Check Bit

Q 1. Given N, set ith bit in N :-

check if ith bit in N is already set, then make no changes.

```
                4 3 2 1 0
N = 26  ⇒       1 1 0 1 0
```

- i = 2
- ↳ set ith bit ⇒ 1 1 1 1 0 → 30 O/P

```
                5 4 3 2 1 0
N = 35  ⇒       1 0 0 0 1 1
```

i = 1

O/P = 35

```
                4 3 2 1 0
ex ⇒  N =       1 0 0 0 1   → 17
```

i = 2 set

⇒ $2^4 + 2^0$

```
                4 3 2 1 0
                1 0 1 0 1
```

⇒ $2^4 + 2^2 + 2^0$ → 17 + 4 = 21.

prev. value + 2^i

↓

(1<<i)

Soln

```
setBit ( N, i ) {
    if ( checkBit ( N, i ) )
        return N
    else
        return N | (1<<i)
}
```

→ O(1)

2^i

TC ⇒ O(1)
SC ⇒ O(1)

Same instruction if / else

1 | 1 = 1

1 | 0 = 1.

eg: N = 150 ⇒ 1 0 0 1 0 1 1 0

i = 4.

	7	6	5	4	3	2	1	0
N =	1	0	0	1	0	1	1	0
x =	0	0	0	1	0	0	0	0
N | x ⇒	1	0	0	1	0	1	1	0

2^4 → 1<<4

→ N.

or. N | (1<<4) Ans.

	7	6	5	4	3	2	1	0	
N =	1	0	0	1	0	1	1	0	
i = 5 ⇒	0	0	1	0	0	0	0	0	← 2^5 → (1<<5)
	1	0	1	1	0	1	1	0	

N | (1<<5) ans

in general

ans = N | (1<<i)

Q2. Given 0 set the x^{th} bit & y^{th} bit.

ex ⇒ I/p = x = 2, y = 5

	5	4	3	2	1	0	
O ⇒	1	0	0	1	0	0	→ $2^5 + 2^2 = 36$

I/p ⇒ x = 3, y = 4

	4	3	2	1	0	
O ⇒	1	1	0	0	0	⇒ $2^4 + 2^3$
						⇒ 24

O → x^{th} bit ⟶ 2^x

O → y^{th} bit ⟶ 2^y

0 → x^{th} & y^{th}

↳ $2^x + 2^y$

⇒ (1<<x) + (1<<y)

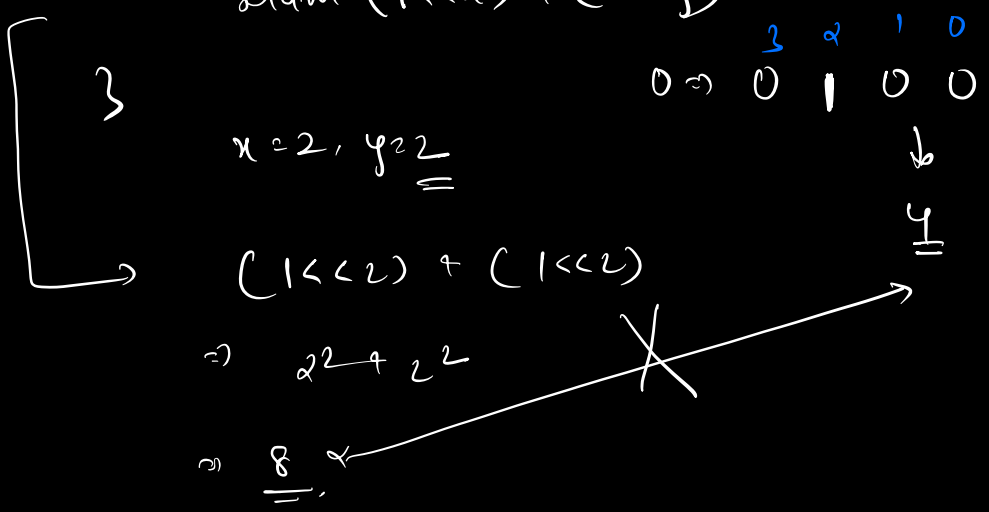
```
setBit (x,y) {
    return (1<<x) + (1<<y)
}
```

x = 2, y = 2

(1<<2) + (1<<2)

