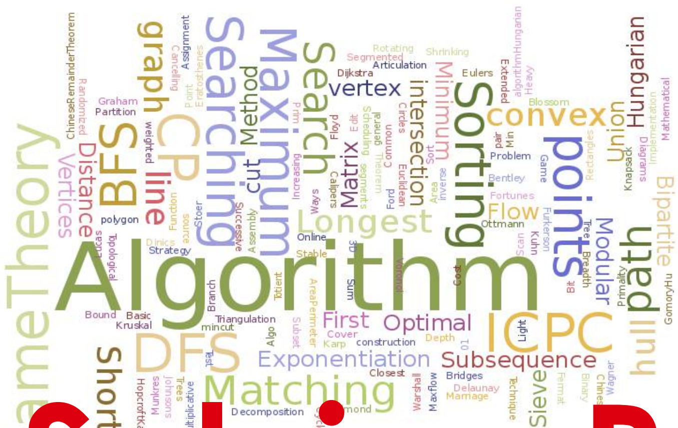
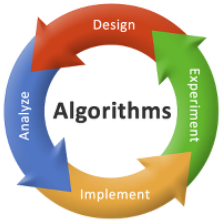


Solving Recurrence Relations

Course: Algorithms



Faculty: Dr. Rajendra Prasath

Autumn 2018

Solving Recurrence Relations

This class covers different methods to solve recurrence relations. This lecture illustrates a few methods of **solving recurrence relations** with a few selected examples and their analysis as well

Recap: Complexity Spectrum

Function	Common name
$n!$	factorial
2^n	exponential
$n^d,\ d > 3$	polynomial
n^3	cubic
n^2	quadratic
$n\sqrt{n}$	
$n \log n$	quasi-linear
n	linear
$\sqrt{n}$	root - n
$\log n$	logarithmic
1	constant

Recap: Handling Recursions?

- How to we define a recursive approach?
  - In terms of the input size n (smaller to bigger)
  - The rate of change of n (growing / decaying)

- Two Steps:
  - Basic Steps (terminating step !!)
  - Recursive Steps (Repeating Step)

- Important Points
  - Optimize Recursive Calls
  - Avoid Inefficient Recursions as $n \to \infty$
  - Understand the way recursion works for the given problem

Recap: Analysis of Recursion

```
void f(int n) {
    if(n > 0) {
        DoSomething(n); // O(n)

        f(n/2);         → O(n)

        f(n/2);         f(n/2); → O(n logn)

        f(n - 1);       → O(n^2)

        f(n - 1);       f(n - 1); → O(2^n)
    }
}
```

Recursion – The Closed Form

- How to get a closed form of a recurrence relation?

$$a_0 = 4$$

$$a_n = a_{n-1} - n$$

- The Closed form solution for T(n) is

$$a_n = 4 - n(n+1)/2$$

- How to find such Closed Form solutions?

Closed Form – Example 1

- Consider the following recurrence relation:

$$T(n) = \begin{cases} 5, & \text{if } n <= 2 \\ T(n-1) + n, & \text{otherwise} \end{cases}$$

- Find the Closed form solution for T(n)?

Example 1 - Solution

$T(k + 1) = T(k) + k + 1$
$T(k + 2) = T(k + 1) + k + 2 = T(k) + k + 1 + k + 2$
$T(k + m) = T(k) + k + 1 + k + 2 + \ldots + k + m$
$\quad = T(k) + mk + 1 + 2 + \ldots + m = T(k) + mk + m*(m + 1) / 2$
$\rightarrow T(k + m) = T(k) + mk + m * (m + 1) / 2$

Let $k = 2$:
$T(m + 2) = T(2) + 2m + m * (1 + m) / 2$
$\quad = 5 + 2m + m * (m + 1) / 2$

Finally, let $m + 2 = n$ or $m = n – 2$:
$T(n) = 5 + 2 * (n - 2) + (n - 1) * (n - 2) / 2 = n * (n + 1) / 2 + 2$

We have
$T(n) = n * (n + 1) / 2 + 2$, if $n > 2$
$T(n) = 5$ if $n <= 2$

Solving Recursive Relations

- How to find the closed form solution for T(n)

- Several Methods

  - Substitution Method
  - Recursive Tree based Method
  - Iterative Method
  - The Characteristic Root Method

and so on.

