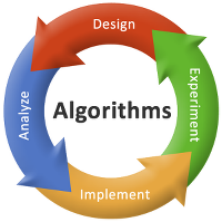
Sorting – Selection Sort & Quick Sort

Course: Algorithms

Faculty: Dr. Rajendra Prasath



Autumn 2018

Sorting – Selection Sort and Quick Sort

This lecture covers two sorting algorithms for sorting a set of n elements. The first one is the Selection sort and the second one is the quick sort. We present the algorithms and their complexity analysis. We also discuss the suitable data structures for each of the algorithm.

Recap: Sorting Algorithms

- Suggest a simple algorithm for Sorting n elements
  - Correctness: First Test whether will the algorithm work for a small set of the input? Then apply on a bigger set
  - Choose a suitable data structure
  - Perform complexity analysis
    - How much space and time required in the worst case?
  - Is the solution adaptable?
    - With the growing size of the input n
- How to get the tight bound of the sorting algorithm in terms of the running time?

Recap: Insertion Sort

- Auxiliary Space: O(1)

- Boundary Cases:
  - Maximum time is taken for reversely sorted sequence
  - Minimum time is taken for already sorted sequence

- Paradigm: Incremental Approach

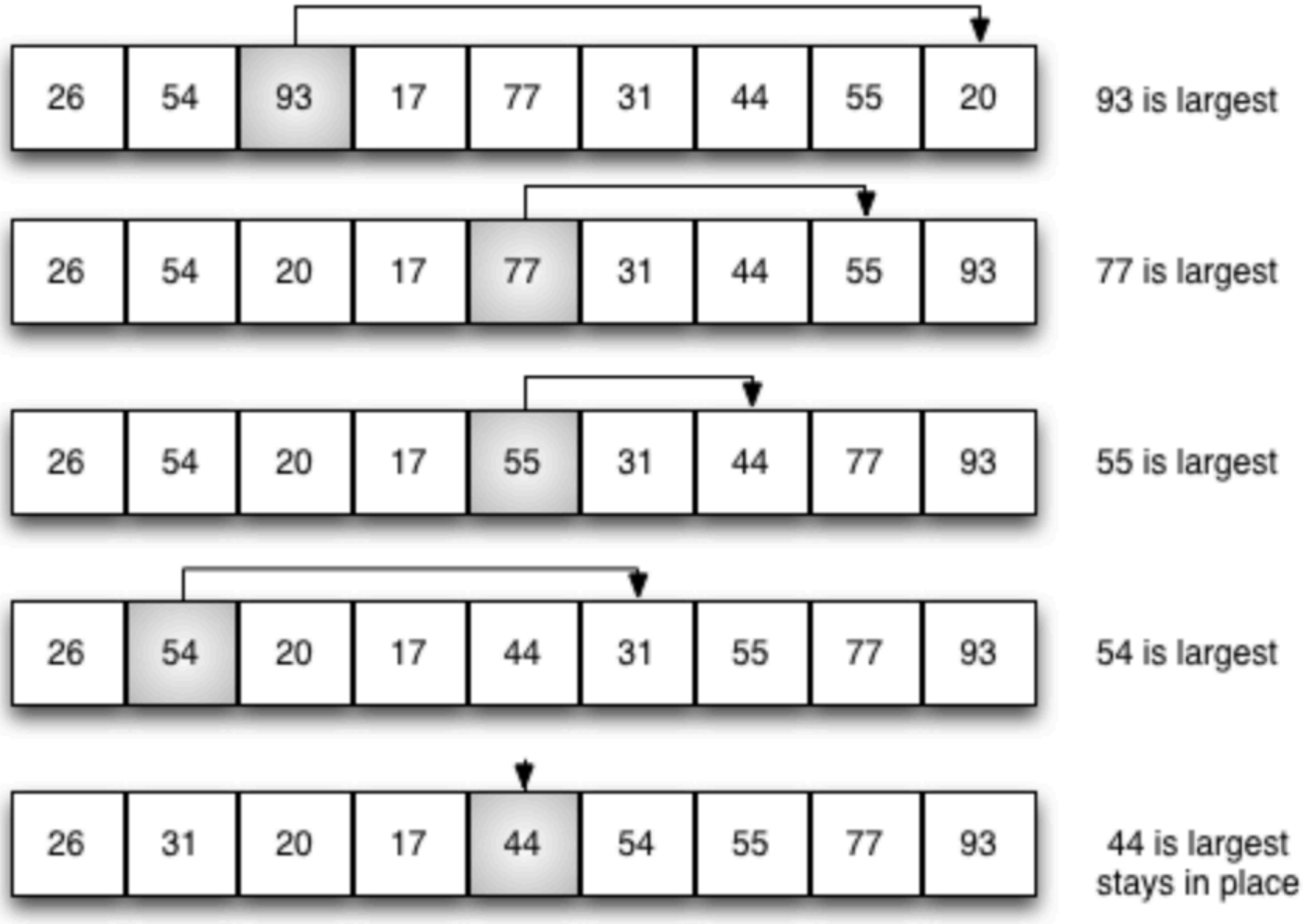
- Sorting the elements **In-Place**
- This is a **Stable Sorting** algorithm

