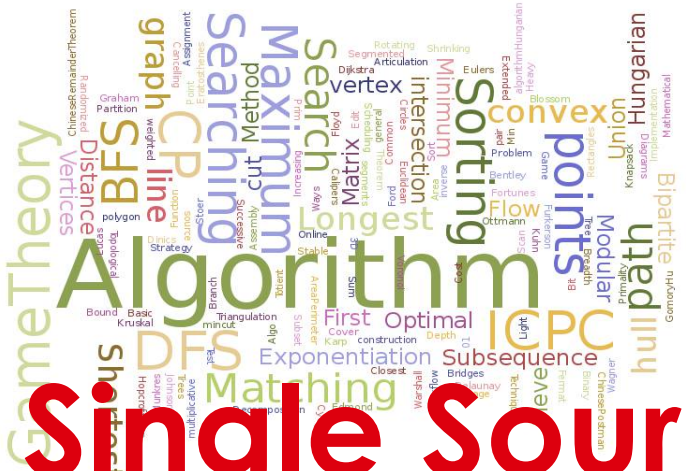
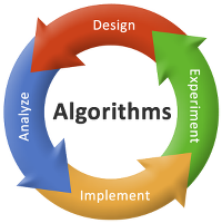


Single Source Shortest Path Algorithm - Dijkstra

Course: Algorithms

Faculty: Dr. Rajendra Prasath



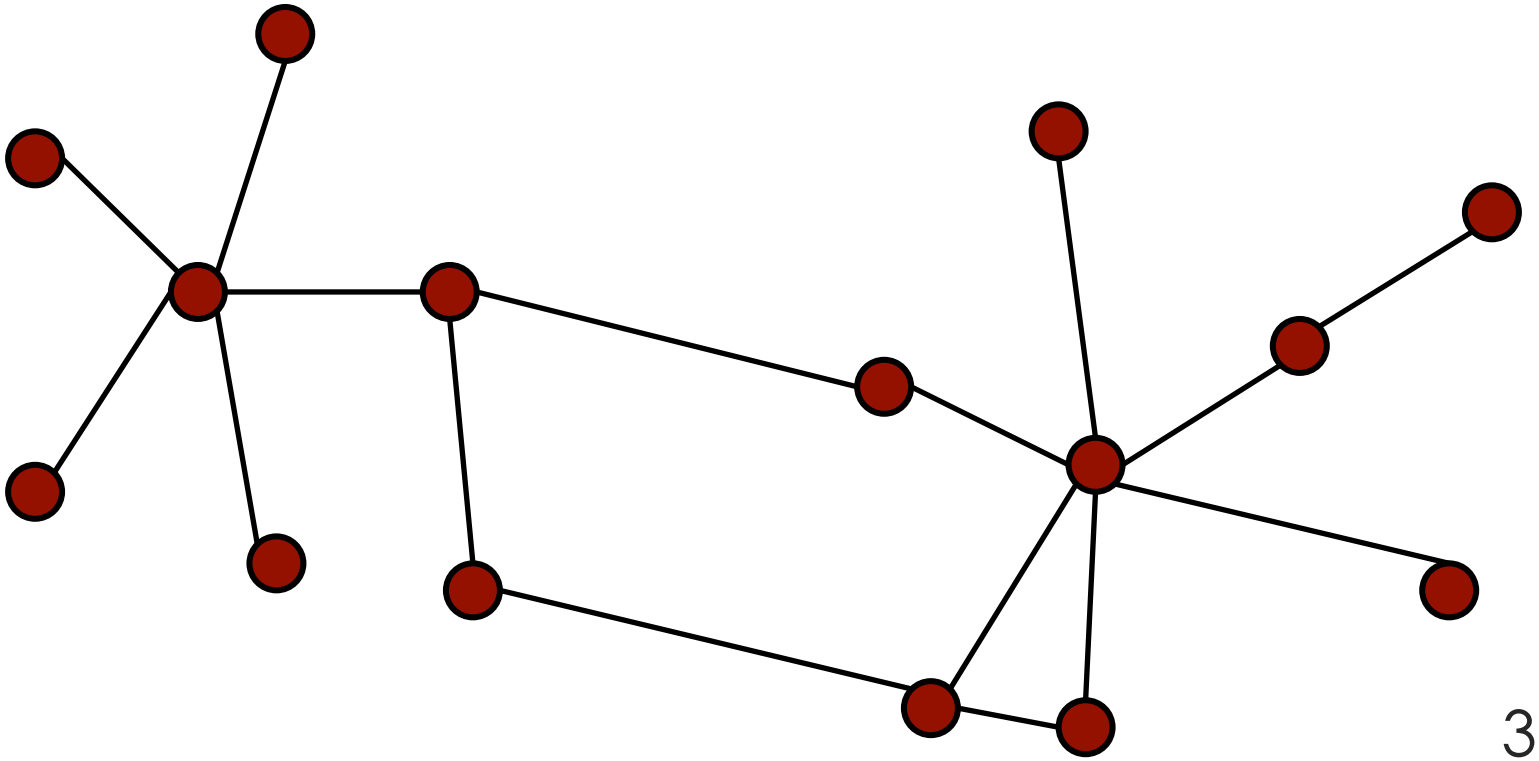
Autumn 2018

Single Source Shortest Path Algorithm - Dijkstra

This lecture covers the interesting aspects of the single source shortest path algorithm – The popular Dijkstra's Algorithm. We will also look at its computation complexity with the limitations.

