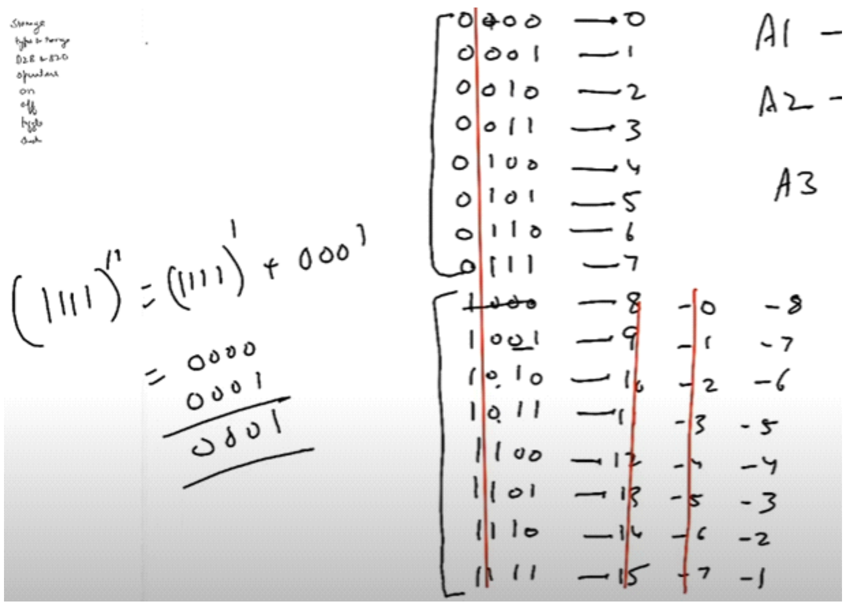
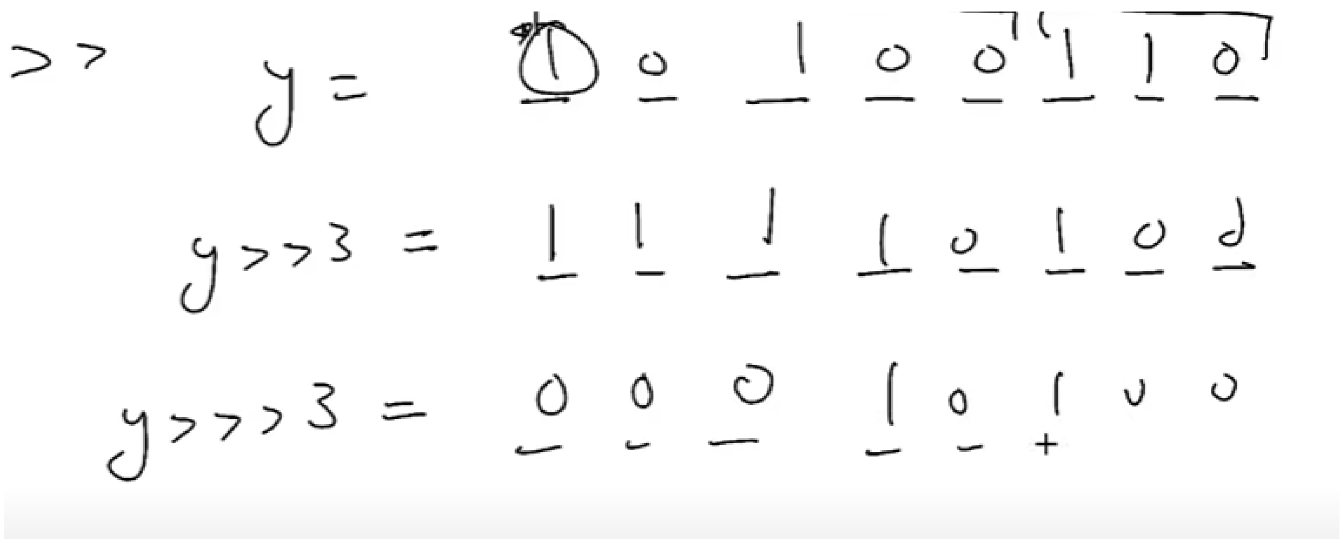
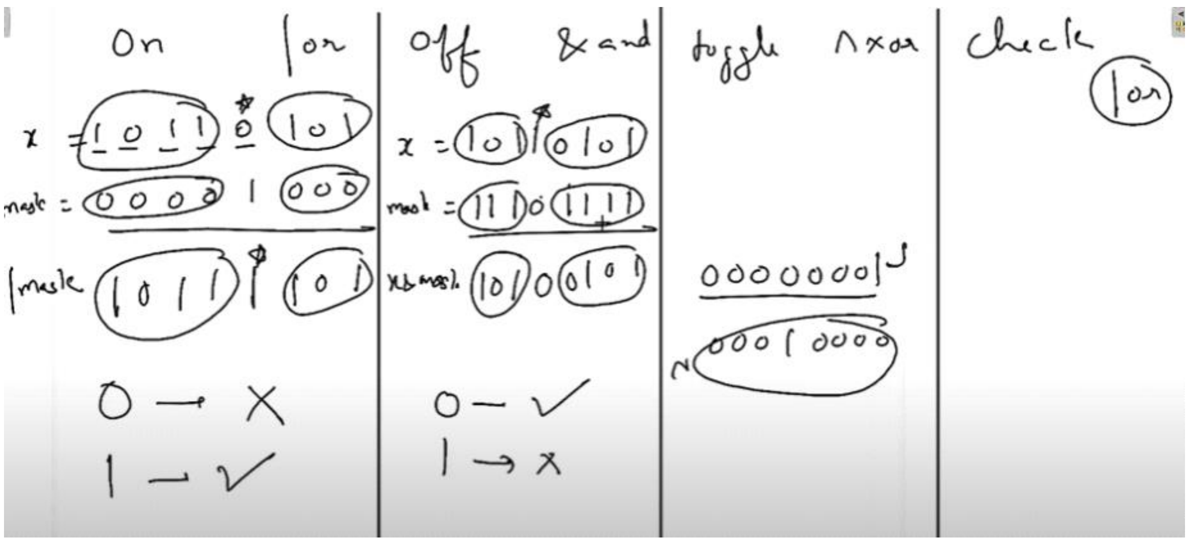