- Computational Complexity ?
  - Running time in terms of the input size n

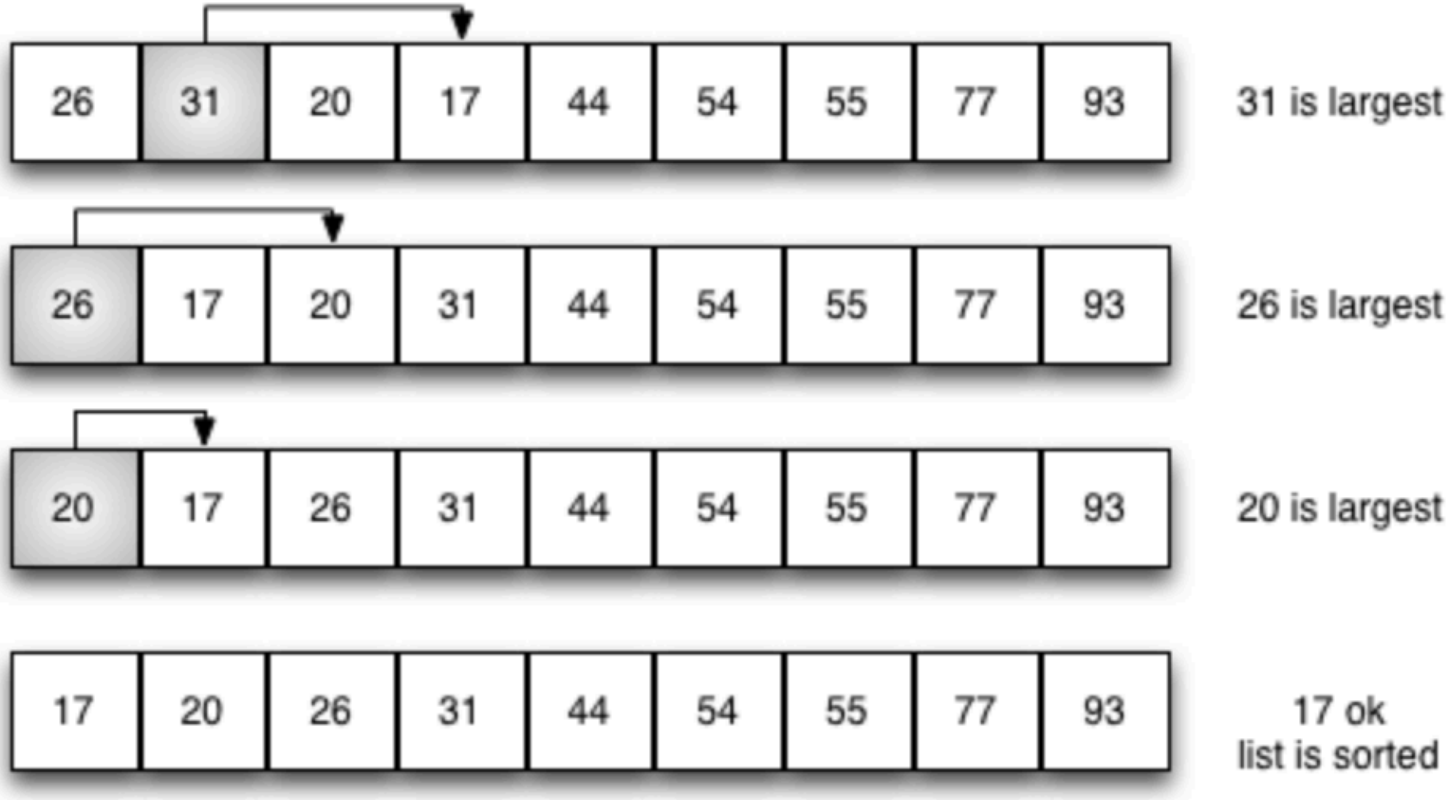
Selection Sort

- Consider the set of n elements
- How to sort them in either increasing or decreasing order?
- Improves on the bubble sort by making only one exchange for every pass through the list

Selection Sort - Illustration



Selection Sort - Illustration



Selection Sort - Steps

- Pick an element (largest or smallest) in the given array
- Swap it with the first element (depending on the partial order)

- Then perform the same with remaining (n-1) elelments

- This implies:
  - The subarray which is already sorted.
  - Remaining subarray which is unsorted.

Selection Sort - Complexity

- Worst case Complexity:
  - $O(n^2)$
- O(1) extraspace

Quick Sort - Introduction

