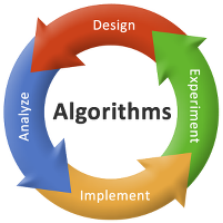
All Pairs Shortest Path Algorithm – Floyd Warshall

Course: Algorithms

Faculty: Dr. Rajendra Prasath



Autumn 2018

Floyd-Warshall Algorithm for All Pairs Shortest Path Problem

This lecture covers the interesting aspects of all pairs shortest path algorithm – The popularly known Floyd–Warshall Algorithm. We will also look at its computation complexity with the limitations.

Dijkstra's Algorithm

- Single Source Shortest Path Algorithm proposed by Dijkstra in 1956 (Originally conceived)

- Basic Idea:
  - Two sets are maintained:
  - one set contains vertices included in shortest path tree and the other set includes vertices not yet included in shortest path tree
  - At every step of the algorithm, we find a vertex which is in the other set (set of not yet included) and has a minimum distance from the source

- Similar to Prim's algorithm for MST

Bellman-Ford Algorithm

- A Single Source Shortest Path Algorithm
- Dijkstra's algorithm does not work with a graph having negative edges
- Basic Idea:
  - Assume that the graph has n nodes and m edges
  - Consider all edges and relax them (n – 1) times
    - Why?
      - Check with the length of the longest path in the graph
    - Dynamic programming

Key Aspect in Bellman-Ford

- Dynamic programming
  - Explore All possible solutions and find the best solution to the given problem

  - Relaxation criteria

    Consider an edge connecting a pair of vertices: (u, v)

    If (d[u] + c[u,v] < d[v]) then
    d[v] = d[u] + c[u,v]

    How many times do we relax all edges?
    - (n-1) times (Why?)
    - the longest possible path connecting n edges

Floyd – Warshall Algorithm

- Problem:
  - Find the shortest path between every pair of source and destination vertices
  - Negative edges are allowed
  - What is the solution?
    - Different approaches can be applied
  - We focus on the solution based on
    - Dynamic Programming
    - Divide and Conquer approach
    - . . .

    and so on

Floyd – Warshall: Basic Idea

- Solution:
  - Dynamic Programming
  - Pick one by one pick vertices and updates all shortest paths which include the picked vertex as an intermediate vertex in the shortest path
  - When we pick a vertex k as an intermediate vertex, we already have considered vertices {0, 1, 2, .. k-1} as intermediate vertices
  - In every step, consider traversing through the middle vertex, if direct edge weight is larger

Floyd – Warshall: Basic Idea

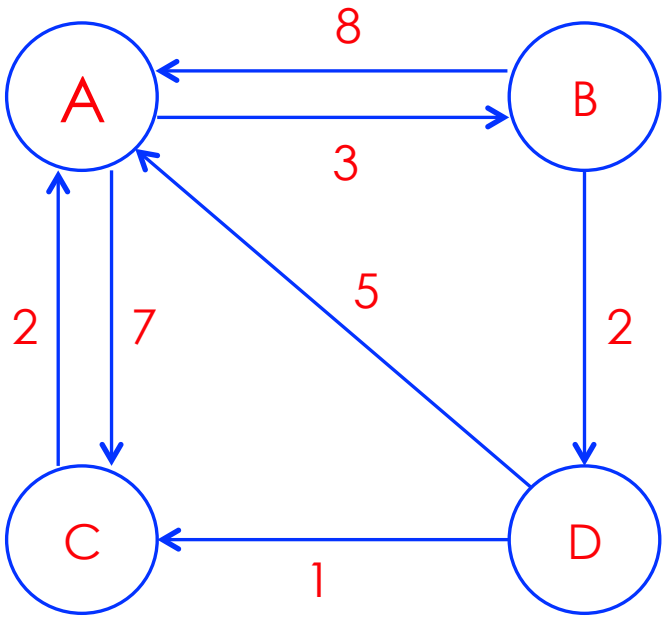
- For every pair (i, j) of vertices, there are two cases:
  - k is not an intermediate vertex in shortest path: i → j
    We keep the value of dist[i][j] unchanged
  - k is an intermediate vertex in shortest path: i → j
    Update the value of dist[i][j] as follows:

```
if dist[i][j] > dist[i][k] + dist[k][j] then
    dist[i][j]  = dist[i][k] + dist[k][j]
```