Substitution Method

- The Basic Idea
  - Make a guess of a possible solution
  - Use Induction to prove it.
- Look at the following Recursive Definition:

$$T(n) = \begin{cases} 1, & \text{if } n = 1 \\ 2T(n/2) + n, & \text{for all } n > 1 \end{cases}$$

- How to solve this recurrence relation?

Substitution Method (cont.)

- Solving the following recursive relation:

$$T(n) = 1, \quad \text{if } n = 1$$
$$= 2T(n/2) + n, \quad \text{for all } n > 1$$

- Solution:

1) $T(n) = n \log n + n$

2) Proof by Induction

Basis: if $n = 1$, => $1 \log 1 = 1$ → **$T(n) = 1$, true for $n = 1$**

Induction Step:

$T(n) = 2T(n/2) + n = 2((n/2) * \log (n/2)+ n/2)+ n$

$= n \log (n/2) + n + n$

$= n (\log n – \log 2) + n + n = n \log n – n + n + n$

$T(n) = n \log n + n$ is true for for all $n > 1$

Iteration Method

- Problem: Assume that a recursive expression a_n is given with initial conditions: a_0

- Can we express a_n without depending on previous terms ??

- Example - 1:

  $a_n = 2\, a_{n-1}$ for $n \geq 1$ with initial condition $a_0 = 1$

- Solution:

$$a_n = 2^n \quad \text{(how??)}$$

Iteration Method (cont.)

- Example - 2: The Growth of Tiger Population

- Tiger Population a_n at time n
- Initial Condition: $a_0 = 100$
- Tiger population increases from time n – 1 to time n is 20%. Then the recursive function is modeled as follows:

