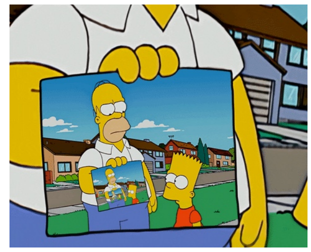
Recursion 2



AGENDA:

- Sorted Array
- Power function
- Fast exponentiation
- Find kth char

Solve classes & objects before Linked lists session.

Q1 Given an array, check if it sorted using a recursive function

Example

1) 2 5 9 10 10 13 ⇒ True

2) 4 8 3 15 ⇒ False

3) -3 0 6 10 ⇒ True

4) 4 4 4 4 4 ⇒ True

N = 5

0	1	2	3	4
2	5	9	10	10

arr[i] <= arr[i+1]

and

for i → [0, n-2]

if arr[i] > arr[i+1]

return false

checkSorted(A, i+1)

Assumption

checkSorted (A, i) check if the array from index i is sorted or not.

```
bool checkSorted (int []A, int i) {
    if( i == A.len - 1)
        return true

    if( arr[i] <= arr[i+1])
        return checkSorted(A, i+1)

    return false
}
```

← Base Case

← Main Logic

Q2 Implement power function using recursion.
Given a, n compute a^n . n >= 0.

$a = 3$, $n = 2$ $\Rightarrow$ $3^2 = 9$

$a = 2$, $n = 4$ $\Rightarrow$ $2^4 = 16$

$$a^n = \underbrace{a \times \overbrace{a \times a \times a \times \ldots \ldots a}^{a^{n-1}}}_{n \text{ times}}$$

$$a^n = a \times \underbrace{a^{n-1}}_{\text{Subproblem}}$$

$\Rightarrow$ pow(a, n) = a × pow(a, n-1)

$a^1 = a$

$a^0 = 1$ ← Base Case

Assumption – pow(a,n) gives a^n.

```
int pow (int a, int n) {
    if (n==0)          ← Base Case
        return 1

    return a * pow(a,n-1)     ← Main Logic
}
```

Tower of Hanoi

↳ Iterative – 300-400 lines

↳ Recursive – 5 lines

Fast Exponentiation

Given a,n. Compute a^n

$$a^n = a \times a^{n-1}$$

$$a^{10} = a \times a^9$$

Even $$a^{10} = a^5 \times a^5 = (a^5)^2$$

Odd $$a^{11} = a^5 \times a^5 \times a = (a^5)^2 \times a$$

Even $$a^{14} = a^7 \times a^7 = (a^7)^2$$

Odd $$a^{19} = a^9 \times a^9 \times a = (a^9)^2 \times a$$

$$\frac{10}{2} = 5, \quad \frac{11}{2} = 5, \quad \frac{14}{2} = 7, \quad \frac{19}{2} = 9$$

$$a^n = \begin{cases} \text{if } n \text{ is even} \rightarrow \underline{a^{\frac{n}{2}}} \times \underline{a^{\frac{n}{2}}} \\ \text{if } n \text{ is odd} \rightarrow \underline{a^{\frac{n}{2}}} \times \underline{a^{\frac{n}{2}}} \times a \end{cases}$$

Assumption - Same

```
int pow( int a, int n) {
    if ( n==0)
        return 1                                  Base Case

                                                  ↙ Main Logic
    if ( n is even)
        return pow(a, n/2) * pow(a, n/2)
    else
        return pow(a, n/2) * pow(a, n/2) * a
}
```

```
int pow( int a, int n) {
    if ( n==0)
1       return 1

    int p = power(a, n/2)
```

if (n is even)

2 return p * p

else

3 return p * p * a

}

---

a^{10}

TC: $O(\log_2 N)$

$1 <= a <= 10^5$

$1 <= N <= 10^{12}$

pow(a, 10) {

// N=10

p = pow(a, 5) $p = a^5$

ret p*p

a^{10}

}

pow(a, 5) {

// N=5

p = pow(a, 2) $p = a^2$

ret p*p *a

a^5

}

```
pow(a, 2) {
    // N=2
    p = pow(a,1)        p=a
    ret p*p
        a*a
}

pow(a,1) {
    // N=1
                 p=1
    p = pow(a, 0)
    ret p*p*a
        1*1*a
}

pow(a,0) {
    // N=0
    ret 1
}
```

$\boxed{a**n}$ → fast Exponentiation

Fast Exponentiation with modulo

Given a, n, m. Compute $a^n \% m$

Constraints

$1 <= a <= 10^5$

$0 <= n <= 10^6$

$1 <= m <= 10^9$

Largest value of a^n

$$(10^5)^{10^6} = (10^5)^{1000000} = 10^{50,00,000}$$

Examples

$(a^n) \% m$

a	n	m
2	5	5
3	4	7

$\Rightarrow (2^5) \% 5 = 32 \% 5$

$= 2$

$\Rightarrow (3^4) \% 7 = 81 \% 7$

$= 4$

$$\underline{a} \qquad \underline{n} \qquad \underline{m}$$

$$3 \qquad 4 \qquad 7$$

$$\left((a \% m)^n \right) \% m \rightarrow \left((\underbrace{3 \% 7}_{})^4 \right) \% 7$$