- Choose the **minimum** and store it in dist[i][j]
- Explore optimal substructure property in the all-pairs shortest path problem

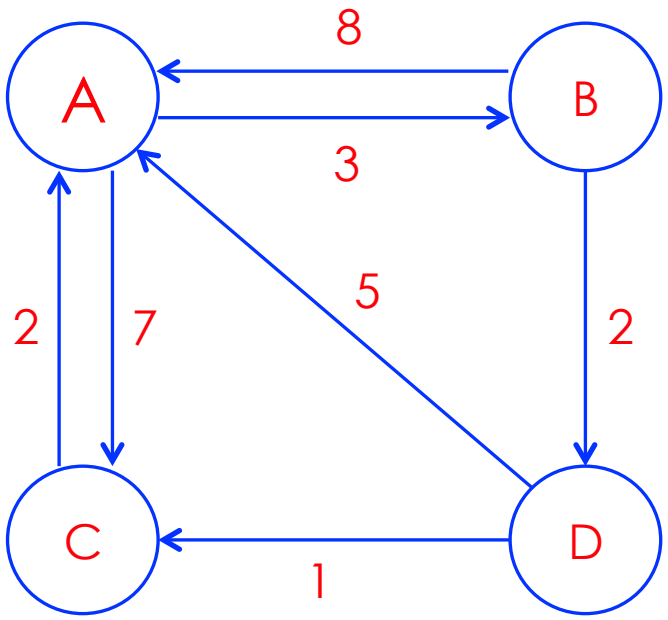
Floyd – Warshall: Illustration

- Apply Floyd – Warshall Algorithm



Floyd – Warshall: Illustration

- Dynamic Programming guides to solve the problem by considering a sequence of decisions
- In each stage we will take a decision
- For a given source and destination vertices, there can be direct path and indirect paths.
- So choose the middle vertices as the source vertex
- Now check whether does the shortest path go via that middle vertex?