⇒ $2^2 + 2^2$

⇒ 8



1 | 1 ⇒ 1

1 | 0 ⇒ 1

```
setBit (x,y) {
    return (1<<x) | (1<<y)
}
```

ex ⇒ x = 2
y = 2

```
  0 1 0 0
  0 1 0 0
-----------
  0 1 0 0  →
```

$x = 3$

$y = 2$

```
1 0 0 0
0 1 0 0
---------
1 1 0 0
---------
```

Q 3. Given N, calculate no. of set bits.

Constr $0 <= N <= 10^9$ → N :- int → 32 bits

N = 150 ⇒ 1 0 0 1 0 1 1 0 → O/P = 4

N = 35 ⇒ 1 0 0 0 1 1 → O/P = 3.

0 → (31)

```
countSetBits(N) {
    count = 0
    for( i=0; i<31; i++) {
        if( checkBit(N, i) )
            count++
    }
}
return count
}
```

TC ⇒ O(1)

SC ⇒ O(1)

: Property :- (N & N-1 == 0)

N	Binary representation	N-1 (Binary)	N & N-1
2 (2^1)	10	1 :- 01	0
4 (2^2)	100	3 :- 011	0
8 (2^3)	1000	7 :- 0111	0
16 (2^4)	10000	15 :- 01111	0

Property :- If N can be represented as a power of 2 then, N & N-1 == 0

Q4. Given x & y, generate a no. that has x set bits followed by y unset bits

ex ⇒ I/p ⇒ x = 2, y = 3 → 11000 ⇒ 24

x = 3, y = 2 → 11100 ⇒ 28

x = 4 y = 2 → 1 1 1 1 0 0 ⇒ 60

x = 3 y = 2 → 4 3 2 1 0 → 1 1 1 0 0 → (1<<2) + (1<<3) + (1<<4)

x = 4 y = 2 → 1 1 1 1 0 0 → $2^5 + 2^4 + 2^3 + 2^2$

decimal ⇒ $2^5 + 2^4 + 2^3 + 2^2$

```
for(i=2; i<=5; i++) {
    ans = ans + 2^i;
```

x = 4 y = 2

```
or. for(i=2; i<=5; i++) {
        ans = ans + (1<<i)
```

TC ⇒ O(x)
SC ⇒ O(1)

```
for(i=y; i<(n+4); i++) {
    ans = ans + ((1<<i)
}
```

a) given x, create a no. with all x set bits.

$x=2 \rightarrow 11 \rightarrow 3 \rightarrow 2^2-1 \rightarrow 2^x-1$

$x=3 \rightarrow 111 \rightarrow 7 \rightarrow 2^3-1 \rightarrow 2^x-1$

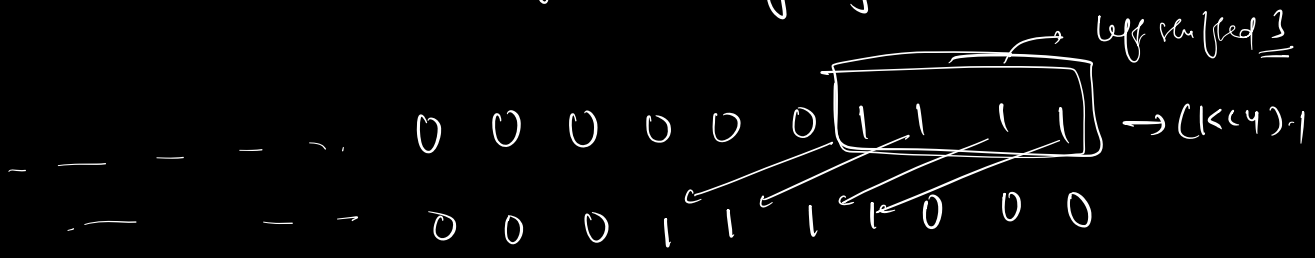
$x=4 \rightarrow 1111 \rightarrow 15 \rightarrow 2^4-1 \rightarrow 2^x-1$

$x=5 \rightarrow 11111 \rightarrow 31 \rightarrow 2^5-1 \rightarrow 2^x-1$

x set bits $\Rightarrow 2^x-1$

$\Rightarrow$ $(1<<x)-1$

with x set bits followed by y unset bits -



$[(1<<4)-1]<<3$ ans.