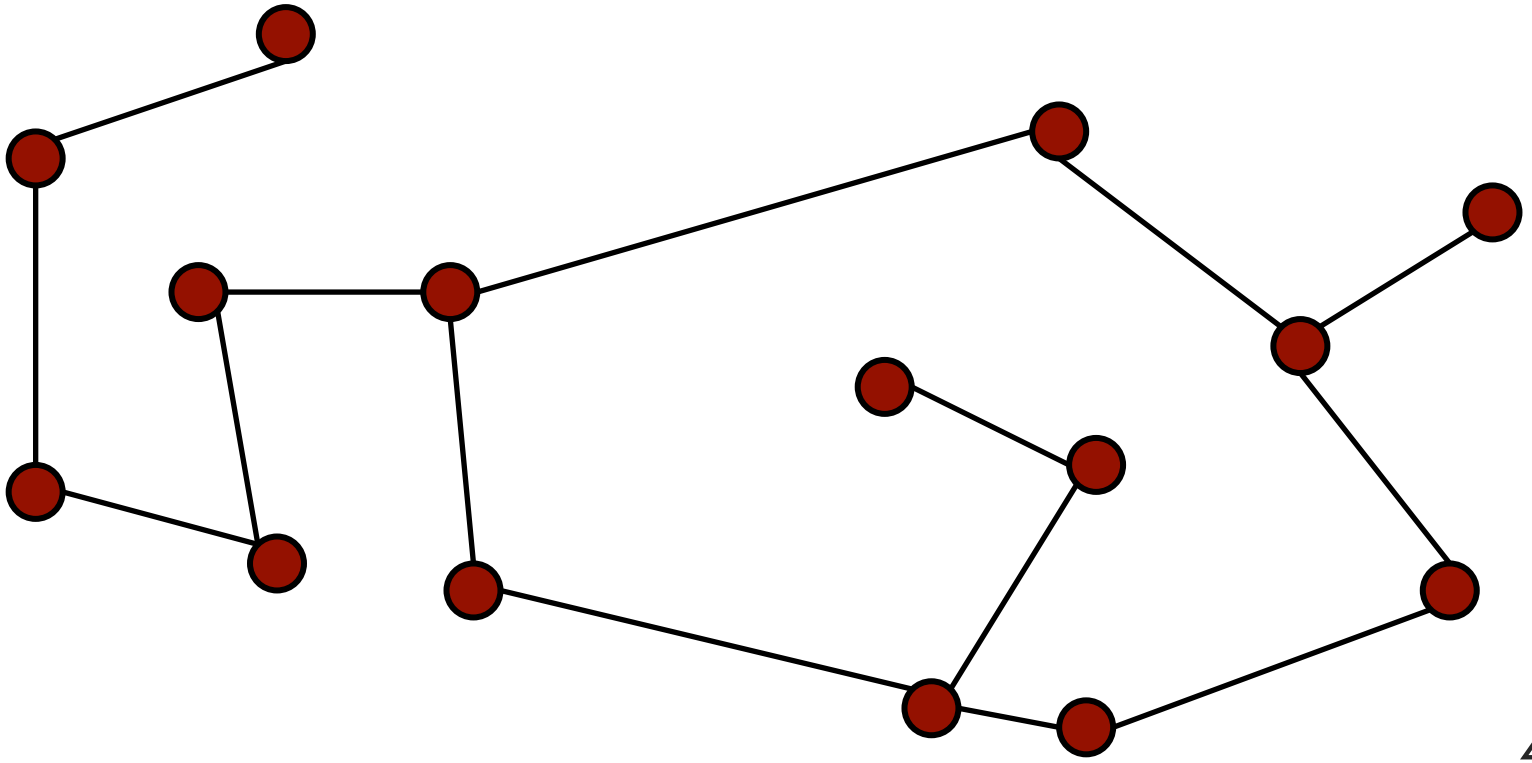
Overlay Construction Divide-and-Conquer Algorithm

