

# Design and Analysis of Algorithms

## L12: Divide and Conquer

Dr. Ram P Rustagi  
Sem IV (2019-H1)  
Dept of CSE, KSIT/KSSEM  
[rprustagi@ksit.edu.in](mailto:rprustagi@ksit.edu.in)

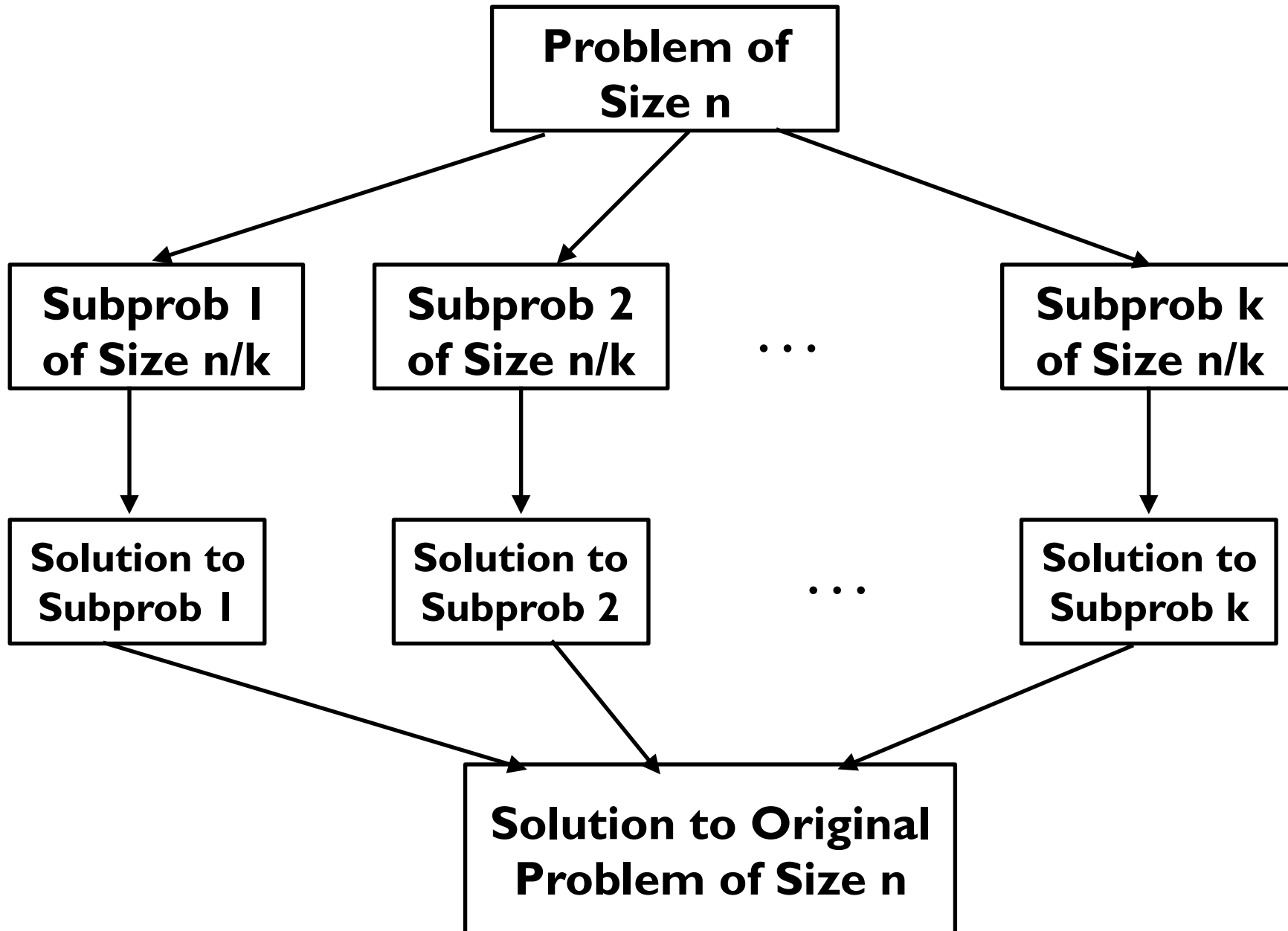
# Resources

- Text book 2: Horowitz
- Text book 1: Levitin
- <https://visualgo.net/en>

# Divide and Conquer Algo

- Divide (break) the problem (size  $n$ ) into similar sub problems
  - Size of sub problems should be some factor of original e.g.  $n/c$ 
    - When small enough, solve by brute force
- Conquer (Solve) the sub-problem
  - Use recursion to solve small problem
- Combine (Merge) the solution of sub-parts
- The cost is
  - cost of breaking
  - cost of solving subproblem
  - cost of combining