$$a_n - a_{n-1} = 0.2\, a_{n-1}$$

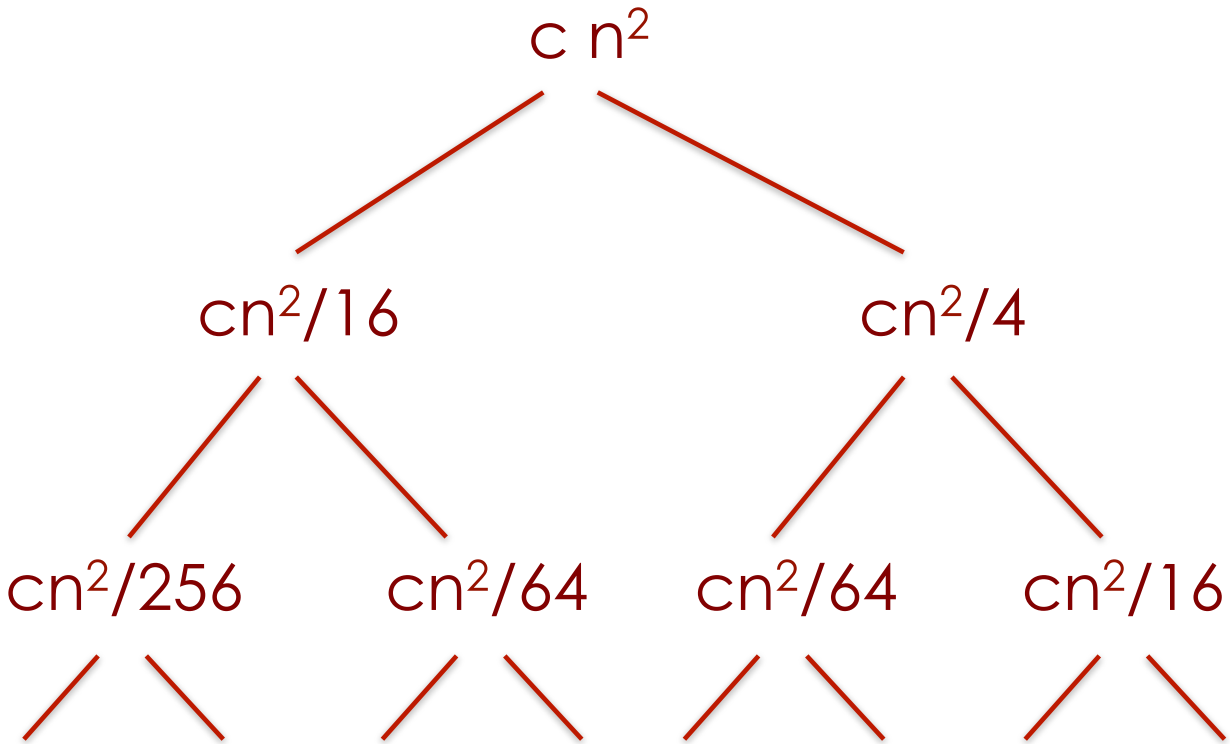
$$\Rightarrow a_n = 1.2\, a_{n-1}$$

- Solution:

$$a_n = 100\,(1.2)^n \quad \text{(how??)}$$

Recurrence Tree - Example

- $T(n) = T(n/4) + T(n/2) + cn^2$
- Solution:



Recurrence Tree - Solution

- $T(n) = T(n/4) + T(n/2) + cn^2$
- Solution: Calculate T(n)?
  - calculate sum of tree nodes level by level
  - The following series is obtained from the tree:
    $T(n) = c(n^2 + 5(n^2)/16 + 25(n^2)/256) + \ldots$
  - This is a geometric progression with ratio 5/16
  - Upper bound:
    Sum $= n^2/(1 - 5/16)$ which is $O(n^2)$

Recurrence Tree Method

- Calculate the time taken by every level of the tree
- Sum the work done at all levels
- Pattern: an arithmetic of a geometric series

- Example: $T(n) = T(n/4) + T(n/2) + cn^2$

- How to solve this recurrence relation?

Master Method

- Consider a recurrence of the form:

  $T(n) = aT(n/b) + f(n)$ where $a \geq 1$ and $b > 1$

- Recursive Tree based approach
- The running time is influenced by:
  - cost at **leaf nodes:**

    If $f(n) = \Theta(n^c)$ where $c < \text{Log}_b a$ then $T(n) = \Theta(n^{\text{Log}_b a})$

  - cost **evenly distributed** throughout the tree:

