Issue Date: 28-Mar-2014

Objective:

- Reviewing the Bitwise operators.
- Implementing Bit Vector, which could help us implementing many problems in optimized way.

Bit vector [http://en.wikipedia.org/wiki/Bit_array http://www.sgi.com/tech/stl/bitset.html]

A bit array (also known as bitmap, bitset, bit string, or bit vector) is an array data structure that compactly stores bits. It can be used to implement a simple set data structure.

More often, when we define an array say, which can hold 10 elements of type integer, the total size of the array is 10 * 32, i.e., 320 bits. Do we actually need 320 bits? The answer is no. Say we are storing numbers 4, 6, 5. How many bits are needed to represent them? 3 * 3 = 9 bits.

But the space allocated for these three numbers, is 3 * 32 i.e., 96bits, in short; 86 bits are wasted. Whew! That is one hell of a space loss!

The easiest way is to declare a character array of some desired size and initially make the bits 0 ,i.e., by using the *new* operator and storing the return address in an identifier of type char *. Each block can hold 8 bits since size of character is 8 bit.

Inserting a number

If we want to add 7 to the array:

- Find the block to which 7 belongs How?
 - Divide 7 by 8 (since each block is of 8 bit byte). The quotient is the block number. So, 7 belong to the first block.
- Find the bit position of 7.

Taking the remainder (using the % operator) helps. 7 % 8 is 7. So the 7th bit in the first block is the position of the number 7. Please note that we are not inserting the number 7 into the array, instead we are setting (making the bit 1) the corresponding bit in the corresponding block . Setting the bit means: ORing the value in the block with 1 leftshited n times (n is the bit position)

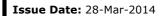
Deleting a number

- Find the position of the number.
- Clear the corresponding bit (Make the bit 0)
 Clearing the bit means: ANDing the value in the block with the negation of 1 leftshifted n times (n is the bit position)

Checking the presence of a number

- Find the block to which the number belongs
- Find the bit position
- If the corresponding bit is 0, the number is not present.

 If the value returned by ANDing number in the block with 1 leftshifted n times (n is the bit position) is 1, the number is present





Task-1: Bit Vector Implementation in C++.

In this task we shall define an ADT BitArray, which will support the operations discussed above.

```
class BitArray
{
private:
   int capacity;
   const int wordSize;
   unsigned char * data;
   int isValidBit(int i)
      return i>=0 && i < capacity;
   }
public:
   BitArray(int n) : wordSize(8)
      capacity = n;
          int s = (int)ceil((float)capacity/wordSize);
      data = new unsigned char[s];
      for (int i=0; i<s; i++)
            data[i] = data[i] & 0;
   BitArray(const BitArray & ref):wordSize(ref.wordSize)
      // Complete yourself
   void on( int value);
   void off(int value);
   int checkBitStatus(int value);
   void invert(int value);
   void dump();
   BitArray AND(const BitArray & ref) const;
   BitArray OR(const BitArray & ref) const;
   ~BitArray();
};
Sample Run:
int main()
   BitArray ba(17);
                            1 00000001 00001110
   ba.on(1);
                            1 00000001 01001010
   ba.on(2);
                            Program ended with exit code: 0
   ba.on(3);
   ba.on(8);
   ba.on(16);
   ba.dump();
   ba.invert(2);
   ba.invert(6);
   cout<<endl:
   ba.dump();
   cout<<endl;</pre>
 }
```

Issue Date: 28-Mar-2014

Task-2: Discussion about the Usage of Bit Vector.

1. Think of implementing Set ADT using Bit Vector, and then compare the time/space factor of your code with Set ADT done in some previous lab of OOP.

Pros:

- isMember, insertElement, removeElement should take O(1) time using Bit Vector.
- Union, intersection, difference are O(N), where N is the size of universe (number of bits/values).
- For non-sparse sets, bit-vectors can represent the set in a compact way.
- Bit operations are typically faster.

Cons:

- · Takes too much memory for large universe.
- · Needs a mapping function for non-integer members.

2. Applications

- Boolean Retrieval
 - You can take some reading material on this topic, if you are interested to explore this field.
- Data Compression.
 - We shall do a lab in future about data compression where you may have to use it.



"Be like a postage stamp. Stick to one thing until you get there."