# Divide and Conquer Approach

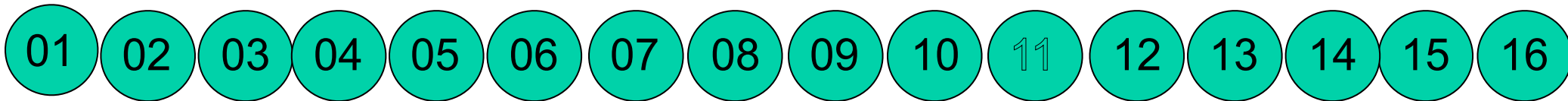


# Divide and Conquer Examples

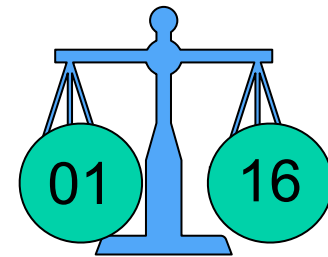
- Sorting and Searching
- Binary Tree traversals
- Binary search
- Multiplication of large numbers (Karatsuba Algo)
- Matrix multiplication - Strassen's algorithm
- Closest pair problem
- Convex Hull problem

# Simple Example

- Given 16 balls with one defective (say lighter)
  - Identify the defective ball.

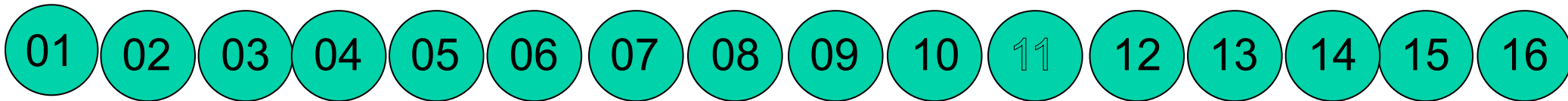


- Solution 1:
  - Compare 1 with 2
  - Compare 1 with 3
  - :
  - Compare 1 with 16
- Time taken:
  - 15 comparisons (worst case)

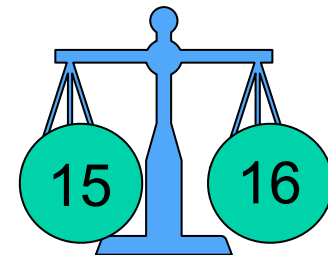
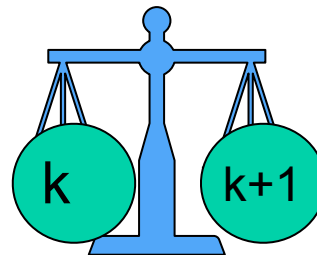


# Simple Example

- Given 16 balls with one defective (say lighter)
  - Identify the defective ball.

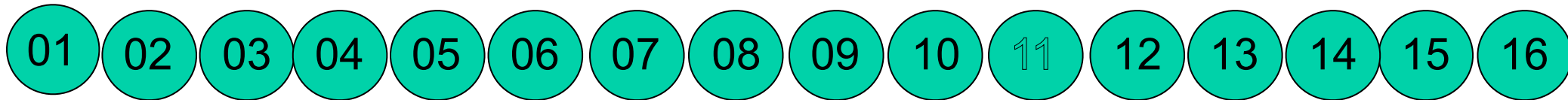


- Solution 2:
  - Compare 1 with 2
  - Compare 3 with 4
  - :
  - Compare 15 with 16
- Time taken:
  - 8 comparisons (worst case)



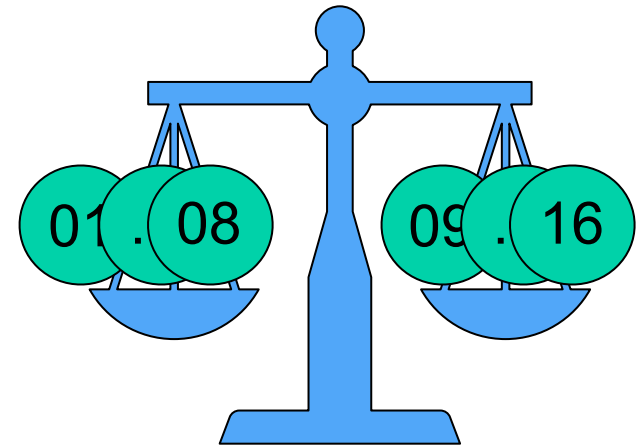
# Simple Example

- Given 16 balls with one defective (say lighter)
  - Identify the defective ball.