111 111000

3 bits

4 bits → 0 0 0 1

0 0 1 0

general soln → $[(1<<n) - 1] << y$

Revision Remarks

1) MSB
2) ranges
3) bitwise operators & properties.
4) Basic check bit properties.
5) overflows

Operating System :-

int → $[-2*10^9, 2*10^9]$

a = 10^5

b = 10^6

int c = a*b

= $10^5 * 10^6$

= 10^{11} [XX overflow]

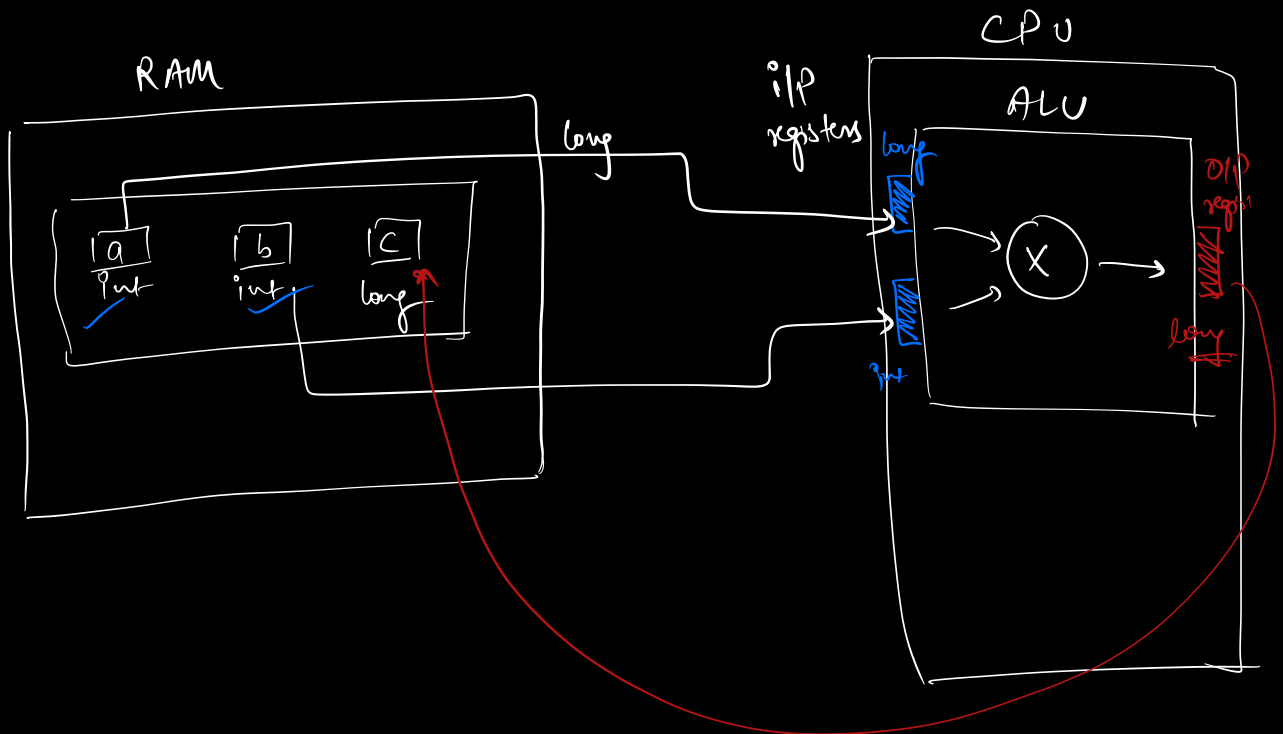
long c = a * b

= 10^{11} [X]

long range $[-2*10^{15}, 2*10^{15}]$

long e = long(a*b) → Conversion happens after a * b is done

= 10^{11} [X]



