Day 11

28 March 2025 08:33 AM

DSA

Time Complexity :-

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

$O(n)$

```
a = a+2;
```

$\longrightarrow O(1)$

n 10

```
for(i=1; i<=n; i=i*2)
{
    ==
}
```

$O(\log_2 n)$

$1 \to 2 \to 4 \to 8 \to 16 \to 32 \ldots$

$32 \to 16 \to 8 \to 4 \to 2 \to 1$

n 8

$8 = 3$ ← Conv

$n = 2^?$

$n = 2^?$

$? = \log_2 n$

```
for(i=1; i<=n; i++)
{
    for(j=1; j<=n; j++)
    {
        ==
    }
}
```

$n * n$

$O(n^2)$

```
for(i=1; i<=n; i++)
{
    for(j=1; j<=n; j++)
```

```
for(j=1; j<=n; j++)
{
    for(k=1; k<=n; k++)
    {
        =
    }
}
```

$n \times n \times n$ — $O(n^3)$

```
for(i=1; i<=n; i++)
{
    for(j=1; j<=n; j=j*2)
    {
        -
    }
}
```

$\log_2 n * n$ — $O(n \log_2 n)$

$O(1)$
$O(n)$
$O(\log_2 n)$
$O(n^2)$
$O(n \log_2 n)$
$O(n^3)$

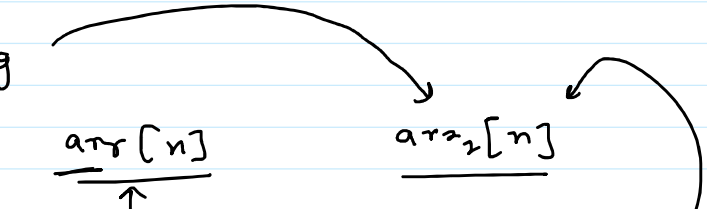
$O(1)$
$O(\log_2 n)$
$O(n)$
$O(n \log_2 n)$
$O(n^2)$
$O(n^2 \log_2 n)$
$O(n^3)$

```
for(i=1; i<=n; i++)
{
    for(j=1; j<=m; j++)
    {
        =
    }
}
```

$n \times m$ — $O(n \times m)$

space Complexity :→

* searching



arr[n] ↑ search

arr2[n] space O(n)

searching

arr[n] ↑ searching

X space O(1)

searching

arr[n]

arr2[n][n] space O(n²)

Bitwise

& | ^ << >> ~

a (dec) ⟶ (binary)

n 25 ⟶ 8 bit 00011001

127 —— -128

5 ~5 → -6

-10 ~(-10) → 9

11001

"10011"

Empty string

string s1 = "";

n 25 6 3 1 r 0

while (n)

{

a = n%2;

if (a==1)

'1'

2 | 25 | 1
2 | 12 | 0
2 | 6 | 0
2 | 3 | 1
2 | 1 | 1
0

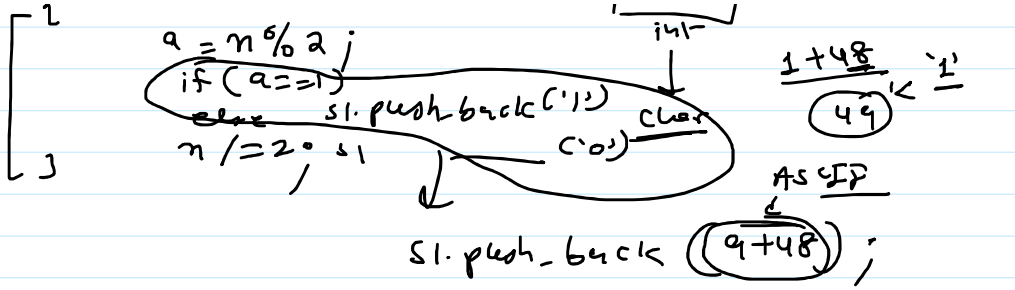
a = 25%2

q 1 0

init

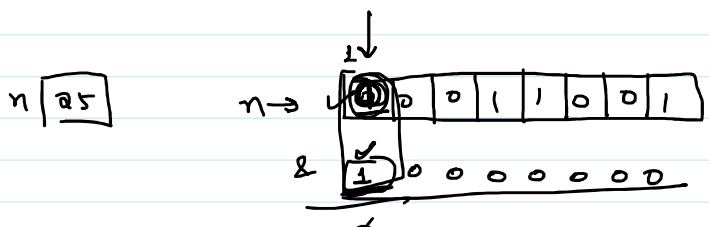
48 to 57

1+48 '1'




```
for(i=s1.length()-1; i>=0; i++)
{
    cout<<s1[i];
}
```

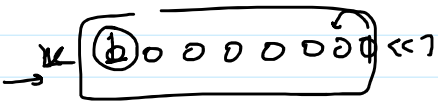
s1 = "10011"