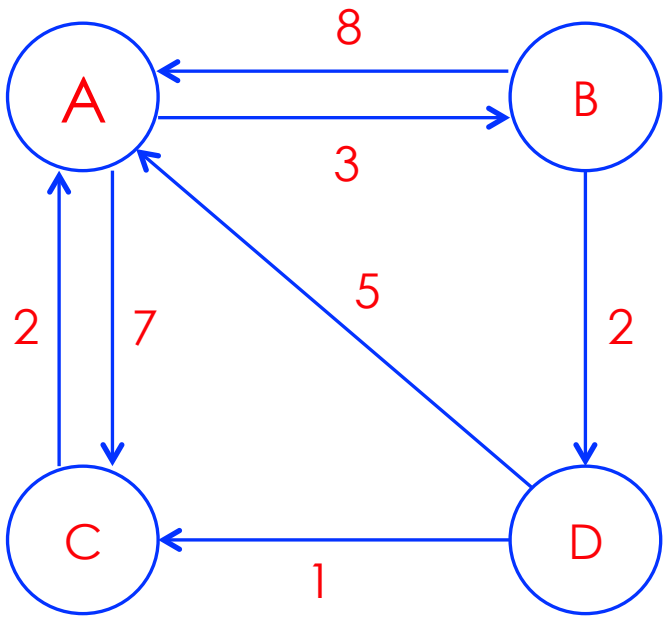


Floyd – Warshall: Illustration

- Create an Initial Matrix

M^0 =

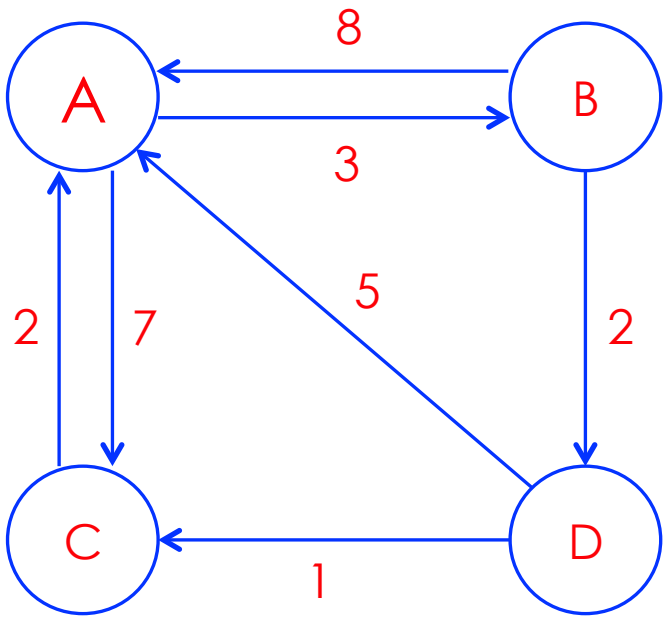
	A	B	C	D
A	0	3	∞	7
B	8	0	2	∞
C	5	∞	0	1
D	2	∞	∞	0



Floyd – Warshall: Illustration

- Compute M^1
  (via node A)

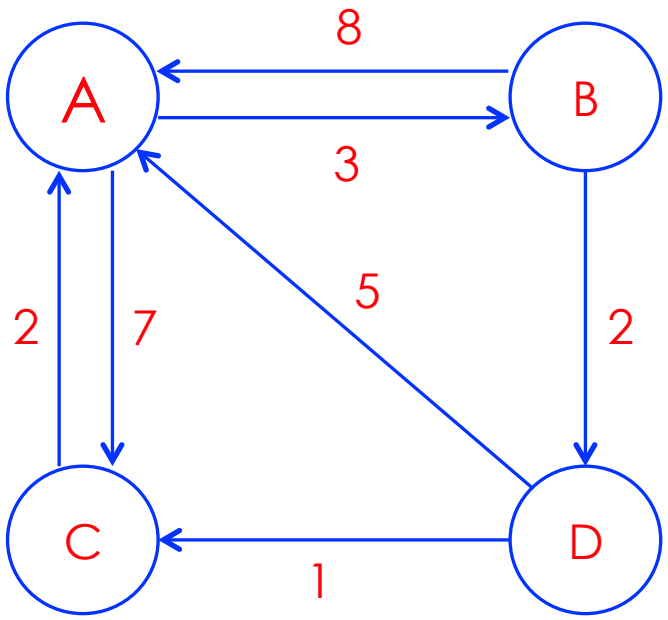
	A	B	C	D
A	0	3	∞	7
B	8	0	2	15
C	5	8	0	1
D	2	5	∞	0



Floyd – Warshall: Illustration

- Compute M^2

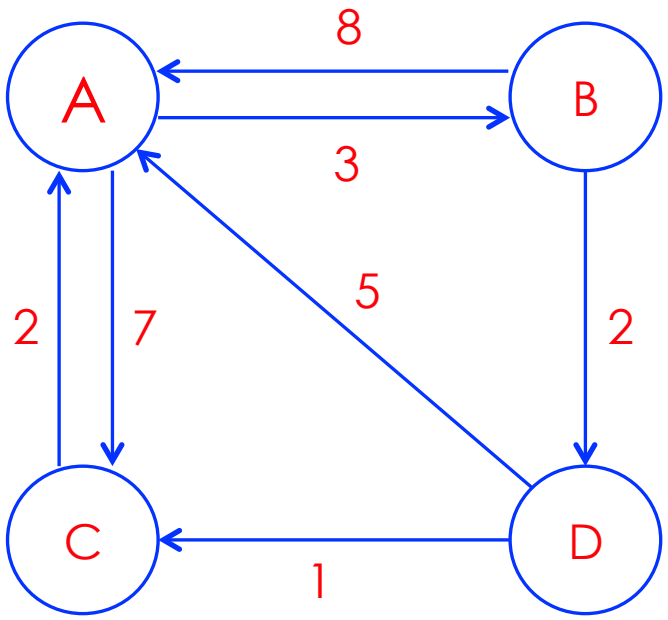
	A	B	C	D
A	0	3	5	7
B	8	0	2	15
C	5	8	0	1
D	2	5	7	0