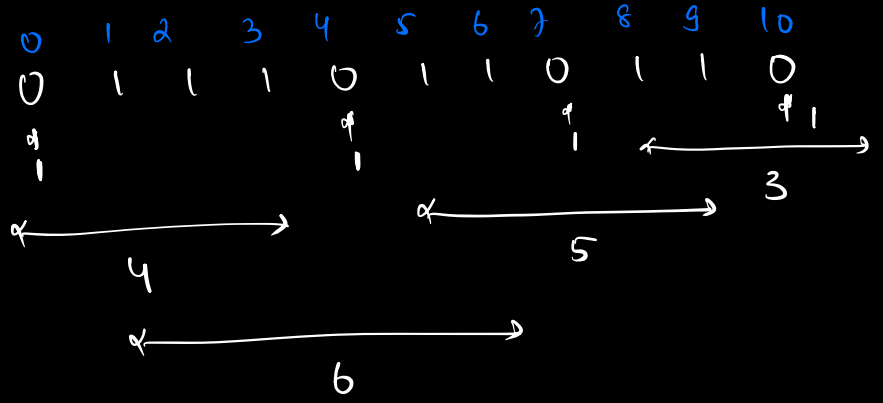
long c = (long) a * b.

long c = (long) a * (long) b

long c = a * (long) b.

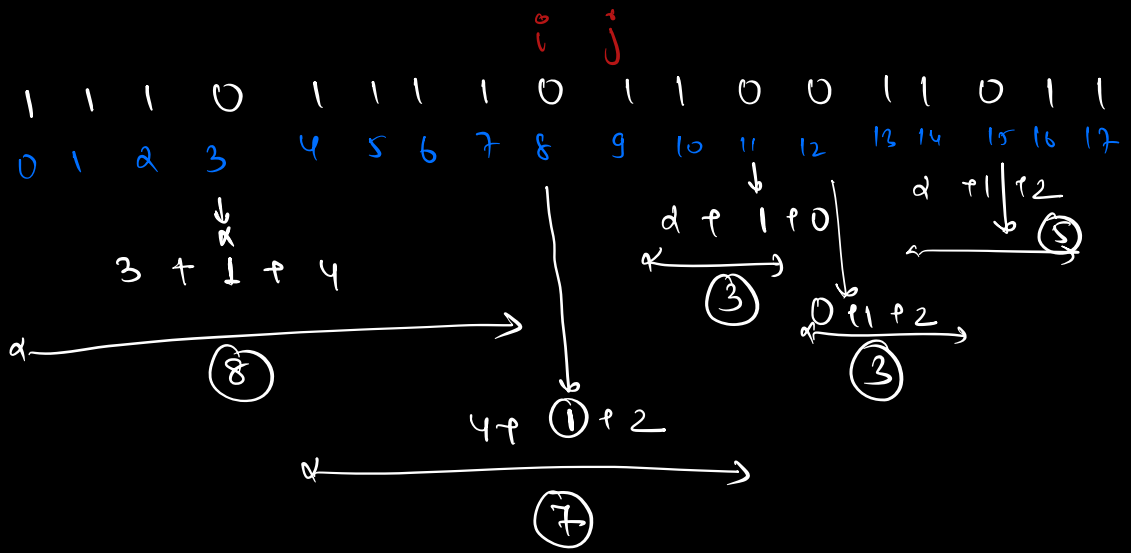
Q 5. Given a binary arr[]. we can atmost replace a single 0 with 1. find maxm consecutive 1's possible.