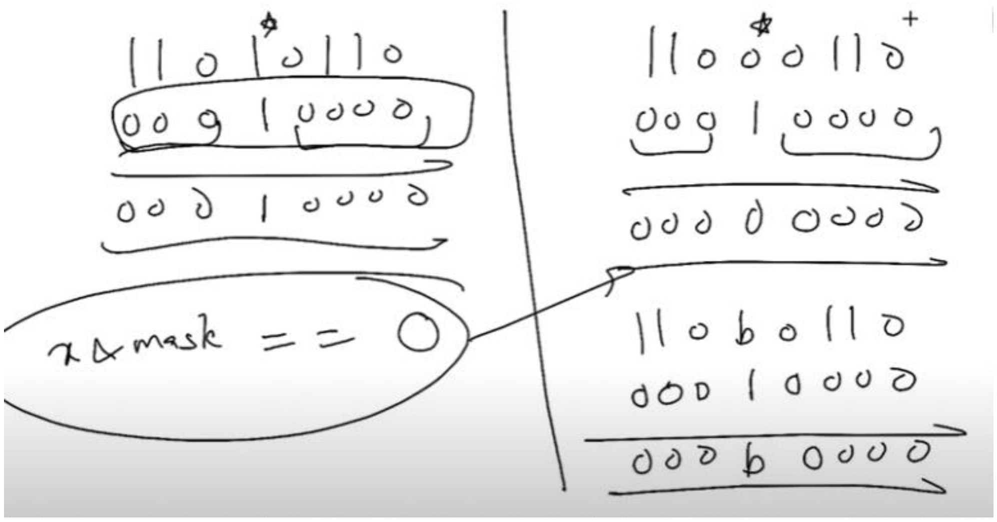
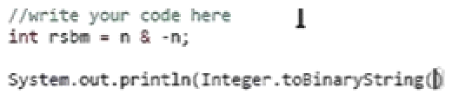