Floyd – Warshall: Illustration

- Compute M^3

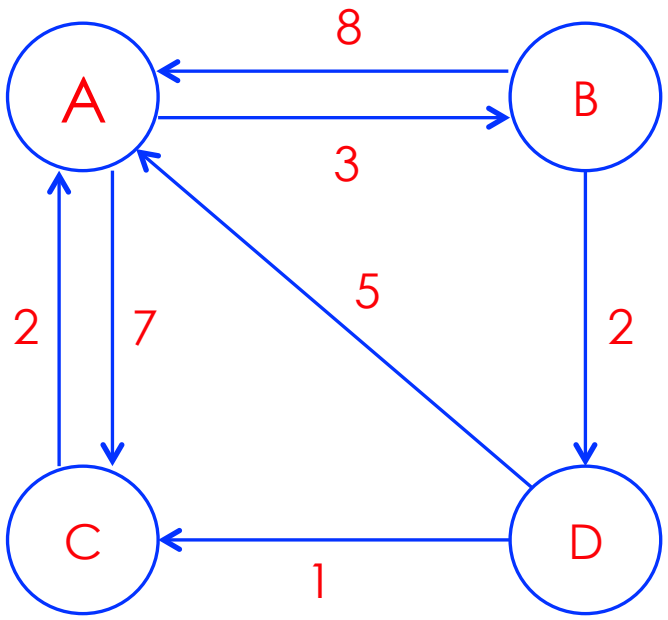
	A	B	C	D
A	0	3	5	6
B	8	0	2	3
C	5	8	0	1
D	2	5	7	0



Floyd – Warshall: Illustration

- Compute M^4

	A	B	C	D
A	0	3	5	6
B	5	0	2	3
C	3	6	0	1
D	2	5	7	0



Floyd–Warshall: Fact

- We follow the simple relaxation formula:

$$M^k[i, j] = \min\{M^{k-1}[i, k], M^{k-1}[i, k] + M^{k-1}[k, j]\}$$

- Code:

```
for (k =1; k <=n; k++) {
      for (i =1; i <=n; i++) {
            for (j =1; j <=n; j++) {
                  M[i,j] = min{ M[i,j], M[i,k] + M[k,j]}
            }
      }
}
```