Initially: connected graph



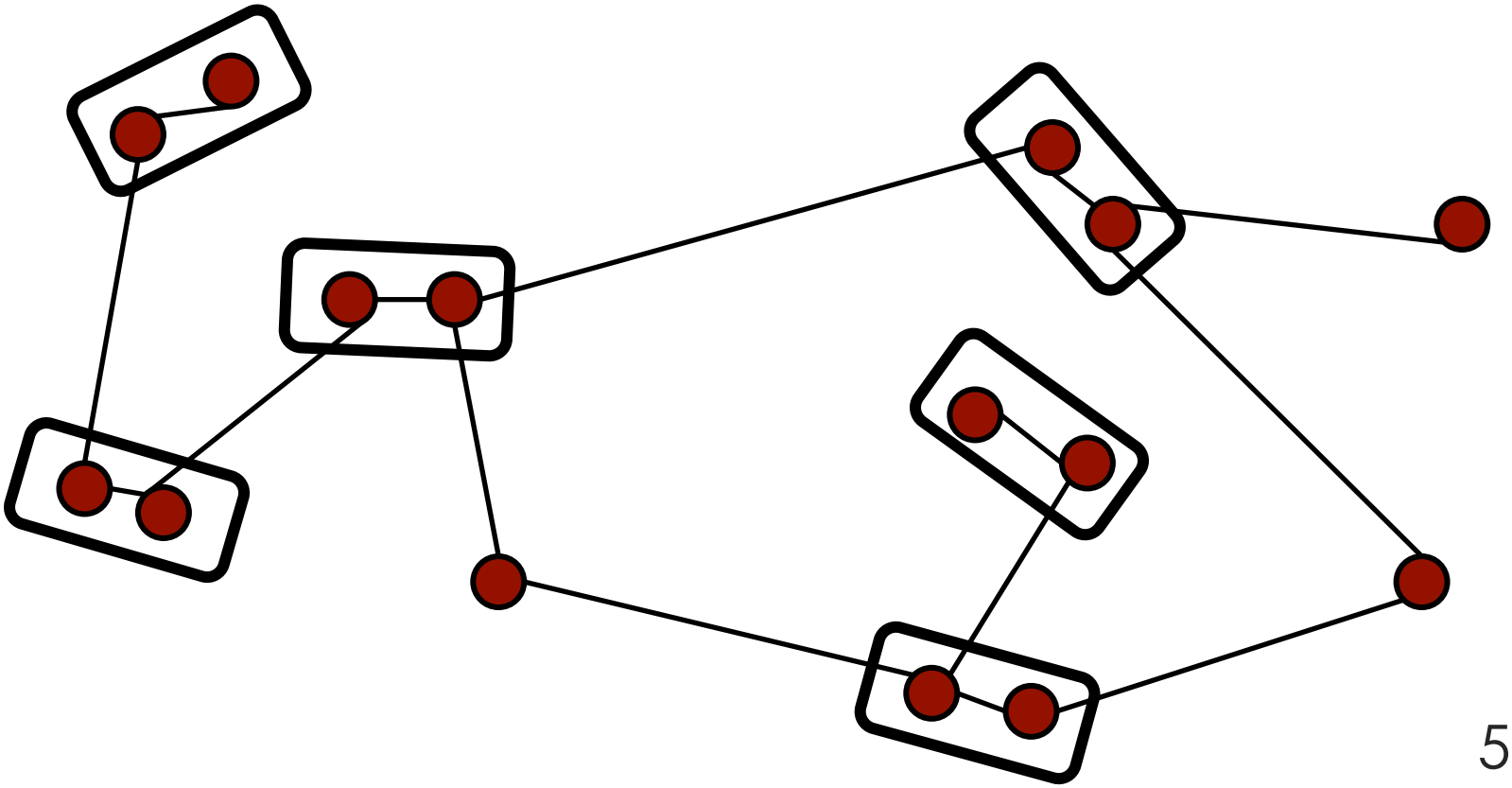
Divide-and-Conquer Algorithm

Sparsify: reduce the degree



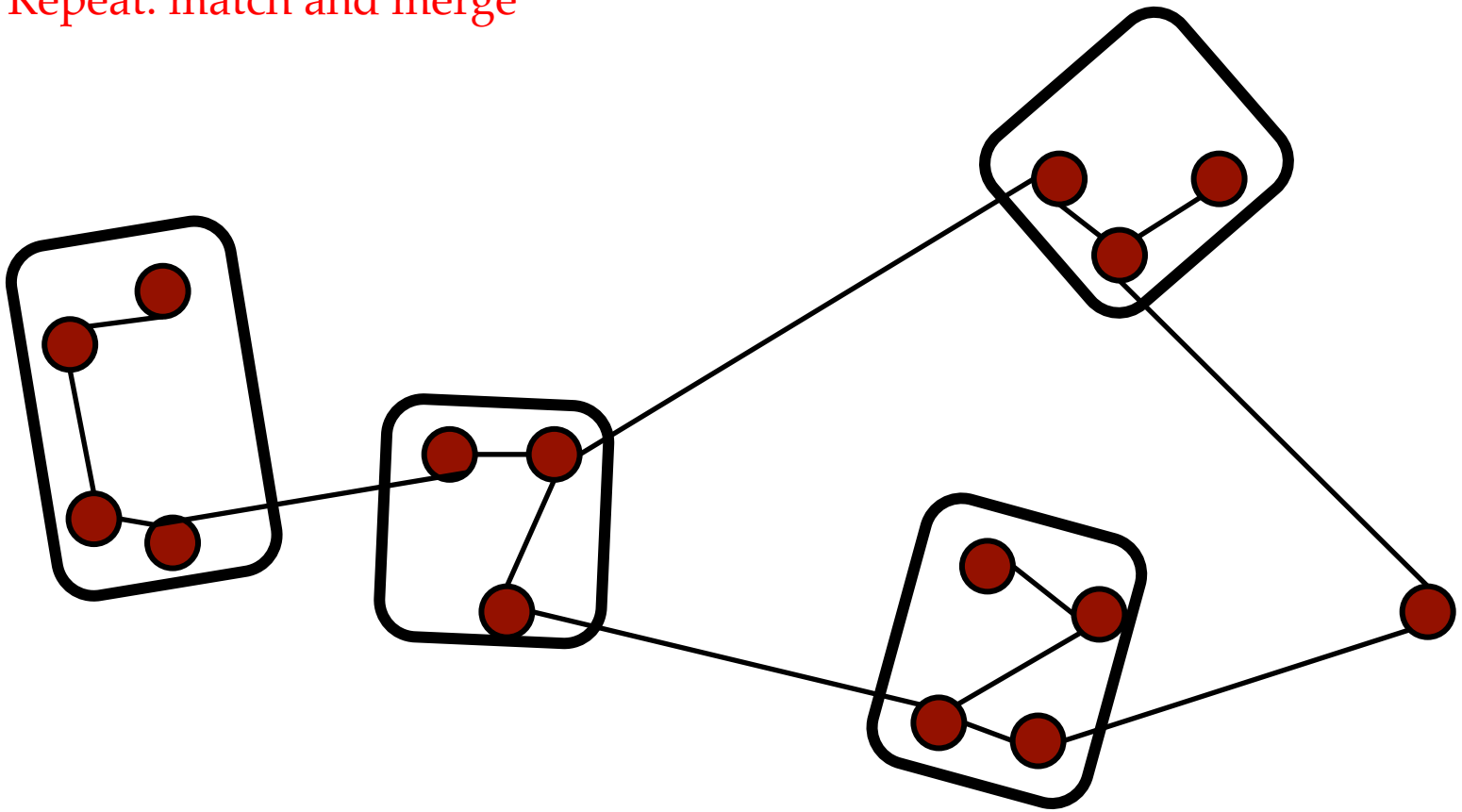
Divide-and-Conquer Algorithm

Repeat: match and merge



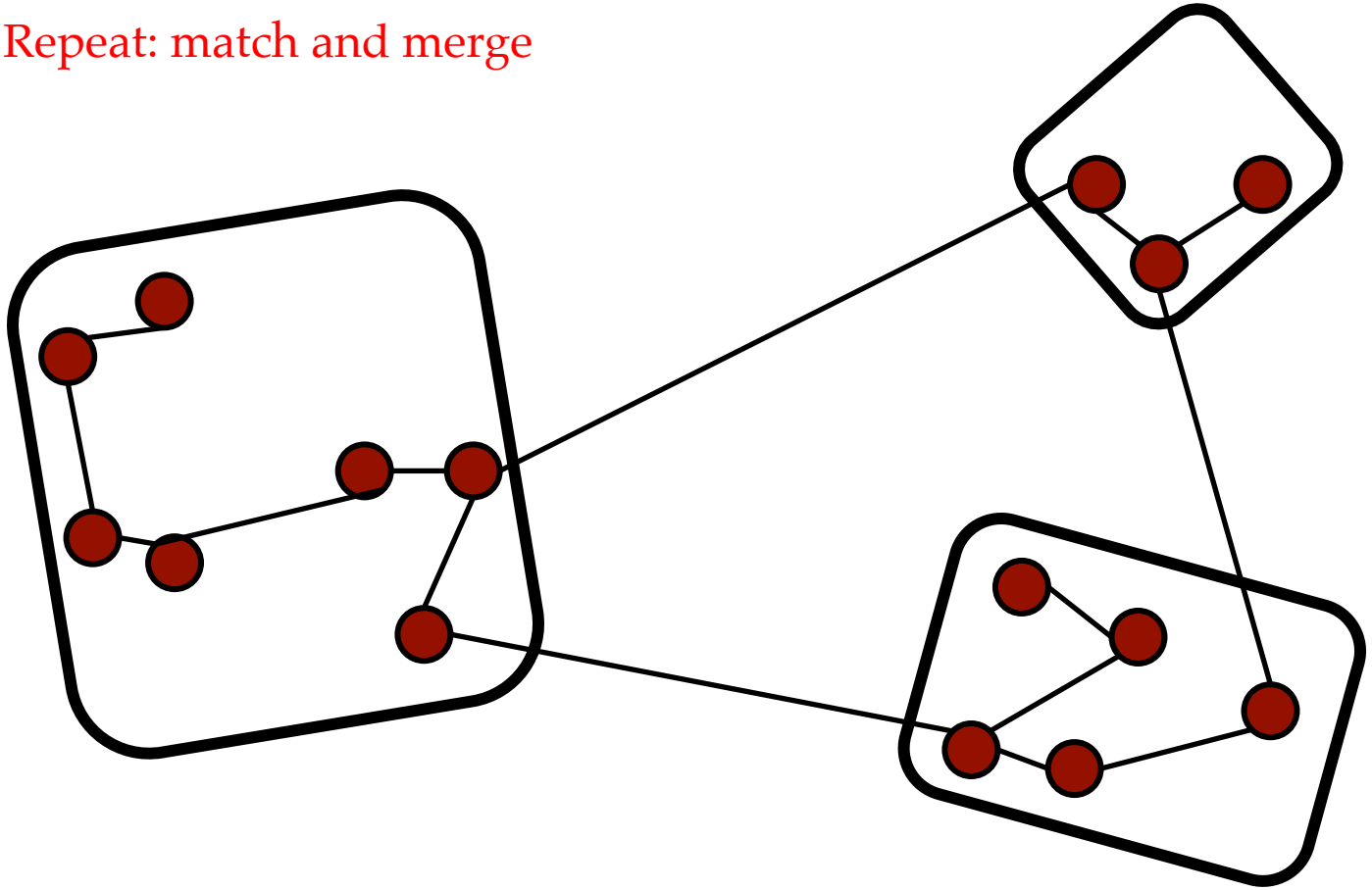
Divide-and-Conquer Algorithm

Repeat: match and merge



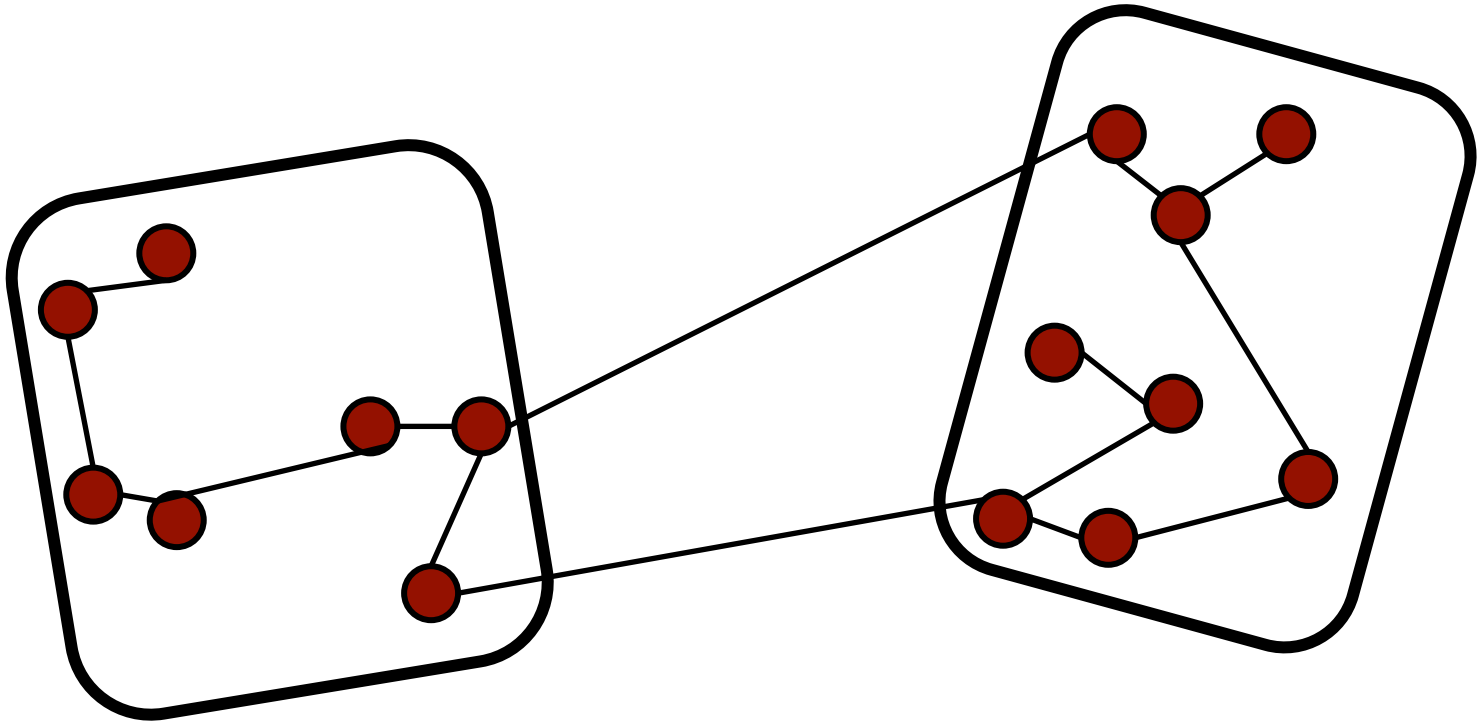
Divide-and-Conquer Algorithm

Repeat: match and merge



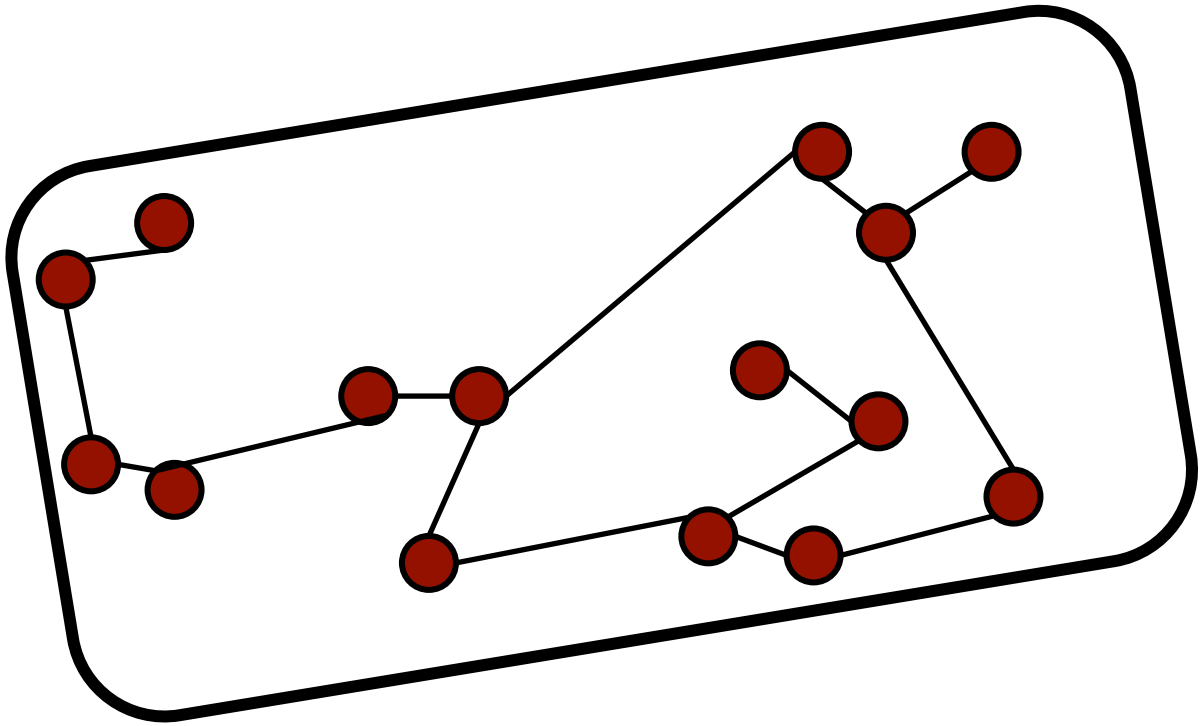
Divide-and-Conquer Algorithm

Repeat: match and merge



Divide-and-Conquer Algorithm