Storage
on
off
toggle
check
0000 — 0
0001 — 1
0010 — 2
0011 — 3
0100 — 4
0101 — 5
0110 — 6
0111 — 7
1000 — 8 -0 -8
1001 — 9 -1 -7
1010 — 10 -2 -6
1011 — 11 -3 -5
1100 — 12 -4 -4
1101 — 13 -5 -3
1110 — 14 -6 -2
1111 — 15 -7 -1
A1 -
A2 -
A3
(1111)'' = (1111)' + 0001
= 0000
0001
0001


>>
y = 1 0 1 0 0 1 1 0
y>>3 = 1 1 1 1 0 1 0 0
y>>>3 = 0 0 0 1 0 1 0 0

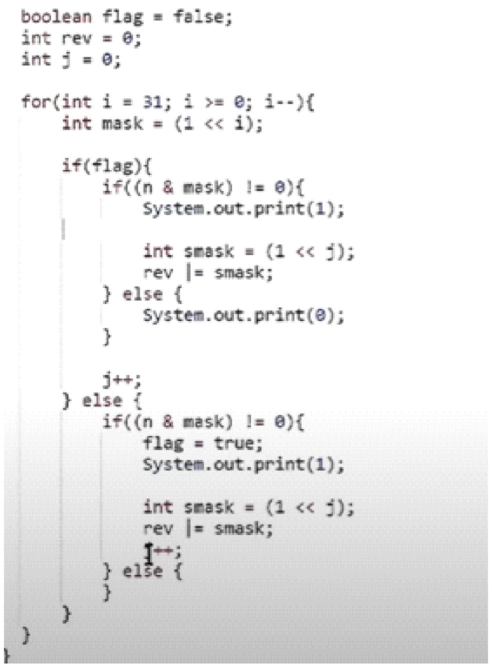

On | or
x = 1 0 1 1 0 1 0 1
mask = 0 0 0 0 1 0 0 0
|mask 1 0 1 1 1 1 0 1
0 → ✗
1 → ✓
Off & and
x = 1 0 1 1 0 1 0 1
mask = 1 1 1 1 0 1 1 1
x&mask 1 0 1 1 0 1 0 1
0 — ✓
1 → ✗
toggle ^ xor
00000001
00010000
check
(or)





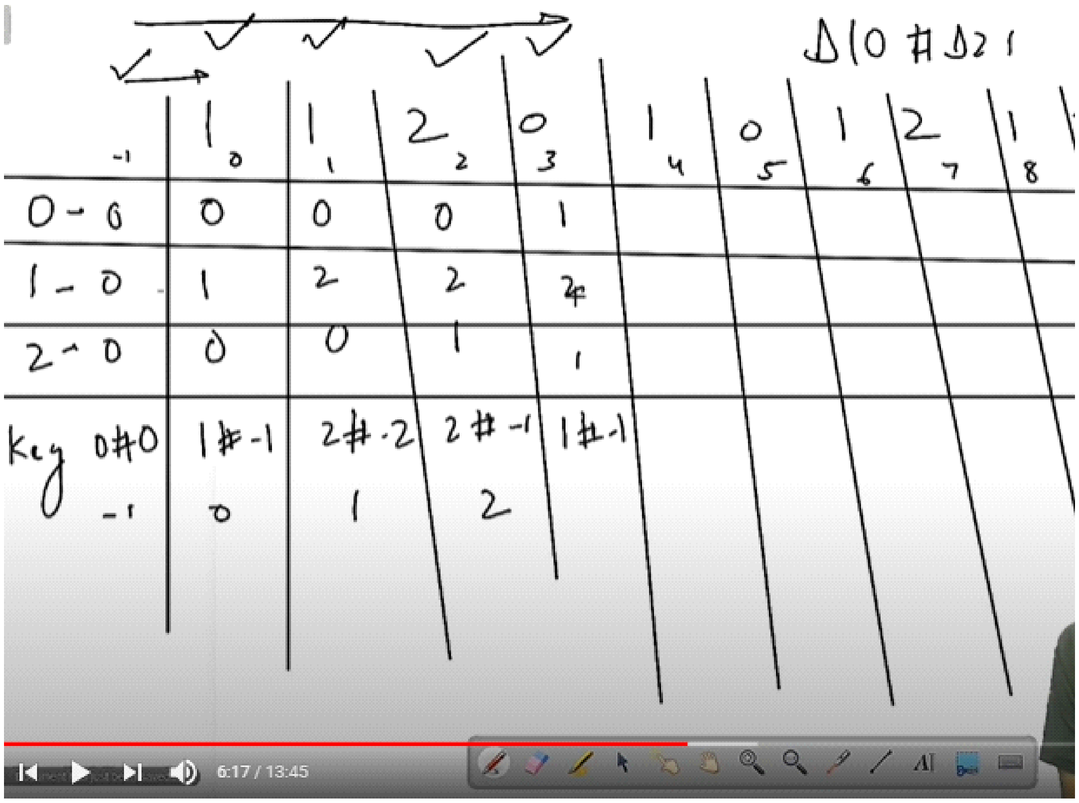
```
//write your code here
int rsbm = n & -n;

System.out.println(Integer.toBinaryString()
```