Floyd–Warshall: Complexity

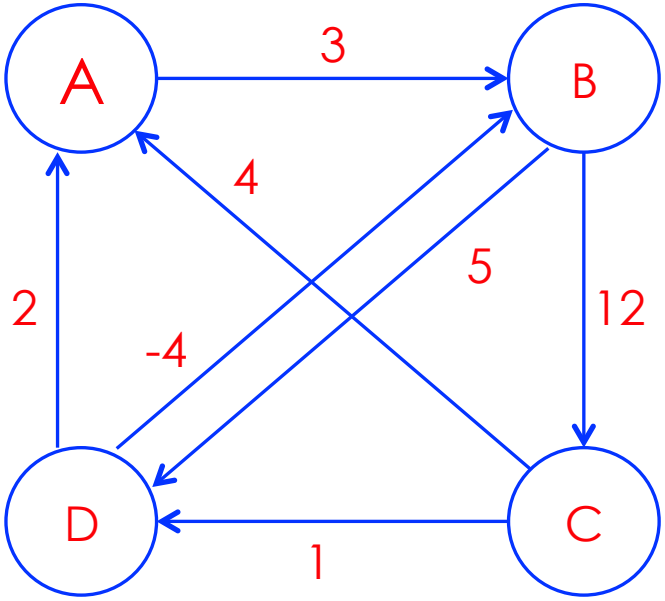
- Time Complexity

- There are $|V| = n$ vertices

➔ Thus the time complexity is $O(n^3)$

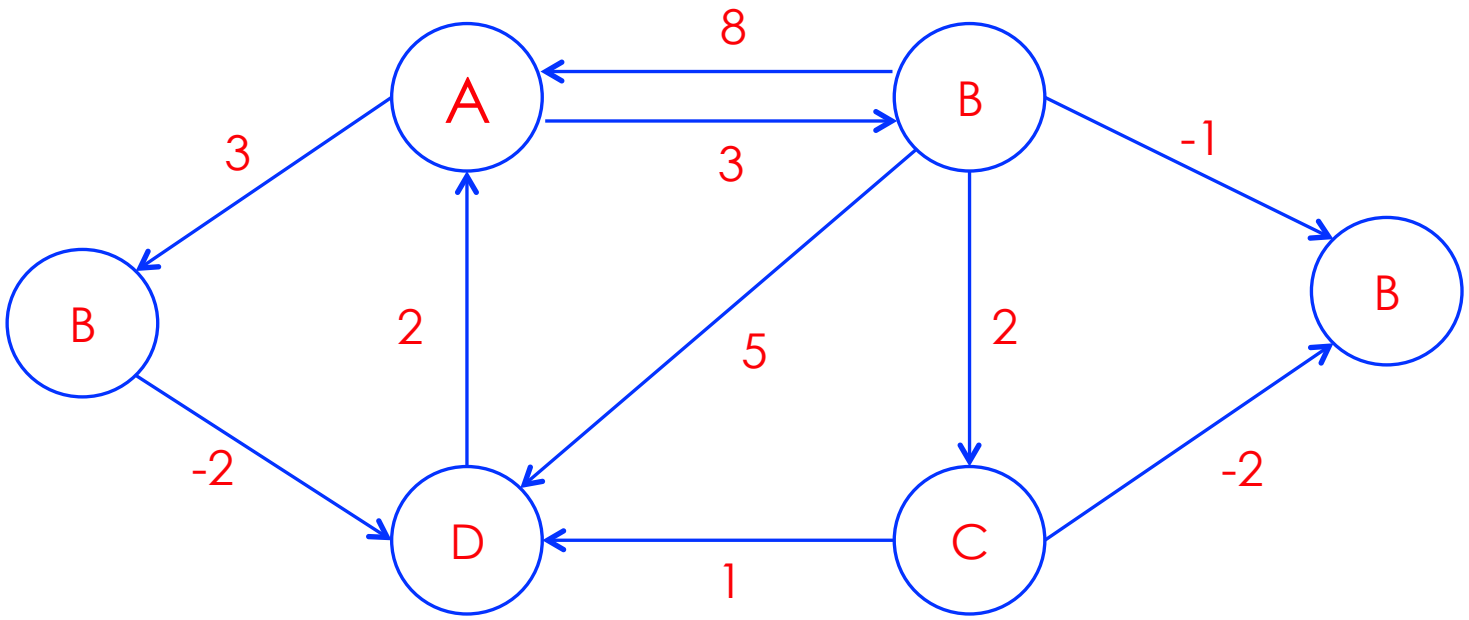
Exercise 1

- Compute All Pairs Shortest Paths