- Like Mergesort (partition and solve)
- Quicksort follows Divide and Conquer approach
- **Basic Idea**
  - pick an element as pivot
  - partitions the given array around the picked pivot
  - Sort the partitions again in the same way

Quick Sort – Algorithm

Given an array A of n elements (e.g., integers):

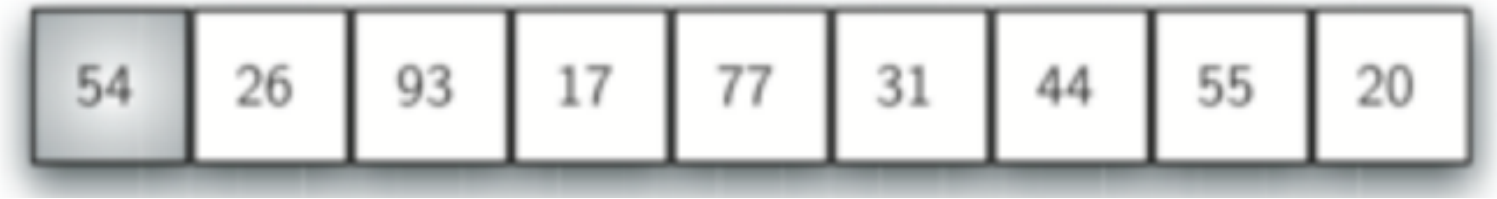
```
Procedure QuickSort()
begin
    If array contains only one element
        return that element
    Else
        pick one element to use as pivot.
        Partition elements into two sub-arrays:
        Elements less than or equal to pivot
        Elements greater than pivot
        Quicksort two sub-arrays
end
```

Quick Sort – Characteristics

- Which part is hard in this quicksort?
  - Partitioning?
  - Finding Pivot?
- How to achieve the best running time with quicksort?