Most significtnt one -> value & 2's compliment value

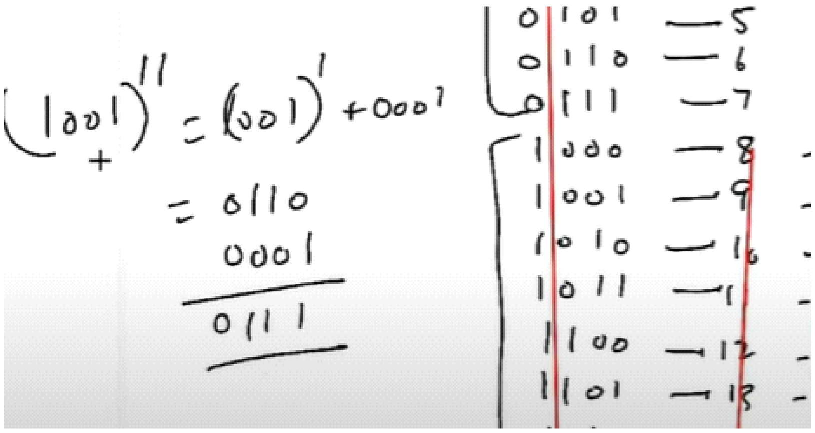


```
boolean flag = false;
int rev = 0;
int j = 0;

for(int i = 31; i >= 0; i--){
    int mask = (1 << i);

    if(flag){
        if((n & mask) != 0){
            System.out.print(1);

            int smask = (1 << j);
            rev |= smask;
        } else {
            System.out.print(0);
        }

        j++;
    } else {
        if((n & mask) != 0){
            flag = true;
            System.out.print(1);

            int smask = (1 << j);
            rev |= smask;
            j++;
        } else {
        }
    }
}
}
```

Reverse bit



ashmaps #datastructure #algorithms
ongest Subarray with Equal 0s 1s and 2s | Hashmap Interview Questions Playlist
2 views • Oct 25, 2020



One compliment toggle whole value
Add+1

```
intv=13;
intw=1<<2;   ~0010=  1101
System.out.println(v&(~w));
Bits off formula at particular position
```

```
Int v=13
Int w =1<<2
System.out.println(v|w) bits on
```

```
System.out.println(Integer.toBinaryString(v));
intw=(~v);
System.out.println(Integer.toBinaryString(v&(w+1)));
intvd=(24&-24);
System.out.println(Integer.toBinaryString(vd));
```

Sum of value

```
Scannersc=newScanner(System.in);
intsum=0;
for(inti=0;i<4;i++){
System.out.println("Entervalue");
sum=sum|1<<sc.nextInt();
}
System.out.println(sum);
```

```
Value=32
Value=31===(1<<5)-1
```

```
25|12
sum|people[i]
```

```
System.out.println(((n<<3)-n)>>3);  =>>8n-n/8
```

```
System.out.println(n<<1); n=8->16
System.out.println(n<<2); n=8->32
System.out.println(n<<3);n=8->64
System.out.println(n>>3);n=64->8
```

```
Quotient remainder
If divide by 4  last 2 will be reminader
15|4
11| 11
```

```
If divide by 8  last 3 will be reminader
15|8
1|111
```

If we shift by any number then it will become twice

```
111<<1-> 111=7
1110->  1*2+1*4+1*8
```

If we shift by two then it will multiplied by 4

$n << x$ $n \propto 2^{x}$

$n >> x$ $\frac{n}{2^{x}} +$

Threads->

Join-> When two thread running then main thread will wait to execute the bot the thread. Because both the thread become after join

Inter thread communictaion-> All should be synchronized
Notify()-> Whent the thread is in waiting state then notify the waiting state to start the resuming the thread executiom
Wait()->When the thread is waiting it is waiting for notify to start execution

Executorservice provide thread pool to execute the task which will automatically work when one thread done the work

Creating the thread is expensive task
Fixed number of thread

```
packagepractice;

importjava.util.concurrent.ExecutorService;
importjava.util.concurrent.Executors;

classCounterimplementsRunnable{
staticintid=0;
intcount;

publicCounter(){
this.count=0;
}

publicsynchronizedintgetCount()throwsInterruptedException{
wait();
this.count=count+1;
System.out.println(count);
returncount;
}


@Override
publicvoidrun(){
try{
getCount();
}catch(Exceptione){

}

}
}

classCounter1implementsRunnable{
staticintid=0;
intcount;

publicCounter1(){
this.count=0;
}

publicsynchronizedvoidgetCount()throwsInterruptedException{

while(true){
this.count=count+1;
System.out.println(count);

notify();
}
}

@Override
publicvoidrun(){
try{
getCount();

}catch(Exceptione){

}
}
}

publicclassApplictaion{
publicstaticvoidmain(String[]args)throwsInterruptedException{
ExecutorServiceexecutorService=Executors.newFixedThreadPool(4);
executorService.execute(newThread(newCounter()));
executorService.execute(newThread(newCounter1()));
}
}
```