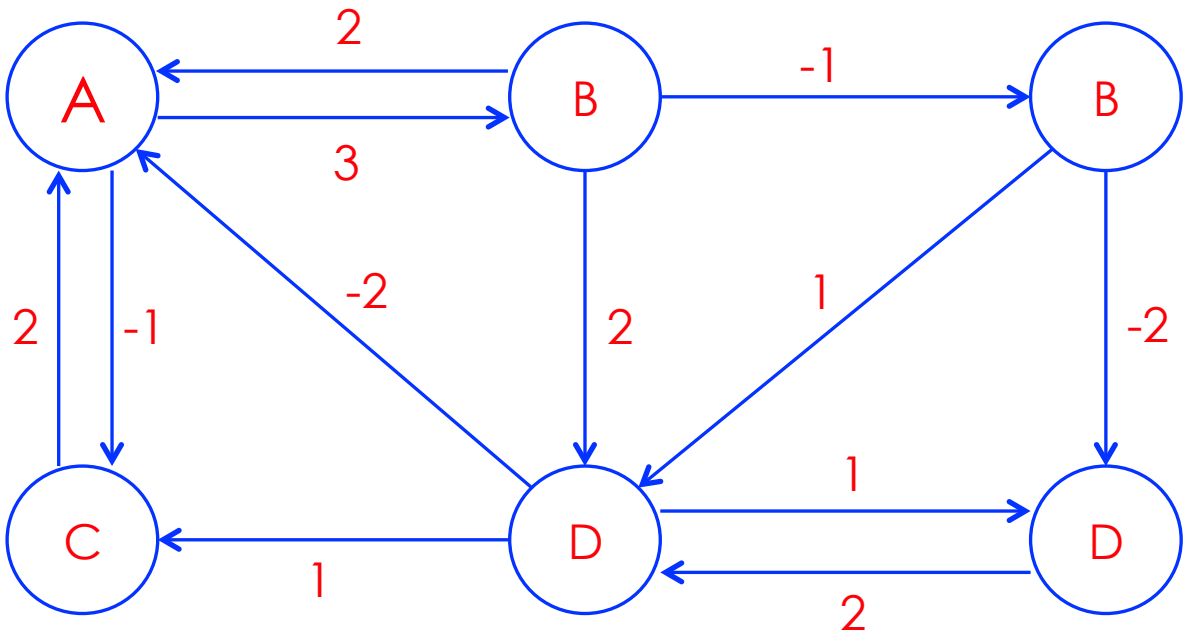
Exercise 1

- Compute All Pairs Shortest Paths



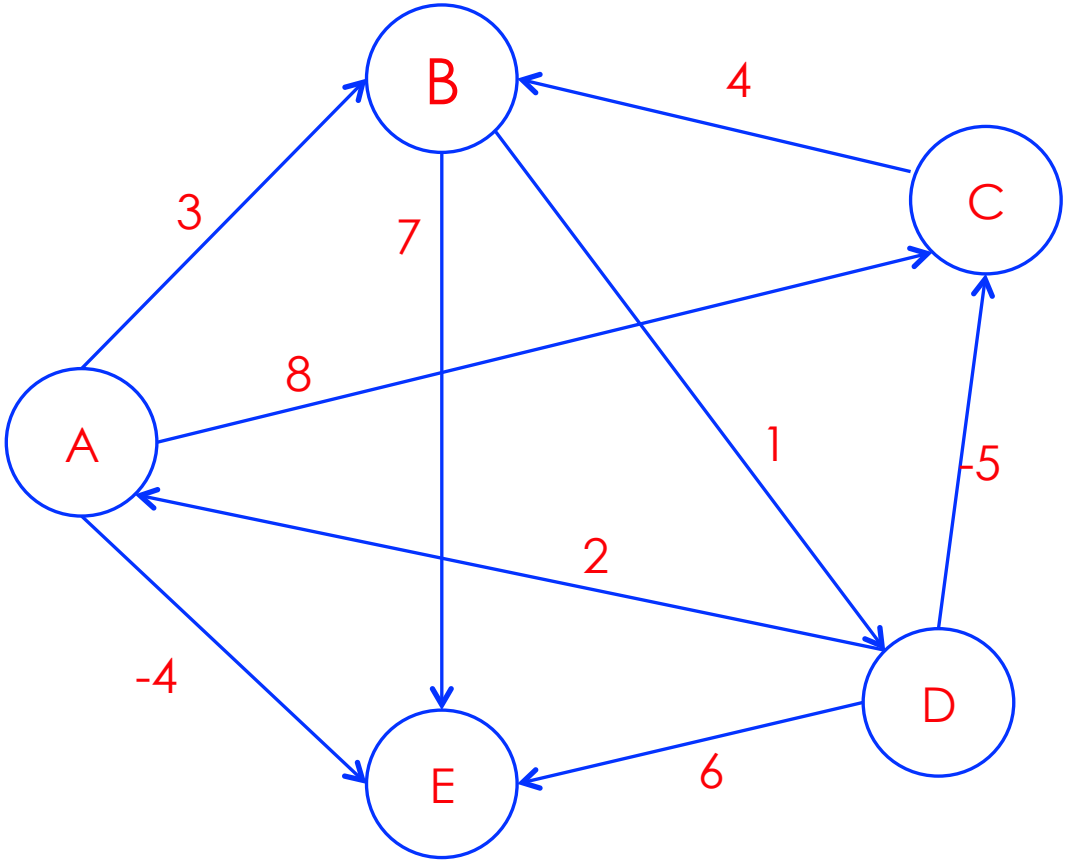
Exercise 3

- Compute All Pairs Shortest Paths



Exercise 4

- Compute All Pairs Shortest Paths



Help among Yourselves?