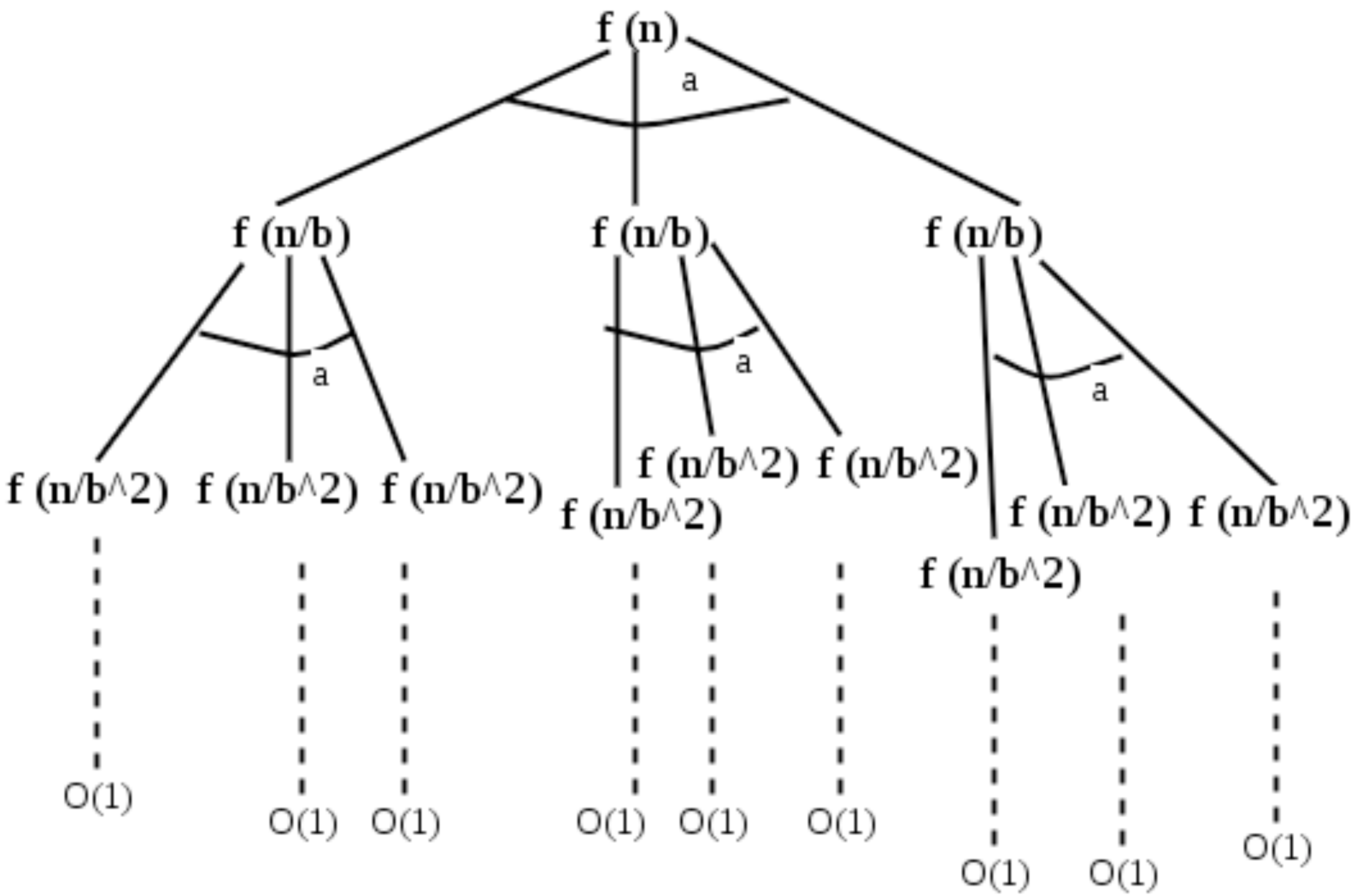
    If $f(n) = \Theta(n^c)$ where $c = \text{Log}_b a$ then $T(n) = \Theta(n^c \text{Log } n)$

  - cost at **root nodes**:

    If $f(n) = \Theta(n^c)$ where $c > \text{Log}_b a$ then $T(n) = \Theta(f(n))$

An Illustration

- Height of recurrence tree is $Log_b n$



Master Theorem: Formal Defn.

- For any recurrence relation of the form

  $T(n) = a\,T(n/b) + cn^k$, $T(1) = c$, the following relationships hold

$$\mathbf{T}(n) = \begin{cases} \Theta(n^{\log_b a}) & \textit{if } a > b^k \\ \Theta(n^k \log n) & \textit{if } a = b^k \\ \Theta(n^k) & \textit{if } a < b^k \end{cases}$$

- Apply this recurrence whenever appropriate without deriving the solution for the recurrence

Master Theorem: Example 1

$$\mathbf{T}(n) = \begin{cases} \Theta(n^{\log_b a}) & \textit{if } a > b^k \\ \Theta(n^k \log n) & \textit{if } a = b^k \\ \Theta(n^k) & \textit{if } a < b^k \end{cases}$$

- Apply the theorem to solve

$$T(n) = 3\,T(n/5) + cn^2, \quad T(1) = c$$

- Here a = 2, b = 5, c = 8 and k = 2
- Check whether $3 < 5^2$??
- As per case 3: The solution is

$$T(n) = \Theta(n^2)$$

Master Theorem: Example 2

$$\mathbf{T}(n) = \begin{cases} \Theta(n^{\log_b a}) & \textit{if } a > b^k \\ \Theta(n^k \log n) & \textit{if } a = b^k \\ \Theta(n^k) & \textit{if } a < b^k \end{cases}$$

- Apply the theorem to solve

  $T(n) = 2\,T(n/2) + n, \quad T(1) = 1$

- Here a = 2, b = 2, c = 1 and k = 1
- Can we find $2 = 2^1$??
- As per case 2: The solution is