eg ⇒ arr[] →



O/P = 6

eg ⇒



O/P = 8

idea

when we are at 0

→ consecutive 1's on left → LC

→ " 1's on right → RC

→ total 1's → LC + RC + 1

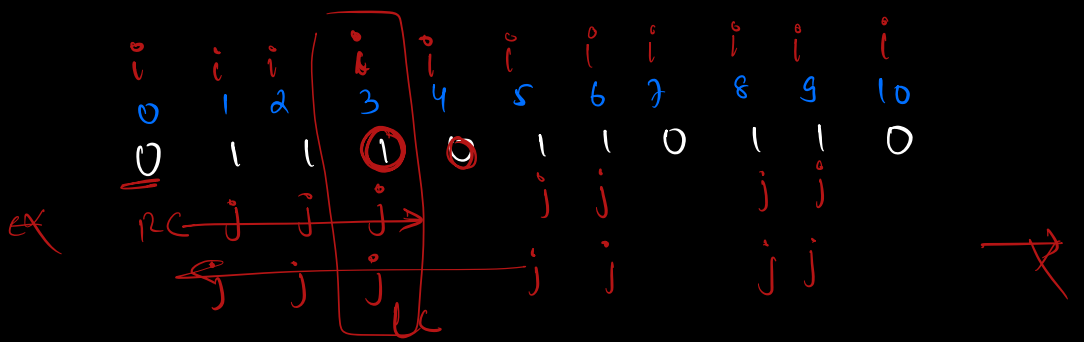
ans = max (all totals for all zeros)

pseudo

```
for( i=0; i<N; i++) {
    if(arr[i]==0){
        LC=0, RC=0
        for( j=i-1; j>=0; j--) {
            if (arr[j] ==1)
                LC++
            else
                break
        }
        for( j=i+1; j<N; j++) {
            if (arr[j]==1)
                RC++
            else
                break
        }
        total = LC + RC + 1
        ans = max (ans, total)
    }
}
```

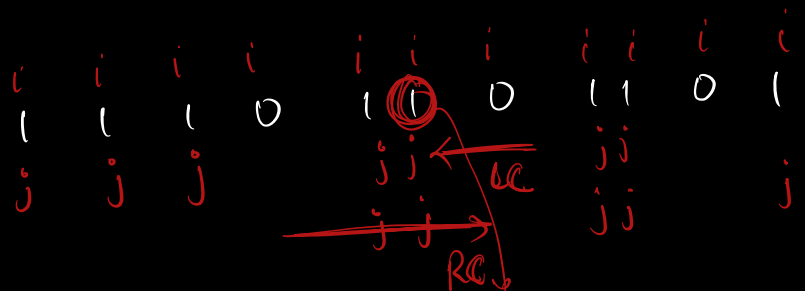
edge cases → all are 1's → return N

all are 0's → return 1.



TC = 3N ≈ O(N)

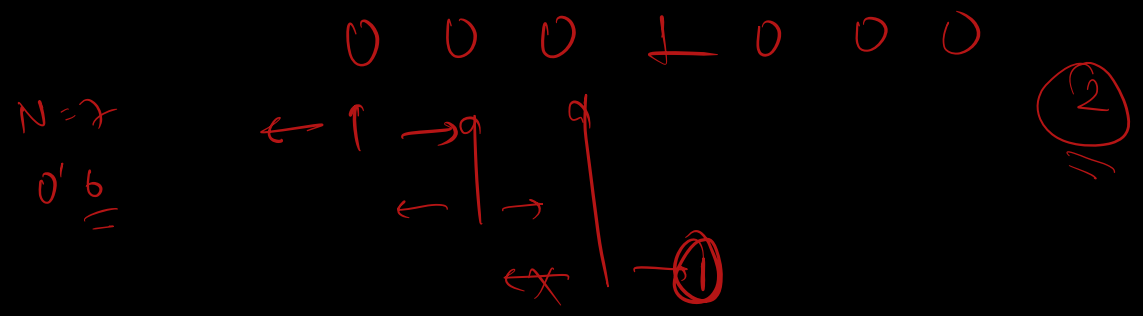
SC = O(1)