Repeat: match and merge



Efficiency Analysis

Merging:

- Random walks: O(log n) cost
- Bridge doubling: O(log n) iterations
- Overall: O(polylog n) time to merge topologies.

Divide-and-Conquer:

- Collection has small diameter ➔ O(log n) cost to coordinate
- Collection merging: O(polylog n) cost per merge step.
- Number of iterations: O(log n) matchings
- Overall: O(polylog n) time to form overlay.

Applications of Overlays

- Peer-to-peer computing
- Multicast Routing
- Delay Tolerant Routing
  - Message will eventually reach the destination
- Social networks
- Small World Effects
- Community Detection Problems
- Graph Searches
- Stable Matching Problems
- Various interaction networks
  - Many more applications …

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

Pseudo Code

```
Procedure Dijkstra(Graph, source)
begin
    create vertex set Q
    for each vertex v in Graph:   // Initialization
        dist[v] ← INFINITY              // Unknown distance from source to v
        prev[v] ← UNDEFINED       // Prev. node in optimal path from source
        add v to Q                      // All nodes initially in Q (unvisited nodes)
    dist[source] ← 0                    // Distance from source to source
    while Q is not empty:
        u ← vertex in Q with min dist[u]    // Select the node with least
distance
        remove u from Q
        for each neighbor v of u:           // where v is still in Q.
            weight ← dist[u] + length(u, v)
            if weight < dist[v]:                // A shorter path to v has been found
                dist[v] ← weight; prev[v] ← u
    return dist[], prev[]
end
```

