input(“enter choice”)

if choose == 1:

print("Enter the array length")

l = Input(“array length”)

nums = new int[l]

print("Enter the array elements")

```
    for i = 0 to l-1:
        nums[i] = Input(“element”)

    print("The unique number is: ")
    print(sol.nonrepeated_num(nums))


else if choose == 2:
    print("Enter an unsigned int")
    num = Input(“number”)

    print("Number of 1 bits = ")
    print(sol.bits(num))
```

Test cases :

```
1-find the unique element in array
2-find the number of 1 bits in a number
1
enter the array length
7
enter the array elements
1
1
2
3
2
5
5
the unique number is=
3
1-find the unique element in array
2-find the number of 1 bits in a number
1
enter the array length
5
enter the array elements
-1
-1
10
0
0
the unique number is=
10
```

```
1-find the unique element in array
2-find the number of 1 bits in a number
2
enter an unsigned int
15
number of 1 bits =
4
1-find the unique element in array
2-find the number of 1 bits in a number
2
enter an unsigned int
1000
number of 1 bits =
6
1-find the unique element in array
2-find the number of 1 bits in a number
```

Bonus part: -

Problem statement: it is required to extract two unique elements in an array of numbers.

Used data structure: only array to store input

Pseudo code:

```
class Bonus:
    findUnique(nums):
      x = 0
      l = length of nums

      for i = 0 to l-1:
         x = x XOR nums[i]

      rightmost_set_bit = x AND (-x)
      n1 = 0
      n2 = 0

      for i = 0 to l-1:
         if (rightmost_set_bit AND nums[i]) != 0:
            n1 = n1 XOR nums[i]
         else:
            n2 = n2 XOR nums[i]

      print("The two unique numbers are " + n1 + " and " + n2)

 main():
   x = new bonus()
 nums[]
   l=input(“enter length”)
For i=0 to l-1:
Nums[i]=input(“enter element”)
      x.findUnique(nums)
```

Code snippets:

```java
public class bonus {
    public void findUnique(int[] nums) {
        int x = 0;
        int l = nums.length;
        for (int i = 0; i < l; i++) {
            x = x ^ nums[i];
        }
        int rightmost_set_bit = x & (-x);
        int n1 = 0, n2 = 0;

        for (int i = 0; i < l; i++) {
            if ((rightmost_set_bit & nums[i]) != 0) {
                n1 = n1 ^ nums[i];
            } else {
                n2 = n2 ^ nums[i];
            }
        }
        System.out.println("The two unique numbers are " + n1 + " and " + n2);
    }

    public static void main(String[] args) {
        bonus x = new bonus();
         Scanner scan= new Scanner(System.in);
         System.out.println("enter size of array");
         int l=scan.nextInt();
         System.out.println("enter the elements");
         int []nums=new int[l];
         for(int i=0;i<l;i++)
         {
             nums[i]=scan.nextInt();
         }
         x.findUnique(nums);

    }
}
```

Test cases:

```
<terminated> bonus [Java Application] C:\Users\20100\.p2\pool\plugins\org.eclipse.justj.oper
enter size of array
8
enter the elements
1
1
2
2
6
4
7
7
The two unique numbers are 6 and 4
```

```
<terminated> Bonus [Java Application] C:\Users\20100\.p2\pool\plugins\org.eclipse.justj.openjdk.hotsp
enter size of array
10
enter the elements
-1
-4
2
2
3
4
4
3
1
1
The two unique numbers are -1 and -4
```

The idea of the solution :

Since the value of Xor all elements of nums will be nothing but the Xor of the two unique numbers

So by finding the rightmost set bit it will be our guide to one of them as all elements and it except one of

Them will give us zero so we have first one and the other is got by Xor the rest of the array.