3

$\uparrow$

$\rightarrow$

$$(a^n) \% m = (\underbrace{a \times a \times a \times a \ldots \ldots a}_{n \text{ times}}) \% m$$

$$a^n \% m = \left(a^{\frac{n}{2}} \times a^{\frac{n}{2}}\right) \% m \quad \leftarrow n \text{ is even}$$

Modulo Multiplication Property

(a * b) % m = (a % m * b % m) % m

Assumption - pow(a,n,m) gives a^n % m

```
int pow( int a, int n, int m) {
    if( n==0 )                          <- Base Case
        return 1

    int p = pow( a, n/2, m)             <- (a^(n/2)) % m
                                               ↓
                                            [0, m-1]
    if ( n is even)
                  [0,m-1]     [0,m-1]
        return ( p     *      p ) % m

    else
                [0,m-1]  [0,m-1]     <m
        return ((p   *    p)%m  * a ) % m

}
```

TC: $O(\log_2 N)$

Analysis in next class

Q3 Find kth character

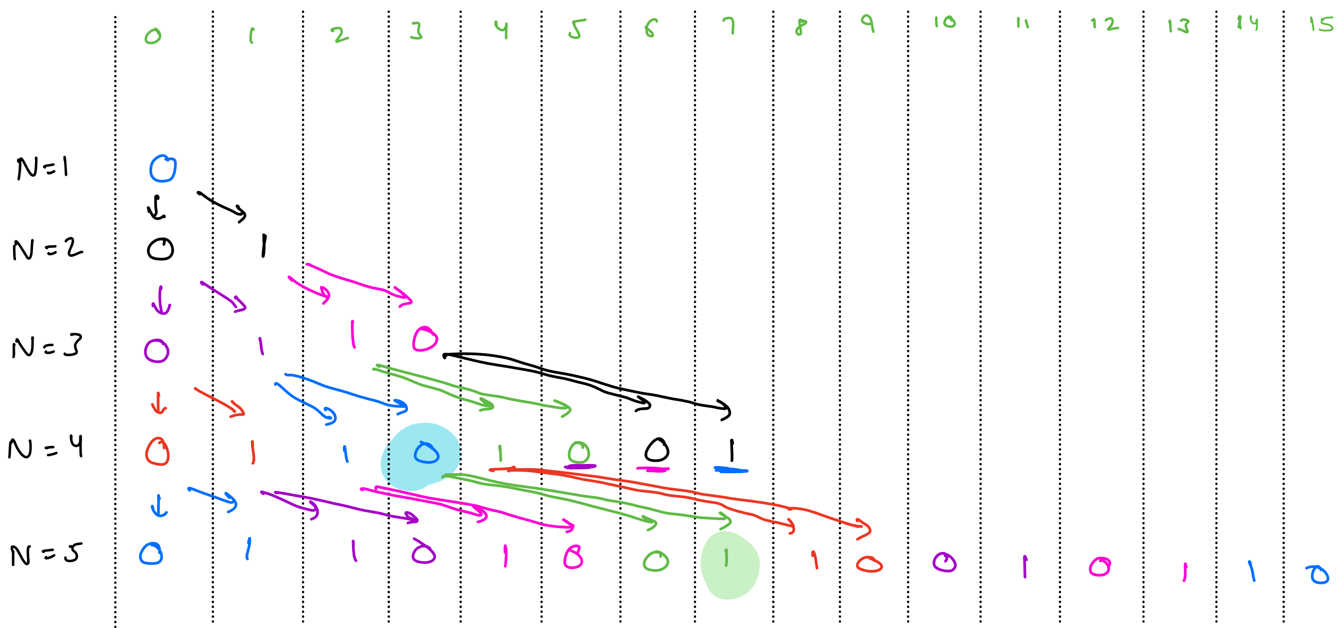
Each row is generated by replacing all elements of the previous row such that,

0 -> 01

1 -> 10

We always start with a 0 for N=1.

Given N, k. Find the kth element in Nth row.



N	k		
5	10	⇒	0
4	3	⇒	0

5 7 $\Rightarrow$ 1

4 10 $\Rightarrow$ Invalid Input

Solve it on your own

HW

$1 <= N <= 10^5$

$1 <= k <= 10^9$

In your HW - it follows

1 based indexing.

Q4 Given an array and a target value, count no of occurrences of target in the array.

Example

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

target = 2

Ans = 2

Count frequency of target

```
c = 0
for i -> [0, n-1]
    if A[i] == target :
        c++
return c
```

Subproblem

remaining Count = Count(A, target, i+1)

```
if A[i] == target
    1 + remaining Count
else :
    remaing Count
```

Do it recursively

<u>Assumption</u> - count (A, target, i) will return the frequency count in array A from index i.

```
                                             0
                                             ↑
int    count ( int [] A, int   target , int  i ) {
        if ( i == n)                        <-- Base Case
              return   0

       remaining Count = count (A, target, i+1)
       if  A[i] == target                         }
            return  1  +  remaining Count         }  Main
                                                  }
       else :                                     }  Logic
            return  remaing Count                 }
}
```

Doubts

Thank You

```
int count ( int [] A, int target , int i ) {
    if( i < A.length ) {
        remaining Count = count (A, target, i+1)
        if A[i] = = target
            return 1 + remaining Count
        else :
            return remaing Count
    }
    return 0
}
```

In python, you have a recursion limit set by default.

→ 1000

```python
import sys
sys.setrecursionlimit(20000)
```

Advanced Batch → Satya Sai

Good Night

Thank You

Wednesday