Lecture #1:

Characterization of types of algorithms: Off-line vs on-line algorithms; deterministic vs randomized algorithms; exact vs approximation algorithms; sequential vs parallel algorithms; centralized vs distributed algorithms. **This course primarily deals with off-line, deterministic, exact, sequential, centralized algorithms.**

What bothers most of about this course (and similar other courses) is the fact that there are no algorithms to show how algorithms are designed. This area is somewhat similar to solving mathematics puzzles. Results are not guaranteed based on effort or time spent.

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
6. Parametric algorithms
7. Duality based algorithms

Example 1 *Binary Search: Worst case effort:* $\Theta(\log_2 n)$

- Given a sorted array of n numbers, $A[1, 2, ..., n]$, 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** (some times we may give an interval $[i, i + 1]$ such that $A[i] < x < A[i + 1]$).

  BINARY-SEARCH$(A[i, j], x)$

  if $j < i$ STOP.

  $k \leftarrow \lfloor \frac{i+j}{2} \rfloor$

  if $A[k] = x$, STOP

  else if $A[k] < x$, do BINARY-SEARCH$(A[i, k], x)$

  else do BINARY-SEARCH$(A[k + 1, j], x)$

  This is an example of divide and conquer where we get one subproblem of half the size.

  Recursion encountered: $t(n) = 1 + t(\lfloor \frac{n}{2} \rfloor)$

Example 2 *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: $t(n) = 1 + 2t(n-1)$

Example 3 *Insertion Sort: (CLR 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 $\Theta(n \log_2 n)$

Example 4 *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.

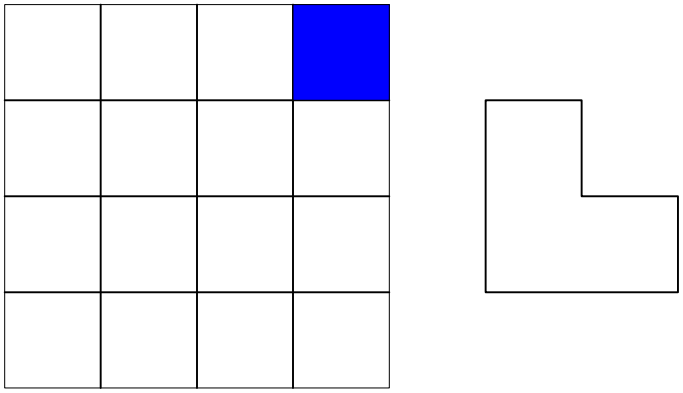
  Recursion: $t(n) = 2t(\frac{n}{2}) + \Theta(n)$

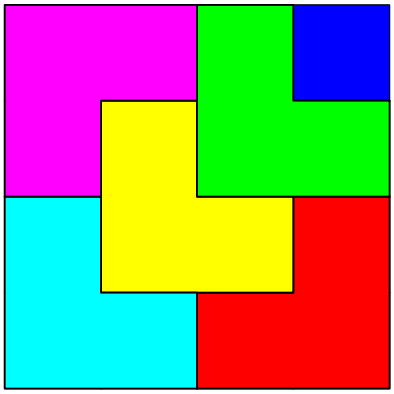
Example 5 *Tiling:*

- 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}. But three of the problems are the same.