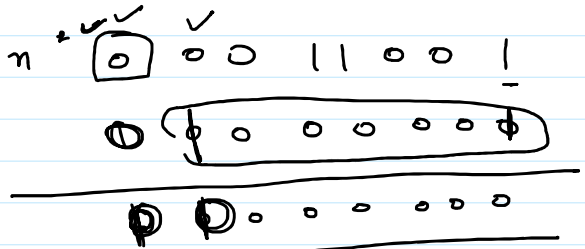
x = 1<<7

(n & x)

n &




```
n 25
for(i=7; i>=0; i--)
{   x = 1<<i
    if (n&x)
        cout<<1;
    else
        cout<<0;
}
```




```
#include<iostream>
using namespace std;
int main()
{
    int n;
    cout<<"Enter a number:";
    cin>>n;
    for(int i=15;i>=0;i--)
    {
        int x =1<<i;
        if(n&x)
            cout<<1;
        else
```

```
            cout<<0;
        }
        return 0;
    }
```

```
#include<iostream>
#include<bitset>
using namespace std;
int main()
{
    int n;
    cout<<"Enter a number:";
    cin>>n;
    bitset<16> a(n);
    cout<<a;
    return 0;
}
```

print nth bit of a number

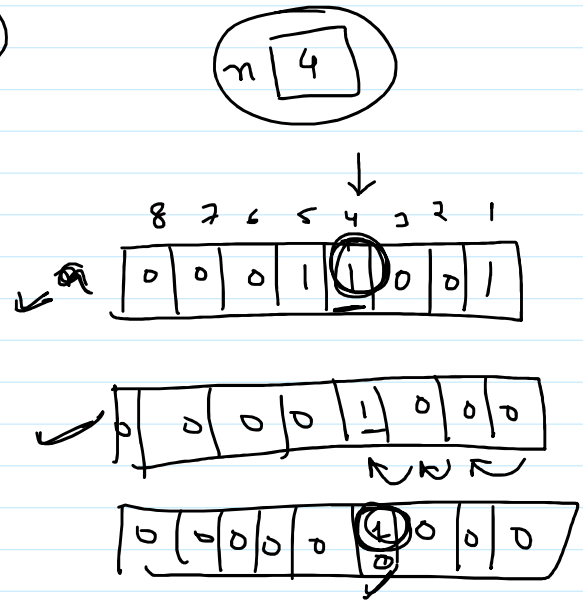
a | 25 | n | 4 |

0 0 0 1 1 0 0 1

8 7 6 5 4 3 2 1

```
int a=25, n=4;
{ if ( a & 1<<(n-1) )
        cout<<1;
  else
        cout<<0;
```

```
if ( a & 1<<(n-1) )
        cout<<1
else
        cout<<0
```

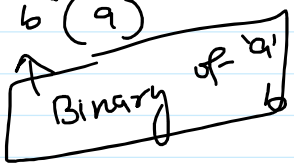


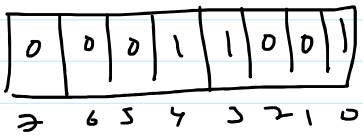
a | 25 |
n | 4 |

bitset<8> b(9)

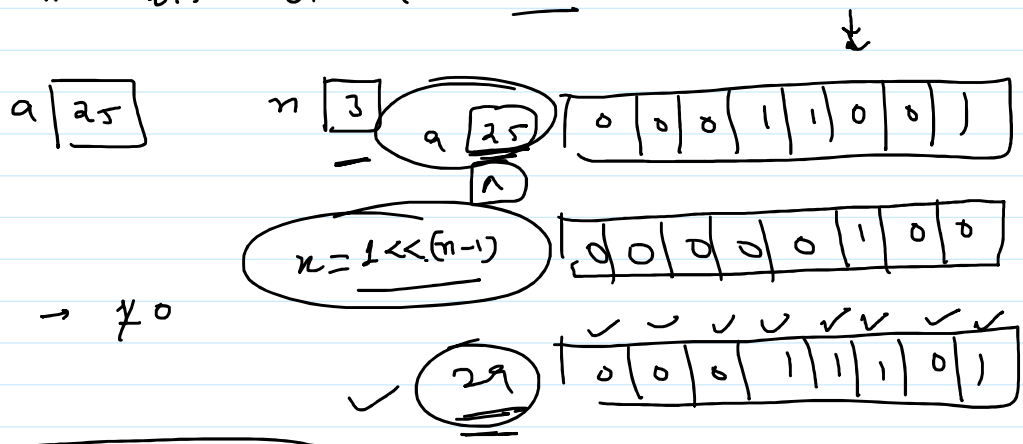
bitset<8> b(9)

cout<<b[n-1]