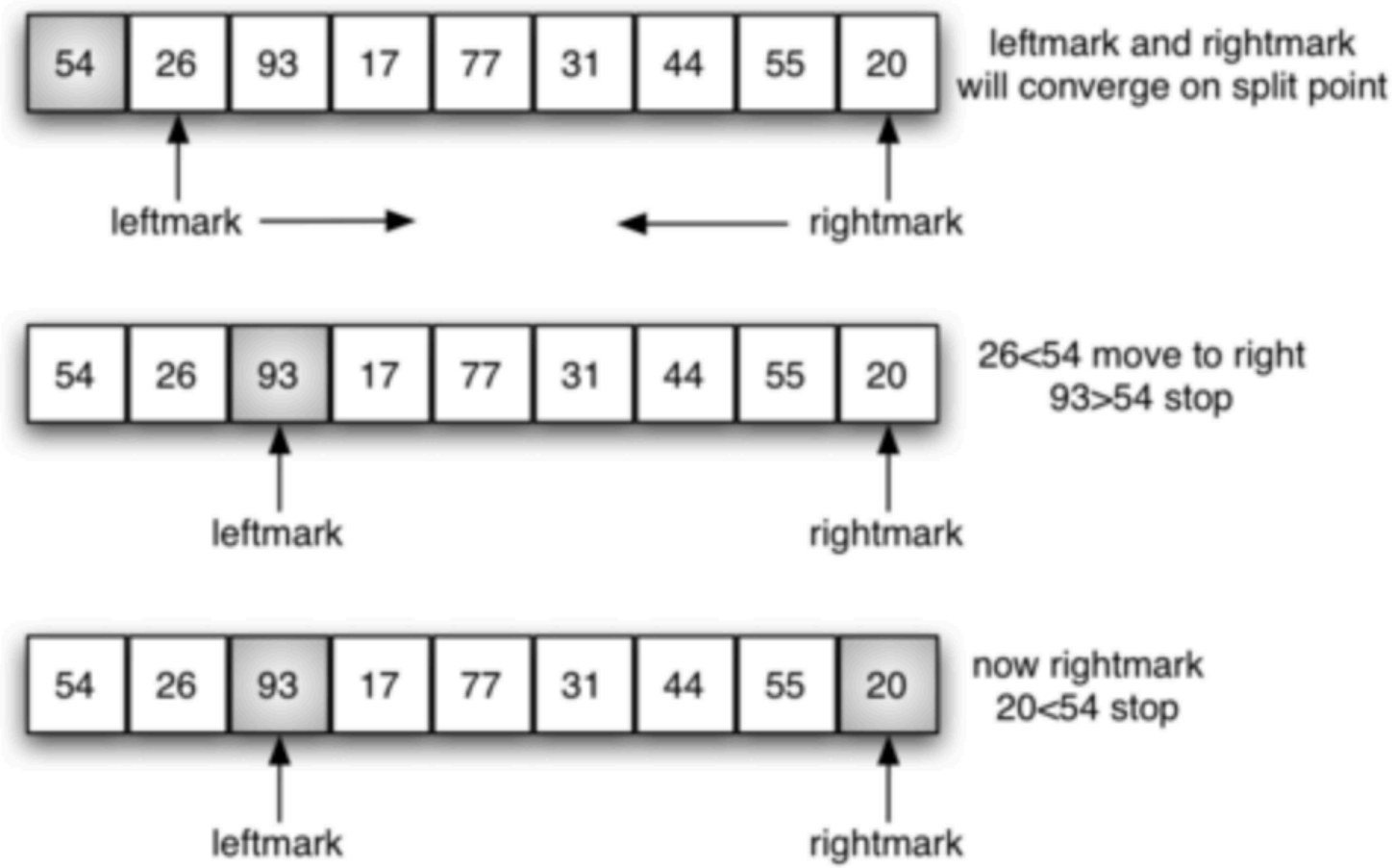
Quick Sort – Example

- Consider the following elements

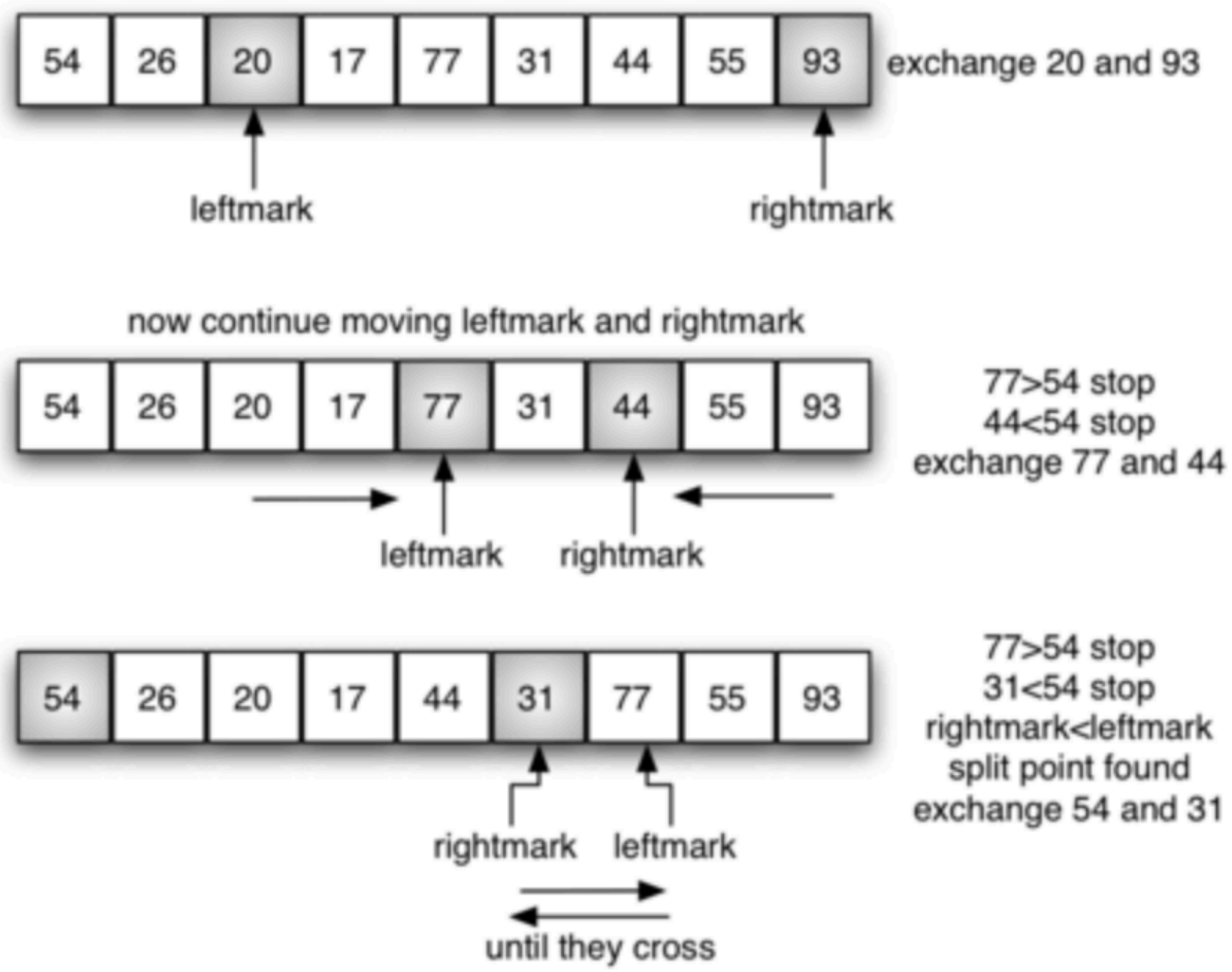


- How to sort out the elements

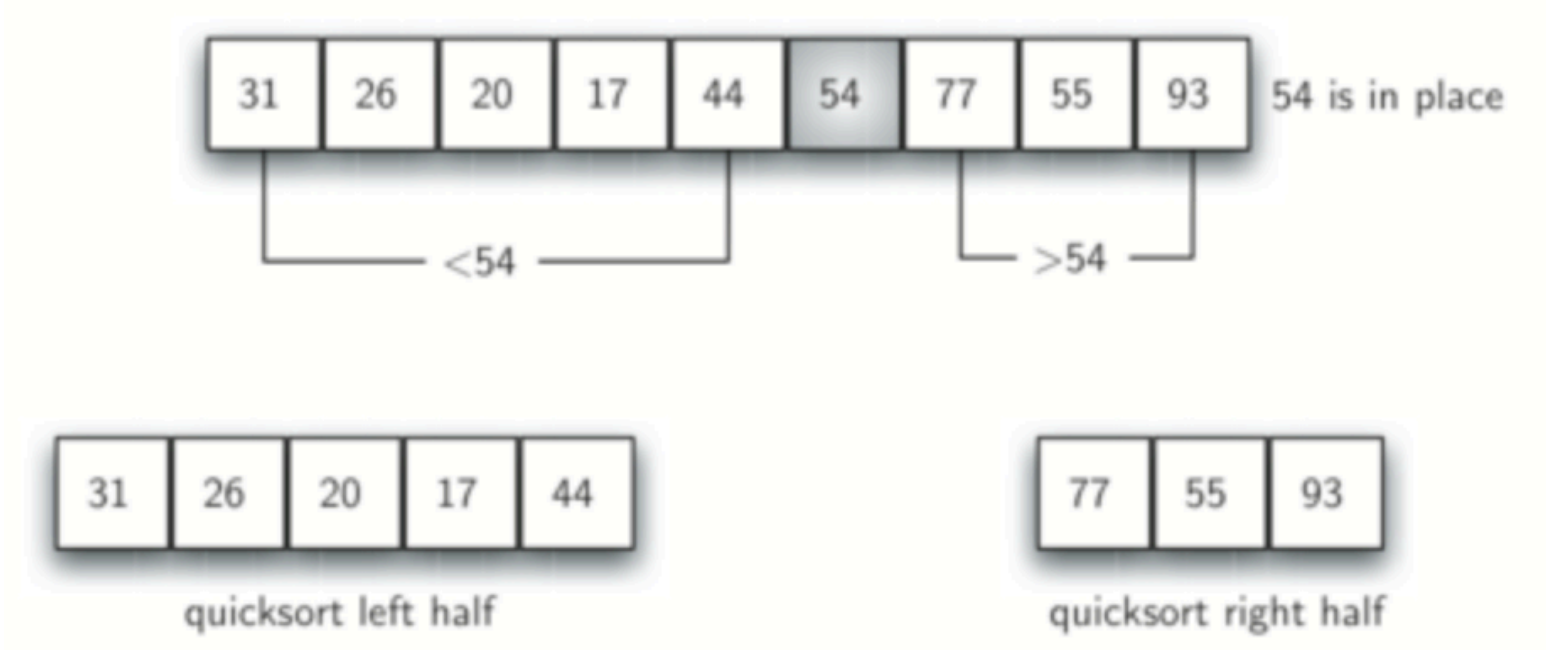
Quick Sort – Illustration



Quick Sort – Illustration



Quick Sort – Illustration



- The Final Sorted Sequence:

  17 20 26 31 44 54 55 77 93

QuickSort - Analysis

- Assume that keys are random, uniformly distributed

- What is best case running time?

  - Recursion:
  - Partition splits array in two sub-arrays of size n/2
  - Quicksort each sub-array
  - Depth of recursion tree? $O(\log_2 n)$
  - Number of accesses in partition? $O(n)$
  - Best case running time: $O(n\log_2 n)$

QuickSort - Analysis

- Assume that keys are random, uniformly distributed.
- Best case running time: O(n log2n)

- Worst case running time?
  - Recursion:
  - Partition splits array in two sub-arrays:
  - one sub-array of size 0
  - the other sub-array of size n-1
  - Quicksort each sub-array
  - Depth of recursion tree? O(n)
  - Number of accesses per partition? O(n)
- Worst case running time: $O(n^2)$!!!