Key Aspect

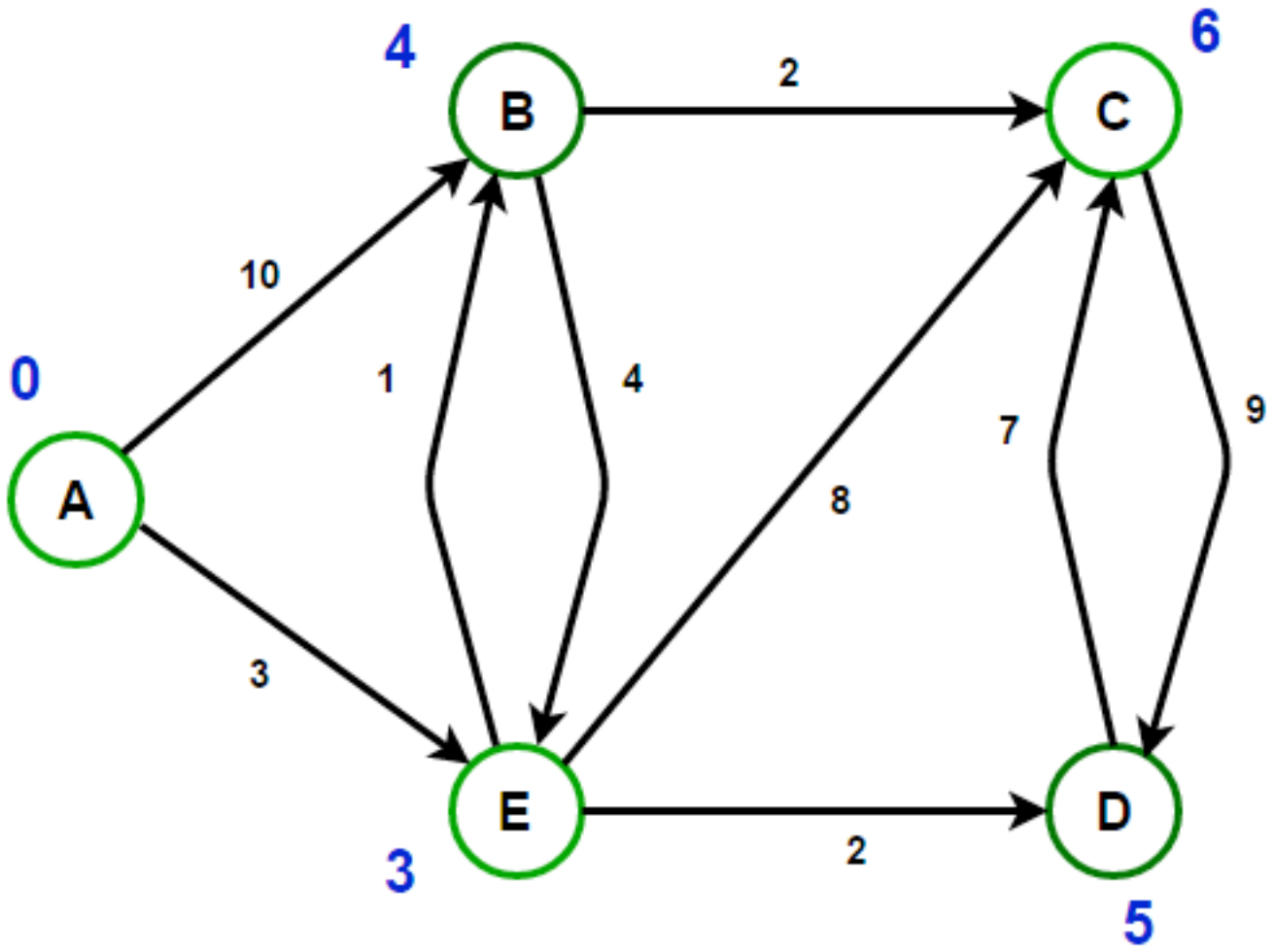
Relaxation Principle

A shorter path to v has been found

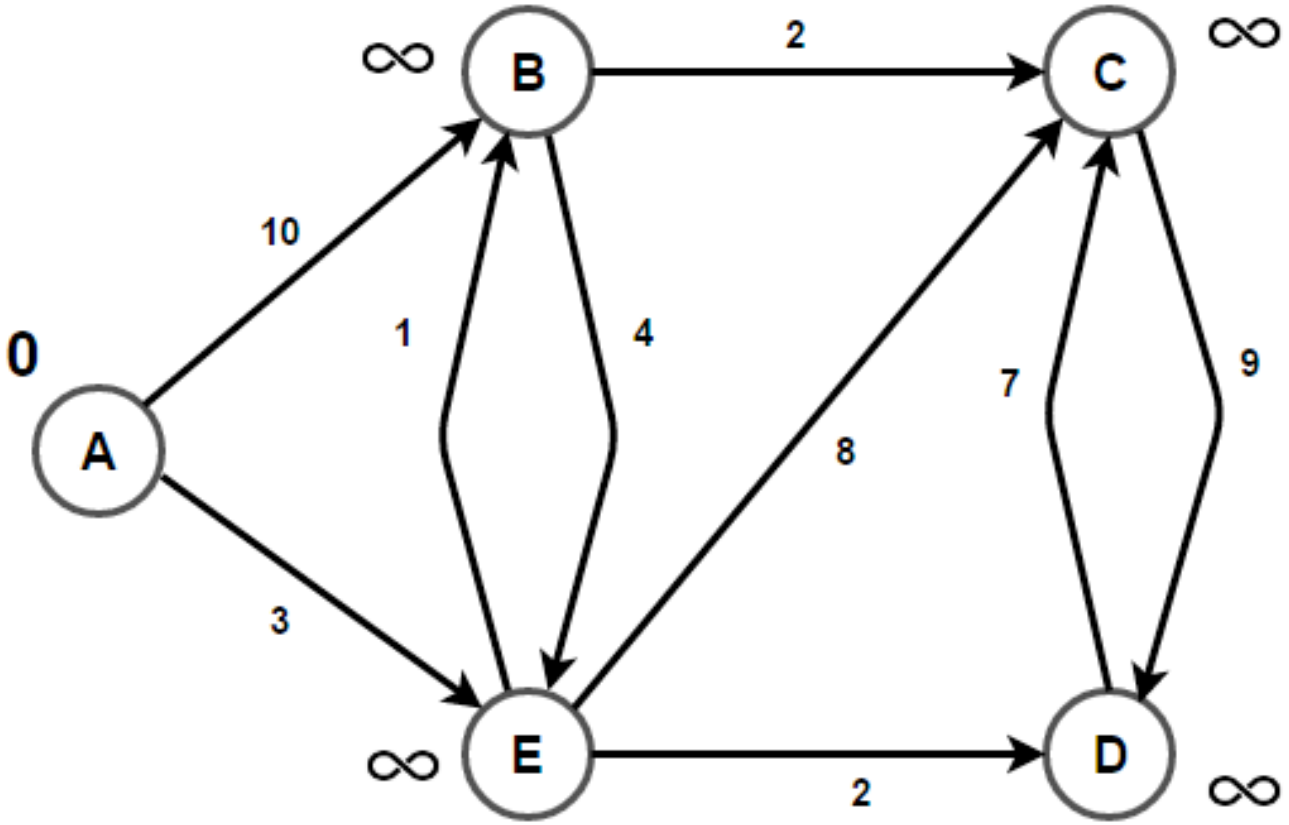
```
weight ← dist[u] + length(u, v)

if ( weight < dist[v] ) then
    dist[v] ← weight
```

