2D Array — Random access

Rows & columns

↳ Matrix computation

- ↳ Multiplication
- ↳ Addition
- ↳ Transpose

2D Array

arr	0 (column)	1	2
	00	01	02
0 (Rows)	2	4	6
	10	11	12
1	8	10	12
	20	21	22
2	14	16	19

16 = arr(2)(1)

arr(rowNum)(colNum)

1D Array

0	1	2	3	4	5	6
2	4	6	8	10	12	14

Num of Rows = Num of cols

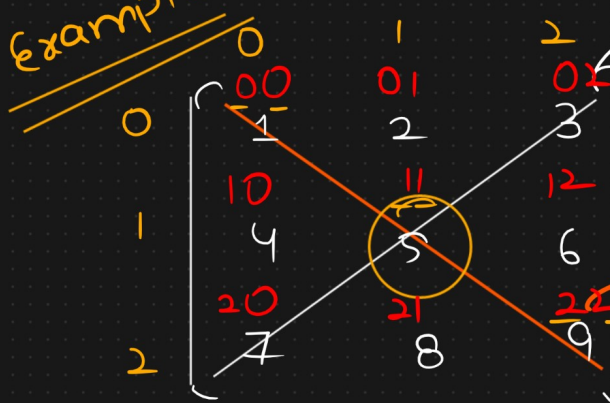
arr(5) = 12

Random Access

Square matrix

O(1)

Example



(3/2) (3/2)

arr(1)(1)

Secondary Diagonal

1 + 5 + 3

Primary Diagonal

1 + 5 + 9

= 25

3 × 3

n × n

n → odd

Element Repeat

Primary + Secondary

Repeated element

Result = FinalAnswer — arr(n/2)(n/2)

Row Position — Col. Position

n = 4

1+6+11+16

Example 2

	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	9	10	11	12
3	13	14	15	16

4+7+10+13

Even

4 × 4

n = 3

result = 0;

for(i = 0 to n-1)

result += arr(i)(i); — Primary diagonal

arr(0)(0) i = 0

Logic

result += arr(n-i-1)(i);

(0)(2) (3-2-1)(2) — Primary Diagonal

arr (2)(0)

Result = 1+5 = 6+9 = ~~15~~

15+7+5+3 ∈ 30

```
    if (n is odd){
        result = result -
                 qor(n/2) (n/2);
    }
    return result;
}
}
```

Repeated Portion

Time complexity = O(n)

Space complexity = O(1)