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

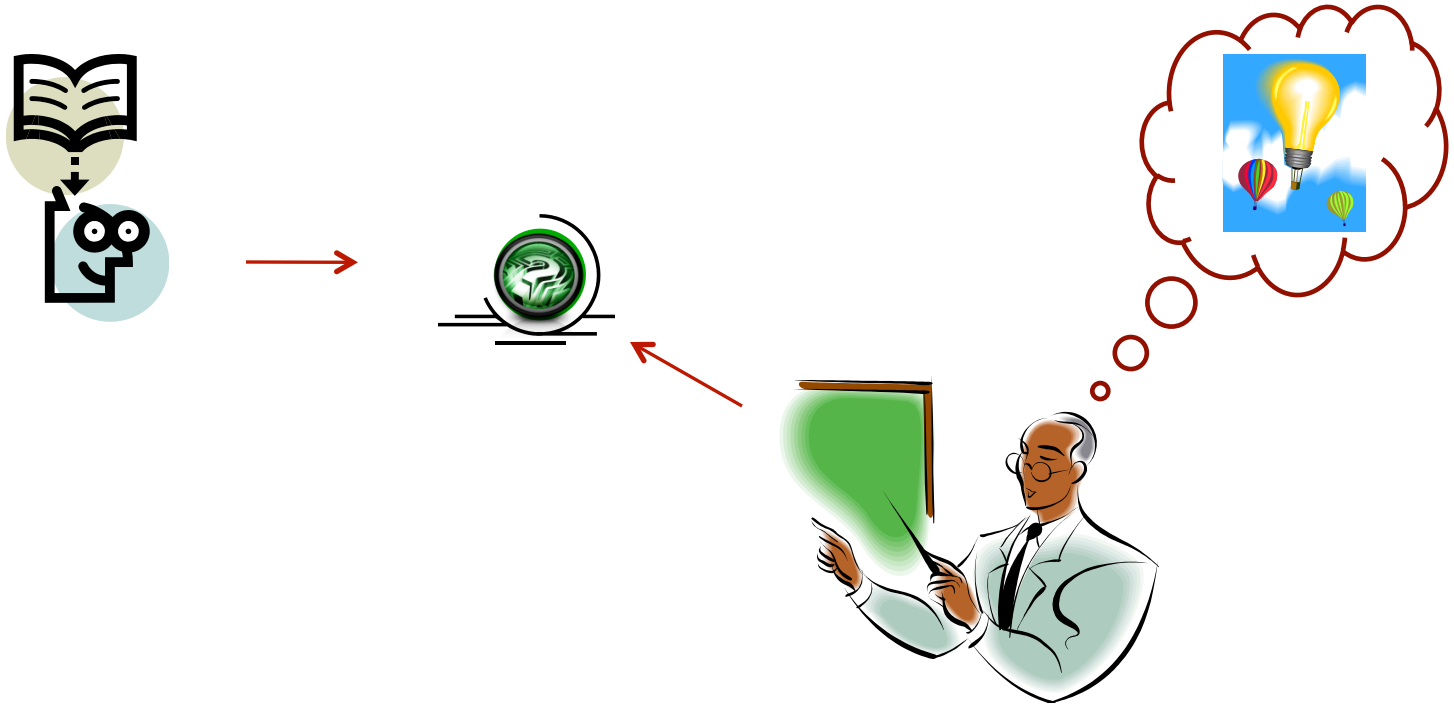
- **Needy Students** (having CGPA less than 6.5)
  - Can the above group help these students? (Your work will also be rewarded)

- 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 ???