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

Master Theorem: Example 3

$$\mathbf{T}(n) = \begin{cases} \Theta(n^{\log_b a}) & \textit{if } a > b^k \\ \Theta(n^k \log n) & \textit{if } a = b^k \\ \Theta(n^k) & \textit{if } a < b^k \end{cases}$$

- Average case Analysis of Quicksort:

$$\mathbf{T}(n) = cn + \frac{1}{n}\sum_{k=0}^{n-1}[\mathbf{T}(k) + \mathbf{T}(n-1-k)]$$

- T(0)
- Solving the above recurrence relation, we get

$$= \quad 2c\left(1 + (n+2)\left(\frac{1}{n+1} + \frac{1}{n} + \cdots + \frac{1}{2}\right)\right)$$

➔ Tight bound complexity:

T(n) = Θ (n log n)

A Few Examples

- Binary Search:
  - $T(n) = T(n/2) + \Theta(1)$
  - case 2: c is 0 and $Log_b a$ is also 0
  - The solution is $\Theta(\log n)$

- Merge Sort:
  - $T(n) = 2T(n/2) + \Theta(n)$
  - case 2: c is 1 and $Log_b a$ is also 1.
  - The solution is $\Theta(n \operatorname{Log} n)$

- Solve: $T(n) = 3\,T(n/4) + n \log n$
  - case: 1, $T(n) = \Theta(n \log n)$

- Can you find more examples yourself ??

Linear Homogeneous RR

- Examples:
  - The recurrence relation $P_n = (1.05)P_{n-1}$ is a linear homogeneous recurrence relation of degree one.
  - The recurrence relation $f_n = f_{n-1} + f_{n-2}$ is a linear homogeneous recurrence relation of degree two.
  - The recurrence relation $a_n = a_{n-5}$ is a linear homogeneous recurrence relation of degree five.

Characteristic Equation

Let $a_n = r^n$ is a solution of the linear homogeneous recurrence relation

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \ldots + c_k a_{n-k}$$

if and only if

$$r^n = c_1 r^{n-1} + c_2 r^{n-2} + \ldots + c_k r^{n-k}$$

Divide this equation by r^{n-k} and subtract the right-hand side from the left:

$$r^k - c_1 r^{k-1} - c_2 r^{k-2} - \ldots - c_{k-1} r - c_k = 0$$

This is called the **characteristic equation** of the recurrence relation

Fibonacci Series: An Example

- Problem: Give an explicit formula for the Fibonacci numbers
- Solution:
  - The Fibonacci numbers satisfy the recurrence relation $f_n = f_{n-1} + f_{n-2}$ with initial conditions $f_0 = 0$ and $f_1 = 1$.
  - The characteristic equation is $r^2 - r - 1 = 0$.
  - Its roots are

$$r_1 = \frac{1+\sqrt{5}}{2}, \quad r_2 = \frac{1-\sqrt{5}}{2}$$

Fibonacci Series: An Example

- The Fibonacci numbers are given by

$$f_n = \alpha_1 \left(\frac{1+\sqrt{5}}{2} \right)^n + \alpha_2 \left(\frac{1-\sqrt{5}}{2} \right)^n$$

- for some constants α_1 and α_2.
- We can determine values for these constants so that the sequence meets the conditions $f_0 = 0$ and $f_1 = 1$:

$$f_0 = \alpha_1 + \alpha_2 = 0$$

$$f_1 = \alpha_1 \left(\frac{1+\sqrt{5}}{2} \right) + \alpha_2 \left(\frac{1-\sqrt{5}}{2} \right) = 1$$

Fibonacci Series: (cont.)