Power of an element

$x = 2 \qquad n = 10$

$x^n = 2^{10} = 1024$

Divide & conquer

1) MergeSort
2) QuickSort
3) BinarySearch

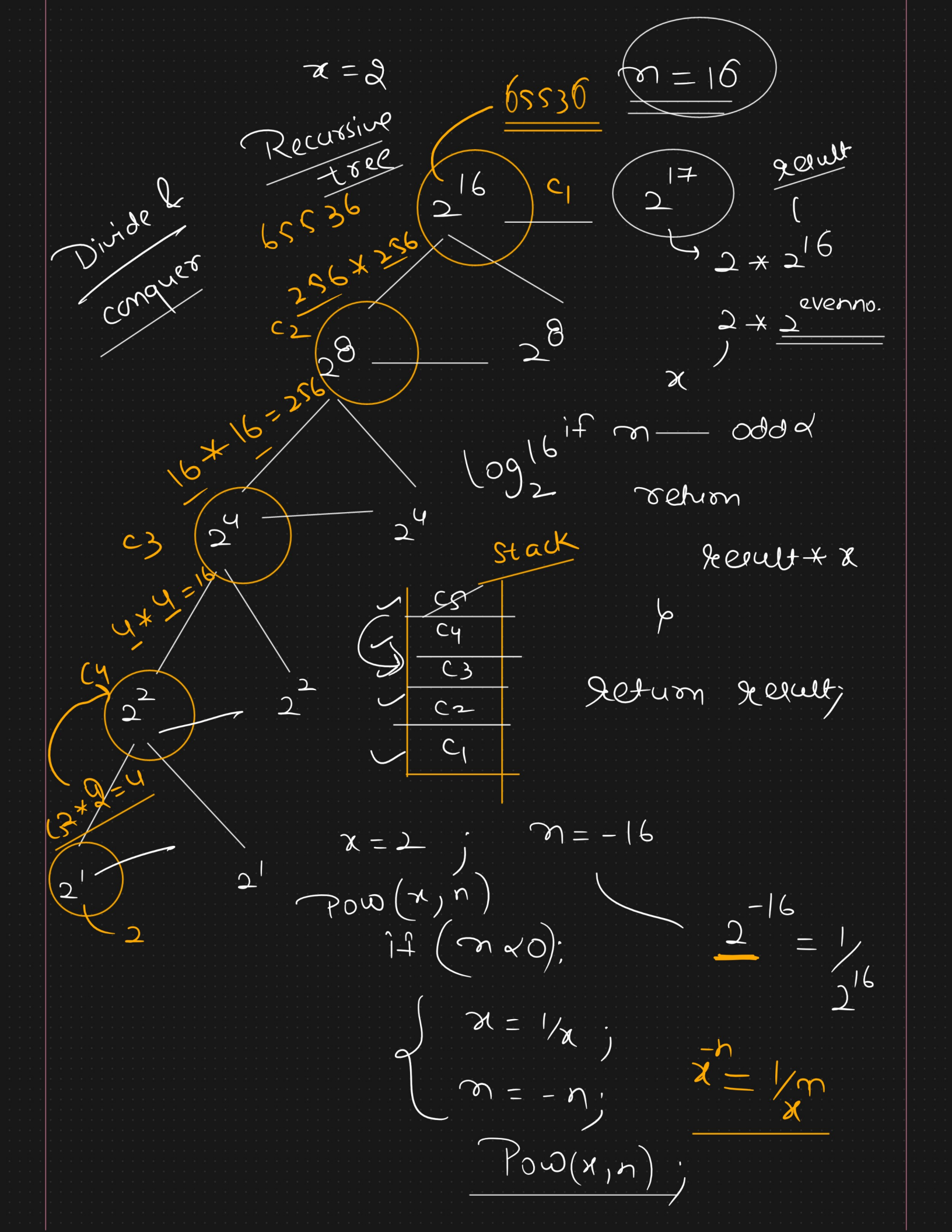
Small Problem

$x = 2 \qquad n = 1$

$x^n = x$

Big Problem → $n > 1$

↳ Divide & conquer


x = 2
n = 16
65536
Recursive tree
65536
Divide & conquer
2^{16}
c_1
2^{17}
result
$2 * 2^{16}$
256 * 256
c_2
2^8
2^8
$2 * 2^{evenno.}$
16 * 16 = 256
c_3
2^4
2^4
$\log_2^{16}$
if n — odd
return
4 * 4 = 16
c_4
2^2
2^2
stack
result * x
c_5
c_4
c_3
c_2
c_1
return result;
2 * 2 = 4
2^1
2^1
2
x = 2 ;
n = -16
Pow(x, n)
if (n < 0);
$2^{-16} = 1/2^{16}$
{ x = 1/x ;
n = -n ;
$x^{-n} = 1/x^n$
Pow(x, n) ;

Pseudocode or logical

$T(n)$

```
Power(x, n)
    ① Small Problem
    if n == 0:
        return 1
    elif n == 1:
        return x
    elif n < 0:
        x = 1/x
        n = -n
    return Power(x,n)
    else:
        mid = n // 2
        b = Power(x, mid)        T(n/2)
        result = b * b
        if n % 2 == 0:
            return result
        else
```

O(1)

Big Problem

Divide & conquer

Recursion

1) Divide & conquer
2) Combine

C

return result*x;

Recurrence Relation

Course name ⟹ Complete DSA in Python

1) Substitution
2) Recursive tree
3) Master's Theorem

Substitution / Master's

$$T(n) = T(n/2) + c$$

$$T(n) = O(\log n)$$

$O(\log_2 n)$ to

Space complexity → $O(n)$

Stack space

levels in a Recursive tree

Complexity → Big O approximation

Java, Python, Javascript, C++

Rotation of Matrix

	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9

Rotation 90° → clockwise

final output →

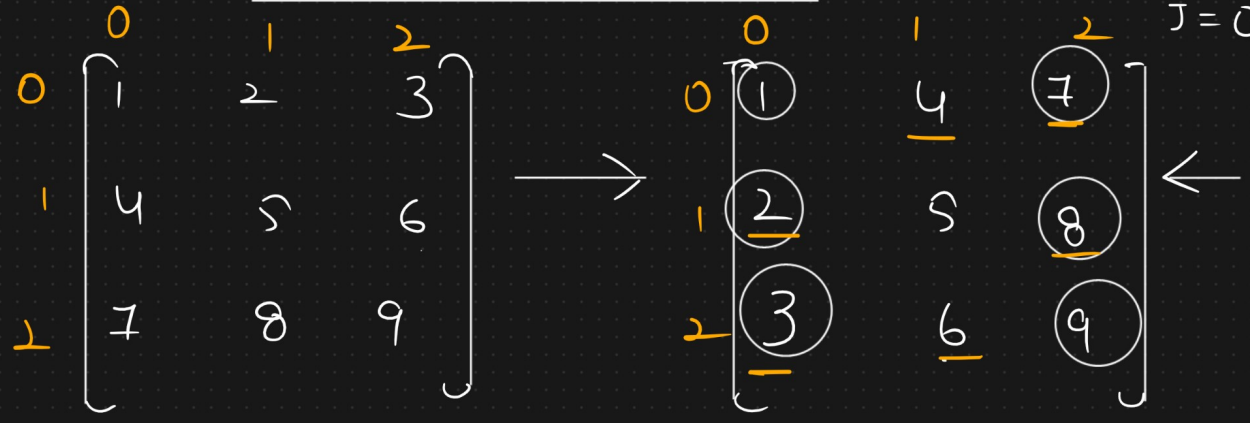
7	4	1
8	5	2
9	6	3

① Matrix transpose

$J = i + 1$

$n = 3$

$J = 0$



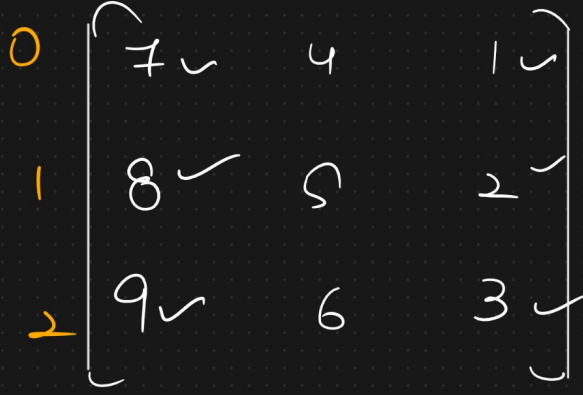
$i = 0$ $J = 1$ $J = 2$

$i = 1$

② swap b/w cols

$n = 3$

$3/2 = 1$



$i = 0$

$J = 0$