QuickSort - Analysis

- Worst case running time: $O(n^2)$!!!
- How to avoid the worst case?

- Improved Pivot Selection

  - Pick median value of three elements from array:
  - A[0], A[n/2], and A[n-1].

  - Use this median value as pivot.

Help among Yourselves?

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

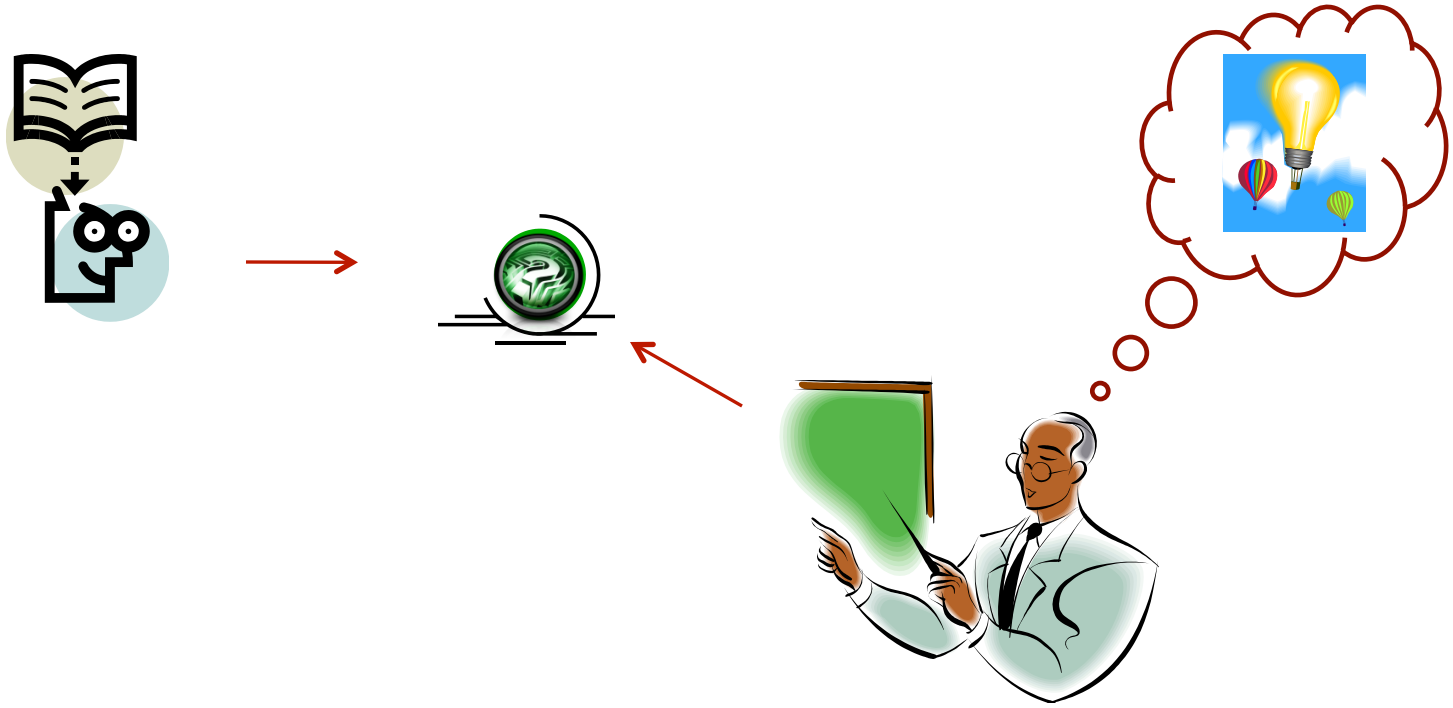
- **Needy Students** (having CGPA less than 6.5)
  - Can the above group help these students? (Your work will also be rewarded)

- 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 ???