Soltion3: Divide and Conquer

- Divide into 2 sets, each of 8 balls
- Compare 1-8 with 9-16, and divide the lighter set into two parts each of 4.
- :
- Continue the process till lighter ball is found

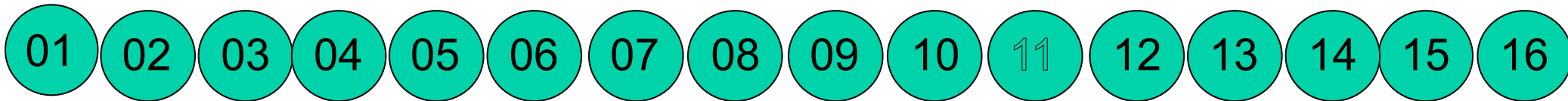


Time taken: 4 comparisons ( $\log_2 16$ )



# Simple Example

- Given 16 balls with one defective (say lighter)
  - Identify the defective ball.



- Solution 3: Time complexity

$$\begin{aligned}T(n) &= T(n/2) + 1 \quad \#1 \text{ comparison reduces it by half} \\&= T(n/4) + 1 + 1 = T(n/2^2) + 2 \\&= T(n/2^3) + 3 \\&\vdots \\&= T(n/2^k) + k \\&= \log_2 n\end{aligned}$$

# Divide & Conquer: Control Abstraction

```
Type DAndC (P) {  
    if Small (P)  
        return S (P)  
    else {  
        Divide P into smaller sets P1, ..., Pk  
        Apply DAndC to each subproblem  
        return Combine (DAndC (P1) ..., DAndC (Pk) )  
    }  
}
```

# Divide and Conquer: Recurrence Relation

$$T(n) = \begin{cases} g(n) & n \text{ small} \\ T(n_1) + T(n_2) + \dots + T(n_k) + f(n) & \text{otherwise} \end{cases}$$

- $T(n)$  : time complexity for a problem of input size  $n$
- $g(n)$  : time complexity for solving directly for small inputs
- $f(n)$  : Time complexity for dividing the problem into  $k$  subproblems and combining again from the solutions of  $k$  sub problems.
- $k$  would vary depending upon the problem
  - Generally,  $n_1 = n_2 = \dots = n_k$
  - Assuming  $a$  instances, each of size  $n/b$

$$T(n) = \begin{cases} T(1) & n = 1 \\ aT(n/b) + f(n) & n > 1 \end{cases}$$

# Solving Recurrence Relation

$$T(n) = aT(n/b) + f(n)$$

- Let  $n=b^k$ , then

$$T(b^k) = aT(b^{k-1}) + f(b^k)$$

$$= a[aT(b^{k-2}) + f(b^{k-1})] + f(b^k)$$

$$= a^2T(b^{k-2}) + af(b^{k-1}) + f(b^k)$$

$$= a^3T(b^{k-3}) + a^2f(b^{k-2}) + af(b^{k-1}) + a^0f(b^k)$$

⋮

$$= a^kT(b^{k-k}) + a^{k-1}f(b^{k-(k-1)}) + a^2f(b^{k-2}) + af(b^{k-1}) + a^0f(b^k)$$

$$= a^kT(1) + a^{k-1}f(b^1) + a^{k-2}f(b^2) + \dots + a^0f(b^k)$$

$$= a^k[T(1) + f(b^1)/a^1 + f(b^2)/a^2 + \dots + f(b^k)/a^k]$$

# Solving Recurrence Relation

$$T(n) = aT(n/b) + f(n)$$

$$T(b^k) = aT(b^{k-1}) + f(b^k)$$

$$= a^k[T(1) + f(b^1)/a^1 + f(b^2)/a^2 + \dots + f(b^k)/a^k]$$

$$= a^k[T(1) + \sum_{j=1}^k \frac{f(b^j)}{a^j}]$$

- Thus,  $T(n)$  depends upon  $a$ ,  $b$ , and  $f()$

**As  $n=b^k$ , then  $k=\log_b n$ , thus**

**$a^k = a^{\log_b n} = n^{\log_b a}$ , the recursion equation becomes**

$$T(n) = n^{\log_b a} [T(1) + \sum_{j=1}^{\log_b n} \frac{f(b^j)}{a^j}] \quad (1)$$

# Recurrence Relation: Examples

- **Example 01:**  $a=2, b=2, T(1)=1, f(n)=n$

$$\begin{aligned}
 T(n) &= 2T(n/2) + n \\
 &= 2[2T(n/2^2) + n/2] = 2^2T(n/2^2) + n + n \\
 &= 2^3T(n/2^3) + n + n + n \\
 &= 2^kT(1) + n + \dots + n \quad (\log_2 n \text{ times}) \\
 &= 2^k + n \cdot \log_2 n \\
 &= n + n \cdot (\log_2 n) \\
 &= n + n \log_2 n = \Theta(n \log_2 n)
 \end{aligned}$$