Reverse nth bit of a num



0 ^ 1 → !0

1

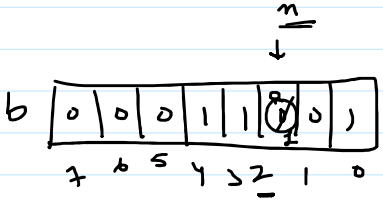
b = a ^ x

```
#include<iostream>
#include<bitset>
using namespace std;
int main()
{
    int n,a;
    cout<<"Enter a number and bit position:";
    cin>>a>>n;
    // int x = 1<<(n-1);
    // int b = a^x;
    int b = a ^ 1<<(n-1);
    cout<<b;
    return 0;
}
```

a 25

n 3

bitset<8> b(9)



b[n-1] = 1

if(b[n-1] == 0)

b[n-1] = 1;

else

b[n-1] = 0;

(1 0 1 0 1 0 1 0)

1 0 1 0 1 ... 1 0 1 0 1 0

n = 21 → True

10 → 1 0 1 0

1 0 1 0 1

0 0 0 0 1 0 1 0

n = 10

n = n >> 1;

n = 0 0 0 0 1 0 1 0

n = 0 0 0 0 0 1 0 1

1 0 1 0

n = 0 0 0 0 1 1 1 1

1 0 0 0 0

(n+1) & n == 0

n = n ^ (n>>1)

(n & n+1) == 0

return True

else

return false

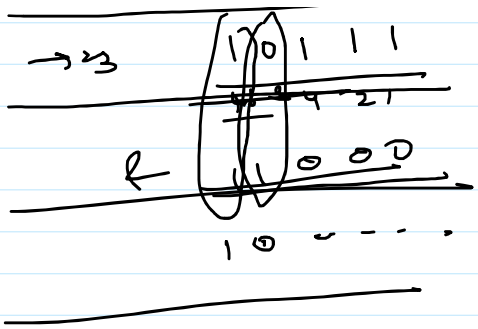
n = 1 1 0 1 0

^ 0 1 1 0 1

1 0 1 1 1

n ^ (n >> 1)

21+1 → 24



```
#include<iostream>
#include<bitset>
using namespace std;
int main()
{
    int n,a;
    cout<<"Enter a number";
    cin>>n;
    n = n^(n>>1);
    if((n & (n+1)) == 0)
        cout<<"True";
    else
        cout<<"False";
    return 0;
}
```

Power of a number →

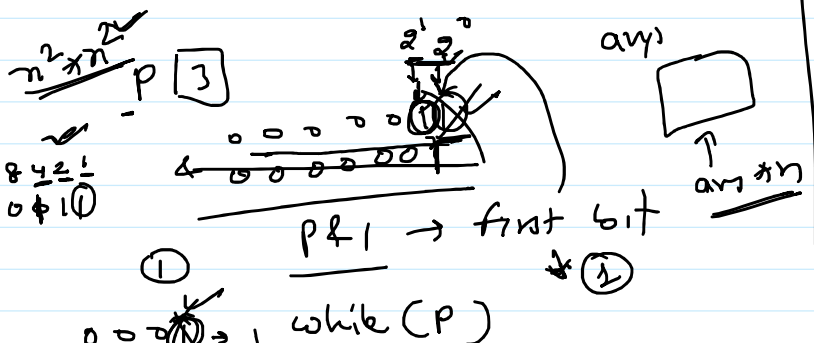
pow (5, 3)

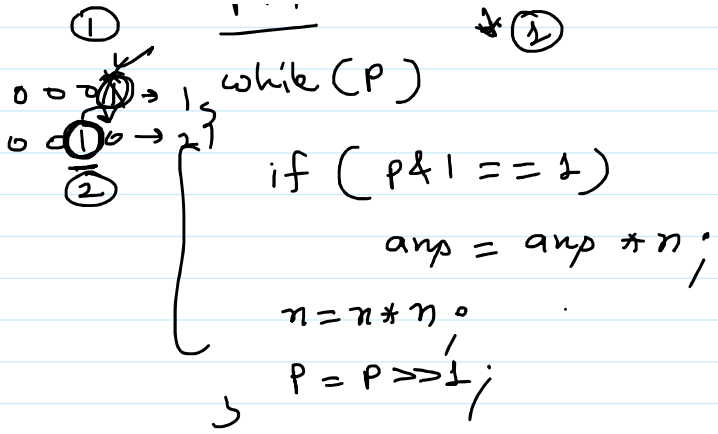
n^p

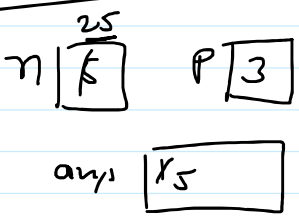
```
ans=1
while (p)
{   ans *=n;
    p--;
}
cout<<ans;
```