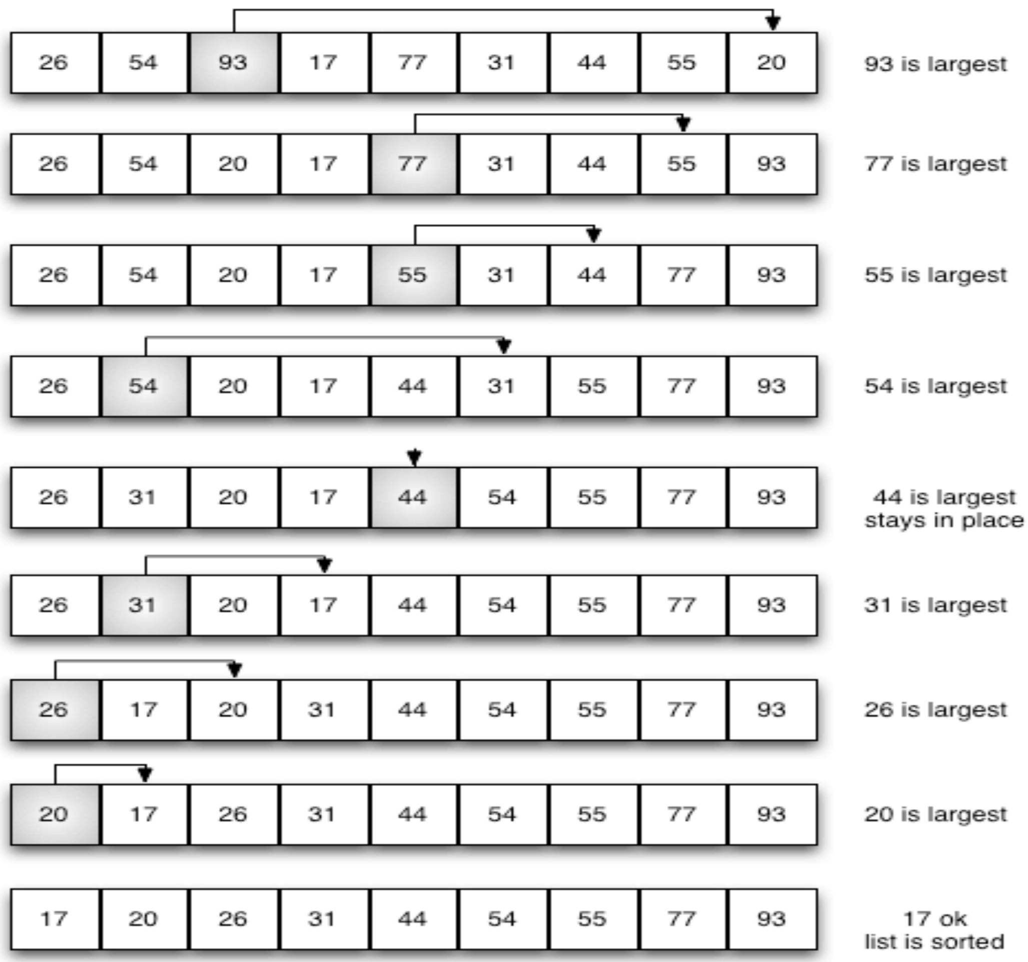
- The unique solution to this system of two equations and two variables is:

$$\alpha_1 = \frac{1}{\sqrt{5}}, \ \alpha_2 = -\frac{1}{\sqrt{5}}$$

- So finally we obtained an explicit formula for the Fibonacci numbers:

$$f_n = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^n$$

Selection Sort



- Why $\Theta(n^2)$ complexity??

Application - 1

- Selection Sorting (put the largest term at the end or the smallest term in the beginning):
  - Given a sequence of n terms b_k, $k = 1, 2, \ldots, n$ to be arranged in increasing order
  - Count the number of comparisons b_n with initial condition $b_1 = 0$
  - Obtain recursion relation

$$b_n = n - 1 + b_{n-1} \text{ for } n = 1, 2, 3, \ldots$$

  - $b_n = n(n-1)/2$ (how??)

$\rightarrow \Theta(n^2)$ complexity

Application - 2

- Binary Search Problem:
  - Search for a value in an increasing sequence. Return the index of the value, or 0 if not found.

- Initial condition $a_1 = 2$

- Recurrence relation $a_n = 1 + a_{\lfloor n/2 \rfloor}$

- Solution: $a_n = \Theta(\log n)$ (How??)

Application - 3

- Problem: Merging Two Sorted Sequences
  - How do we combine two sorted sequences into a single increasing sequence
- Solution:
  - To merge two sorted sequences, the sum of whose lengths is n, the number of comparisons required is n – 1.