Using the eqn (I)

$$\log_b a = \log_2 2 = 1, \quad b/a = 1 \rightarrow f(b^j)/a^j = b^j/a^j = 1$$

$$\begin{aligned}
 T(n) &= n^{\log_b a} \left[ T(1) + \sum_{j=1}^{\log_b n} \frac{f(b^j)}{a^j} \right] \\
 &= n [1 + (1 + 1 + \dots (\log_2 n \text{ times}) + 1)] = n \log_2 n \\
 &= \Theta(n \log_2 n)
 \end{aligned}$$

# Recurrence Relation: Examples

- **Example 02:**  $a=9$ ,  $b=3$ ,  $T(1)=4$ ,  $f(n)=4n^6$

Given

$$\log_b a = \log_3 9 = 2,$$

$$f(b^j) / a^j = 4b^{6j} / a^j = 4 * 3^{6j} / 3^{2j} = 4 * 3^{4j}$$

$$T(n) = n^{\log_b a} \left[ T(1) + \sum_{j=1}^{\log_b n} \frac{f(b^j)}{a^j} \right]$$

$$= n^2 [ 4 + (4 * 3^4 + 4 * 3^{4*2} + \dots + 4 * 3^{4 * \log_3 n}) ]$$

$$= n^2 * 4 (3^{4 * (\log_3 n + 1)} - 1) / (3^4 - 1)$$

$$= c * n^2 * 3^{4 * (\log_3 n)} + d = c * n^2 * n^4 + d$$

$$= \Theta(n^6)$$

# Fun Exercise of Game of 128 numbers

- A practical fun example of Data structures and Algorithm

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64
65	66	67	68	69	70	71	72
73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88
89	90	91	92	93	94	95	96
97	98	99	100	101	102	103	104
105	106	107	108	109	110	111	112
113	114	115	116	117	118	119	120
121	122	123	124	125	126	127	128

## Game:

- . Go thru a set of cards
- . Say Y/N if present or not
- . You will get your number graphically displayed to you

## Q?:

Which algorithm we are discussing?

**Aim:** Can we find more such examples



# Game of 128 numbers - b

1	2	3	4	5	6	7	8	<b>X</b>
9	10	11	12	13	14	15	16	
17	18	19	20	21	22	23	24	
25	26	27	28	29	30	31	32	
33	34	35	36	37	38	39	40	
41	42	43	44	45	46	47	48	
49	50	51	52	53	54	55	56	
57	58	59	60	61	62	63	64	

# Game of 128 numbers - c

1	2	3	4	9	10	11	12	<b>X</b>	
17	18	19	20	25	26	27	28		
33	34	35	36	41	42	43	44		
49	50	51	52	57	58	59	60		
69	70	71	72	77	78	79	80		
85	86	87	88	93	94	95	96		
101	102	103	104	109	110	111	112		
117	118	119	120	125	126	127	128		
								<b>X</b>	

# Game of 128 numbers - d

1	2	3	4	5	6	7	8	x
9	10	11	12	13	14	15	16	
17	18	19	20	21	22	23	24	
25	26	27	28	29	30	31	32	
65	66	67	68	69	70	71	72	
73	74	75	76	77	78	79	80	
81	82	83	84	85	86	87	88	
89	90	91	92	93	94	95	96	
								x

# Game of 128 numbers - d

1	2	5	6	9	10	13	14	x		x	
17	18	21	22	25	26	29	30				
33	34	37	38	41	42	45	46				
49	50	53	54	57	58	61	62				
67	68	71	72	75	76	79	80				
83	84	87	88	91	92	95	96				
99	100	103	104	107	108	111	112				
115	116	119	120	123	124	127	128				
									x		x

# Exercise G

- Exercise G
  - Work out the remaining 3 cards

# Summary: Divide and Conquer

- Break the problem into smaller subsets
  - By a factor  $c$  i.e.  $n \rightarrow n/c$
- Conquer (Solve) the sub-problem
- Combine (Merge) the solution of sub-parts
- Example cases
  - Sorting and Searching
  - Binary Tree traversals
  - Binary search
  - Multiplication of large numbers (Karatsuba Algo)
  - Matrix multiplication - Strassen's algorithm
  - Closest pair problem
  - Convex Hull problem

# Summary

- Divide and Conquer approach
- Cost efficiency:
  - Define recurrence relation
  - Solve the recurrence equation
- Example of binary search (visual)