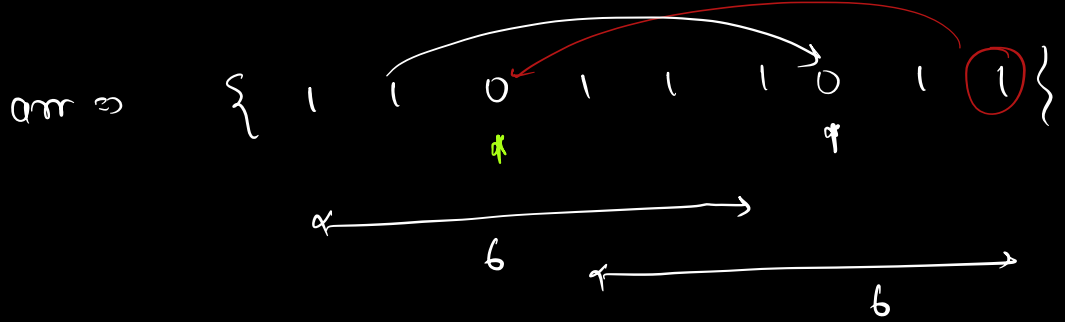
i → outer

i → inner LC

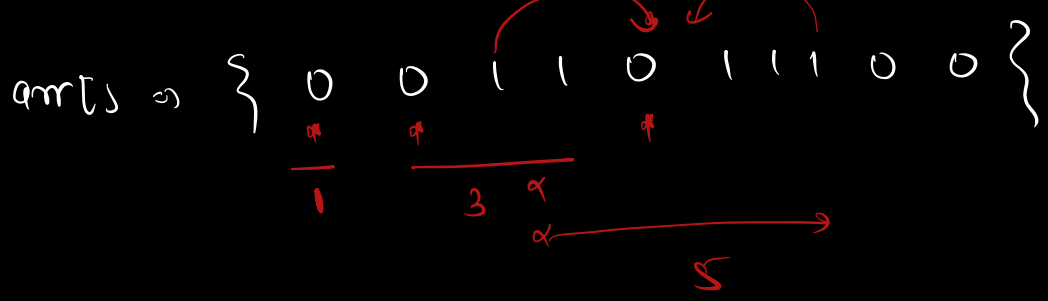
i → inner RC



Qs. Given an arr[], can we swap a single 0 with 1 from the arr, find maximum consecutive 1.

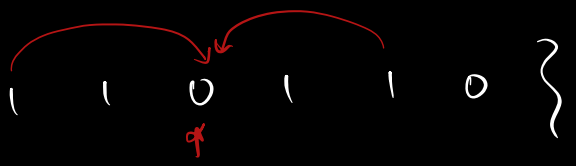


o/p = 6.



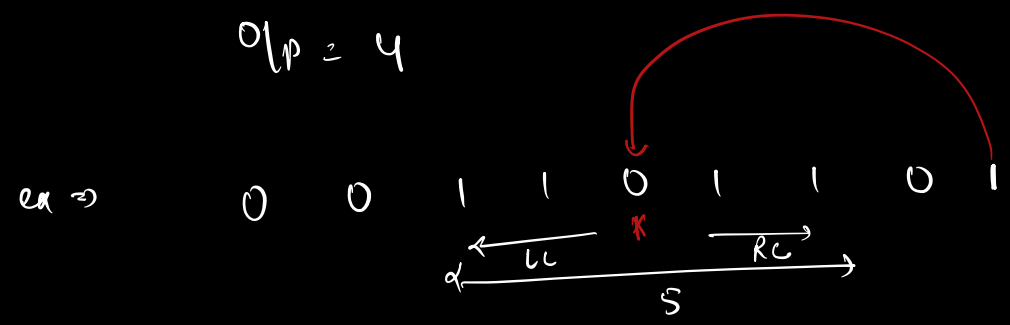
o/p = 5

arr[] = { 0 0 1 1 0 1 1 0 }



o/p = 4

eg =



o/p = 5

CI = 5

LC = 2

RC = 2

(LC + RC) < CI.