$O(p) \rightarrow (\log_2 p)$

```
power(n,p)
if(p==0)
return n* power(n,p-1);
```

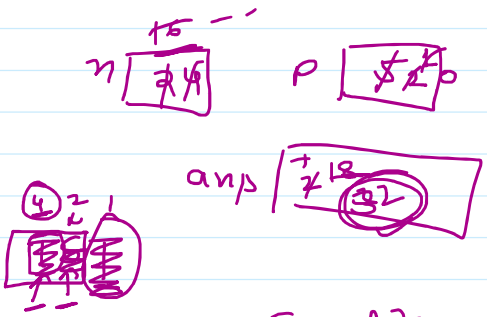


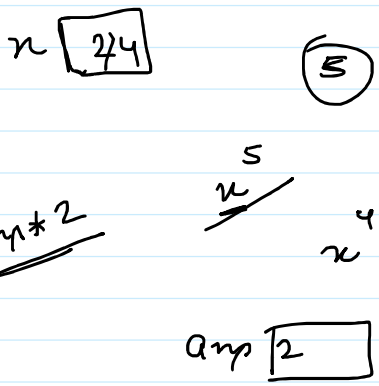


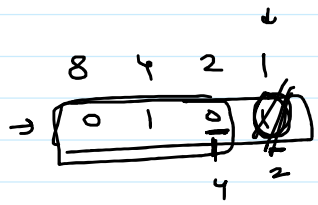



```
#include<iostream>
#include<bitset>
using namespace std;
int main()
{
    int n,p,a;
    cout<<"Enter a number and its power:";
    cin>>n>>p;
    int ans=1;
    while(p)
    {
        if(p&1)
        ans *= n;
        n=n*n;
        p=p>>1;
    }
    cout<<ans;
    return 0;
}
```

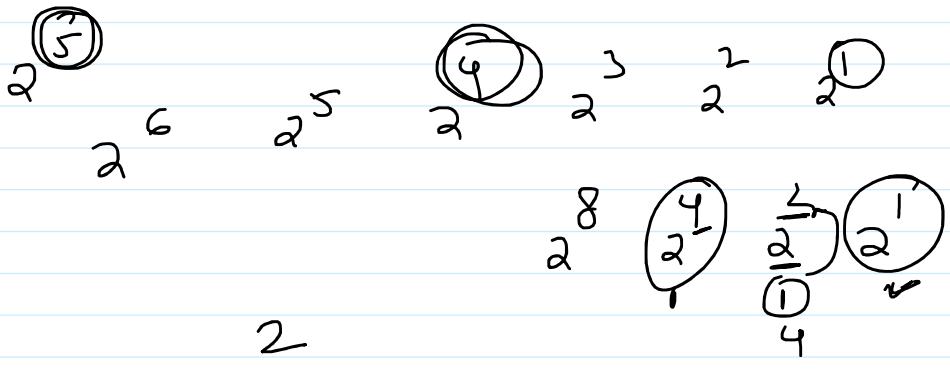
non zero







x = x * x

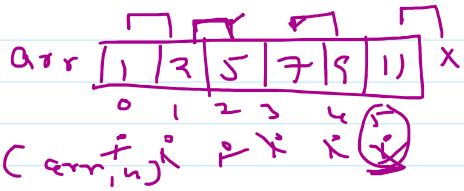


$2^4 = 2 \times 2$

Array →

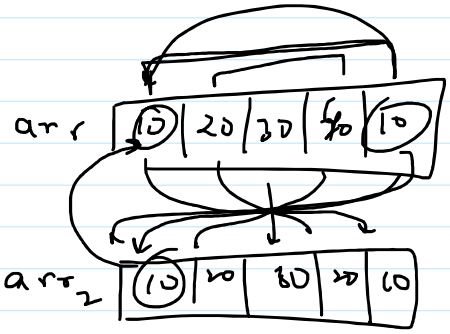
is increasing order :→

n 6




```
bool isIncreasing(arr, n)
{
    for(i=0 ; i<n-1 ; i++)
    {
        if (arr[i] > arr[i+1])
            return false;
    }
    return true;
}
```

is palindrome




```
for(i=0, j=n-1 ; i<j ; i++, j--)
{
    if(arr[i] != arr[j])
        return false;
}
return true;
```

1	2	3	4	5	6

or

```
for (i=0 ; i<n/2 ; i++)
{
    if (arr[i] != arr[n-i-1])
```

```
    if (arr[i] != arr[n-i-1])
        return false;
}
return true;
```