

## Lecture #1:

### Types of Algorithms based on methods used:

1. Divide and Conquer
2. Greedy Methods
3. Dynamic Programming
4. Incremental Algorithms (also known as inductive algorithms)
5. Improvement Algorithms

### Problems discussed so far:

- Binary Search: Effort:  $\Theta(\log_2 n)$   
Given a sorted array of  $n$  numbers,  $A[1, 2, \dots, n]$ , and a number  $x$ .  
Permitted operations: Given two numbers,  $x$  and  $y$  check which of the following holds: (i)  $x = y$ ; (ii)  $x < y$ ; or (iii)  $x > y$   
Question: Is  $x = A[j]$  for some  $j$  between 1 and  $n$ ? If the answer is **yes**, output  $j$ ; else output **no**.
- This is an example of divide and conquer where we get one subproblem of half the size.  
Recursion encountered:  $t(n) = 1 + t(\lfloor \frac{n}{2} \rfloor)$
- Tower of Hanoi/Brahma: Effort:  $2^m - 1$   
Given three rods numbered 1, 2, and 3 with  $m$  rings on rod 1 and none on rods 2 and 3. The rings on rod 1 are stacked in order of size with the smallest on top. (like the toy children play with)  
Permitted operations: Move the top ring on any rod to any other rod provided that at no time a larger ring is on top of a smaller ring on any rod.  
Question: Want to move the rings from rod 1 to rod 2 with minimum number of moves.  
Recursion:  $t(n) = 1 + 2t(n - 1)$
- Insertion Sort: (CLR page 3 has a pseudocode)  
Effort:  $\frac{n(n-1)}{2}$  in the worst case;  $(n - 1)$  in the best case.  
If we use Binary search in determining the location to insert, the worst case effort becomes  $\sum_{i=1}^{n-1} \log_2 i$  and this is  $\Theta(n \log_2 n)$

- Merge Sort:: (CLR pages 12-15):

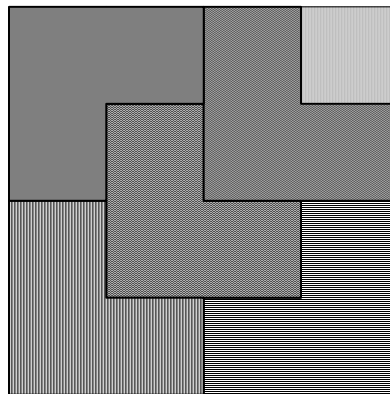
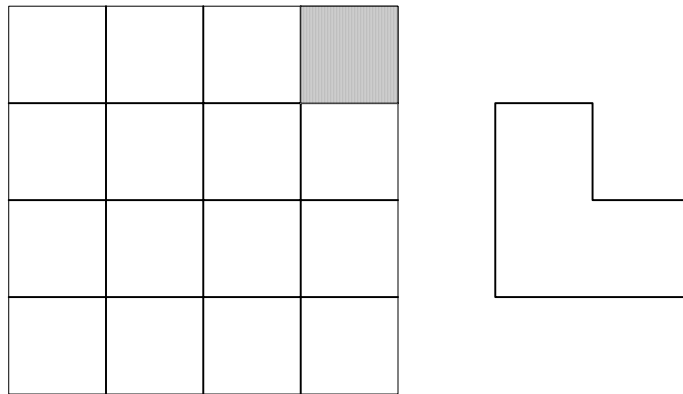
Effort:  $\Theta(n \log_2 n)$  in the worst case. This is an example of divide and conquer where we get two subproblems of half the size.

Recursion:  $t(n) = 2t(\frac{n}{2}) + \Theta(n)$

- Tiling:

Given a square board of size  $2^k$  divided into  $2^{2k}$  equal size squares of size 1 (like in a chess/checker board). An arbitrary square is declared special. Also given *tiles* which are squares of size 2 with one square removed.

Question: Can we cover all the squares in the bigger board (except the special square) with tiles of the above type? If so, provide a method to do this. See the picture below for the case with  $k = 2$ :



This is also an example of Divide and conquer. Here we get 4 subproblems of size  $2^{k-1}$ .