↓

consec is = LC + RC + 1

eg =

CI = 4

LC = 2

RC = 2

(LC + RC) = = CI.

norm cons = LC + RC

SOM

i) find Ct (total count of 1)

ii) for each zero

  i) find LC

  ii) find RC

  iii) if ((LC+RC) < Ct)

    total consec 1 = LC+RC+1

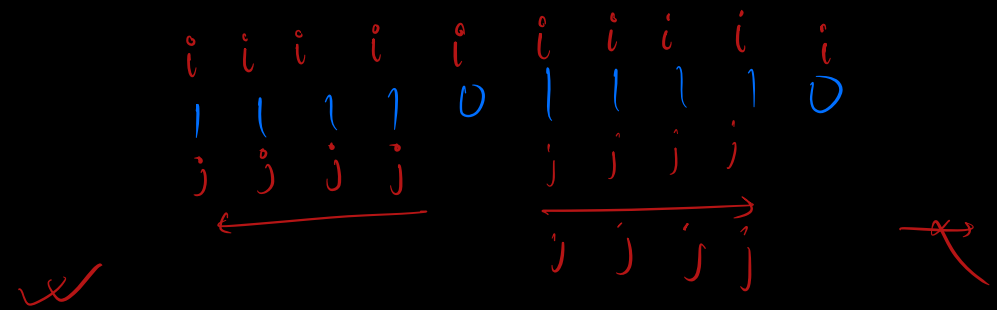
  else

    total con 1 = LC+RC

ans = max (ans, total cons 1)

TC ⇒ O(N)

SC ⇒ O(1)



for(i=0; i<N; i++)

arr[i]

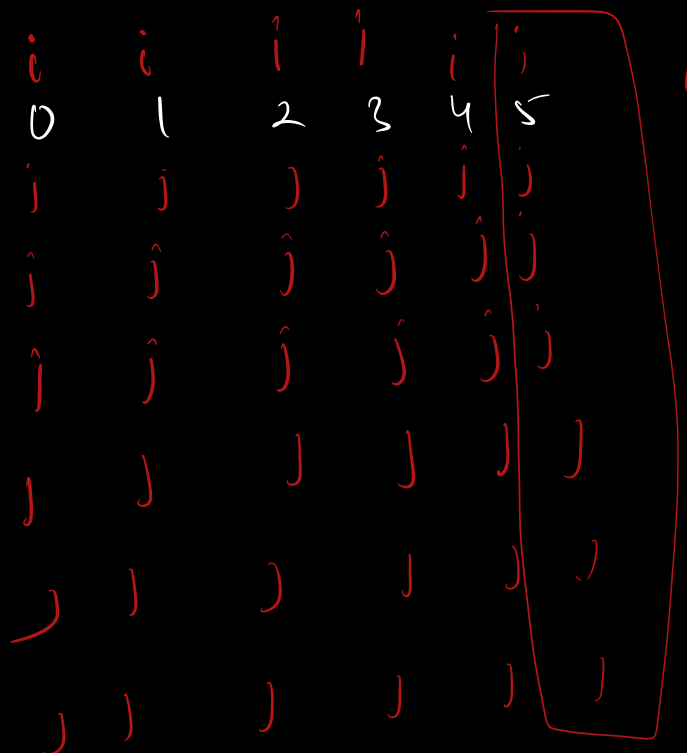
i i i i i i

0 1 2 3 4 5 → $O(N)$

$O(N+1)$

N = 3 → $3N$

```
for( i=0; i<N; i++)
    for(j=0; j<N;j++)
        print(j)
```



$O(N*(N+1)$

↓

$O(N^2)$

Swap

1 0 0 0 0 1 0 0 0 0 1 0 0 0 1 0 0 0 0 1

0 1 2 3

N   bits   => 0 0 0 0 0 0 1 1

N >> 1 => 0 0 0 0 0 1 1 0

N << 1 => 0 0 0 0 1 1 0 0

N << 3 => 0 0 0 1 1 0 0 0

1 << 3 => 0 0 0 0 1 0 0 0

2^3 = 8

(1 << 3) - 1 => 0 0 0 0 0 1 1 1

2^2 2^1 2^0

2^3 - 1 = 7