Application - 4

- Problem: Merge Sort
  - Write a recursive algorithm to sort a given sequence into the increasing order

  (using the algorithm for merging two increasing sequences into one increasing sequence)

- Solution:

  - The merge sort algorithm is $\Theta(n \log n)$ in the worst case

Help among Yourselves?

- **Perspective Students** (having CGPA above 8.5 and above)
- **Promising Students** (having CGPA above 6.5 and less than 8.5)

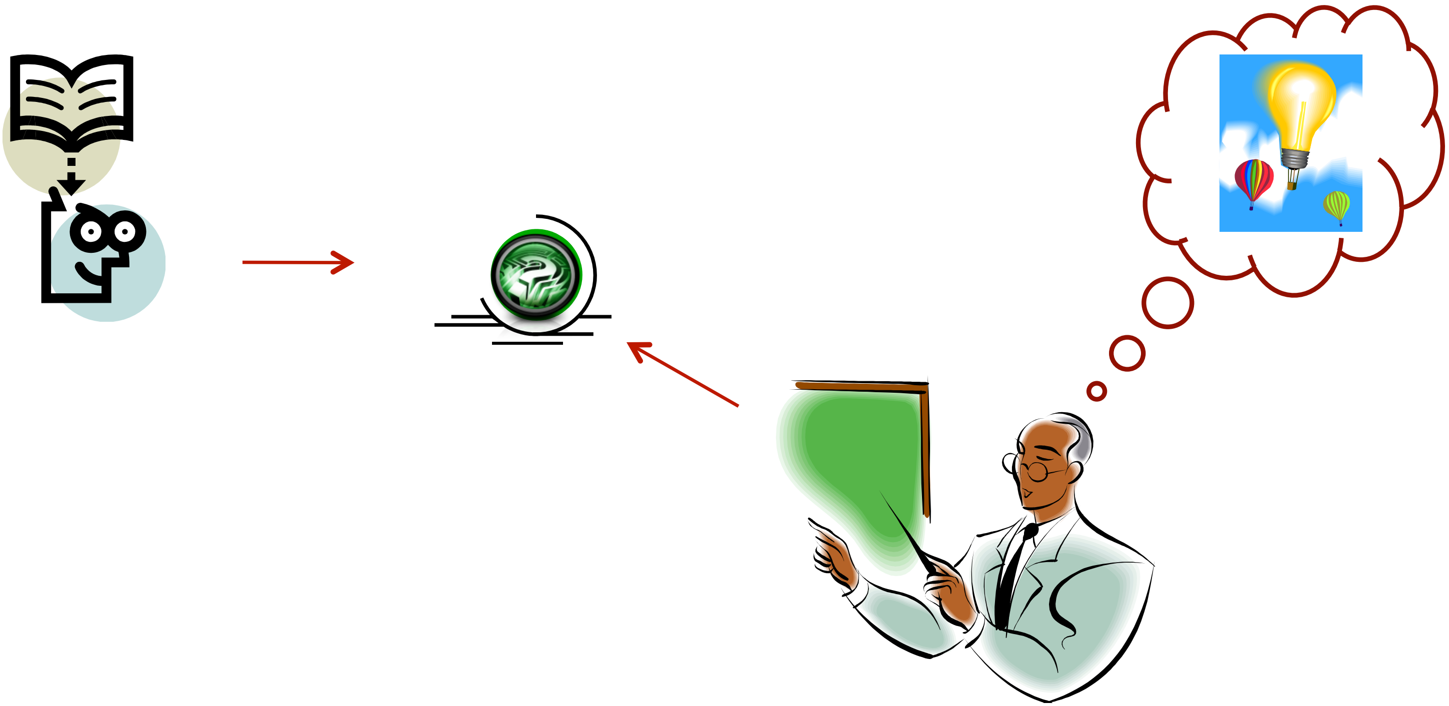
- **Needy Students** (having CGPA less than 6.5)
  - Can the above group help these students? (Your work will also be rewarded)

- You may grow a culture of **collaborative learning** by helping the needy students

Assistance

- You may post your questions to me at any time
- You may meet me in person on available time or with an appointment
- TA s would assist you to clear your doubts.
- You may leave me an email any time (email is the best way to reach me faster)

Thanks ...



... Questions ???