Dijkstra's - Illustration

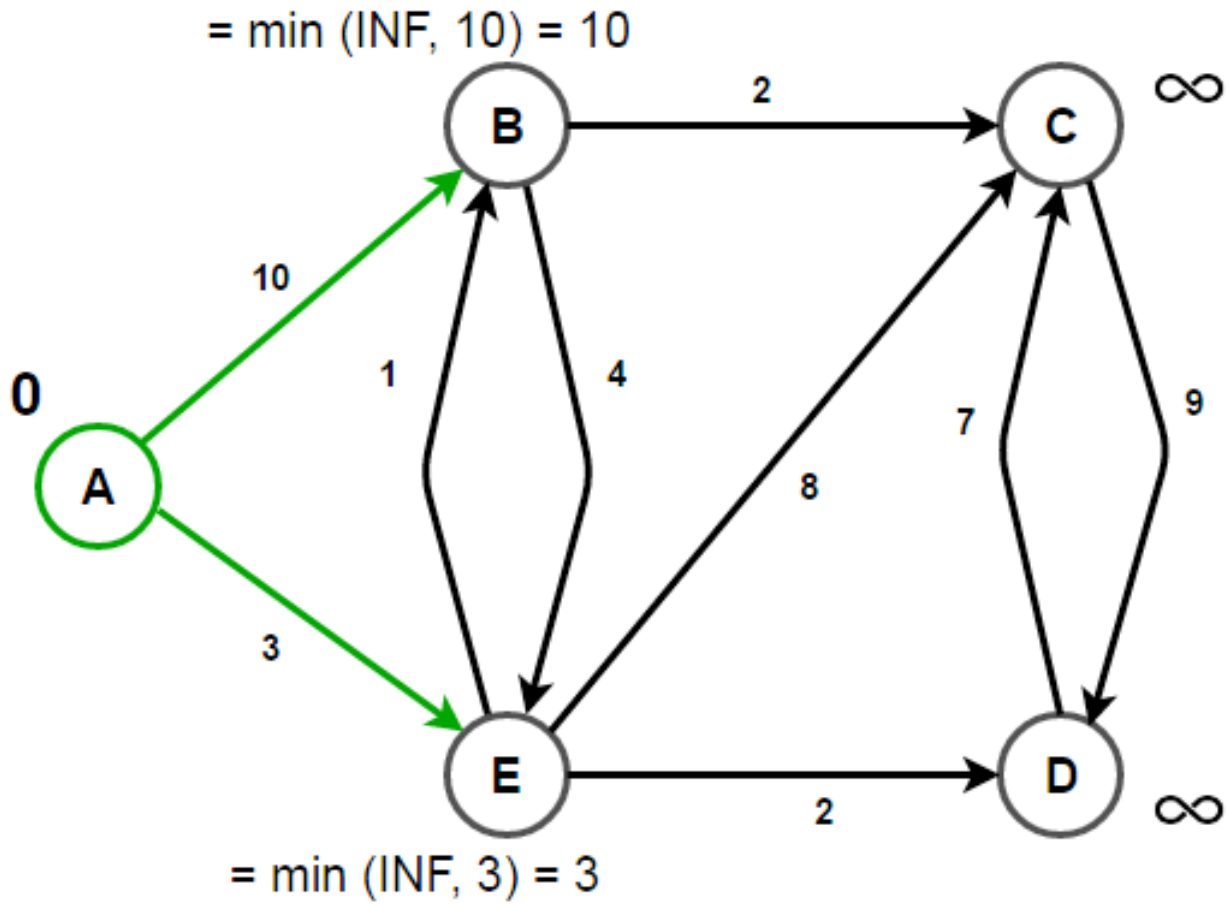


Dijkstra's - Illustration

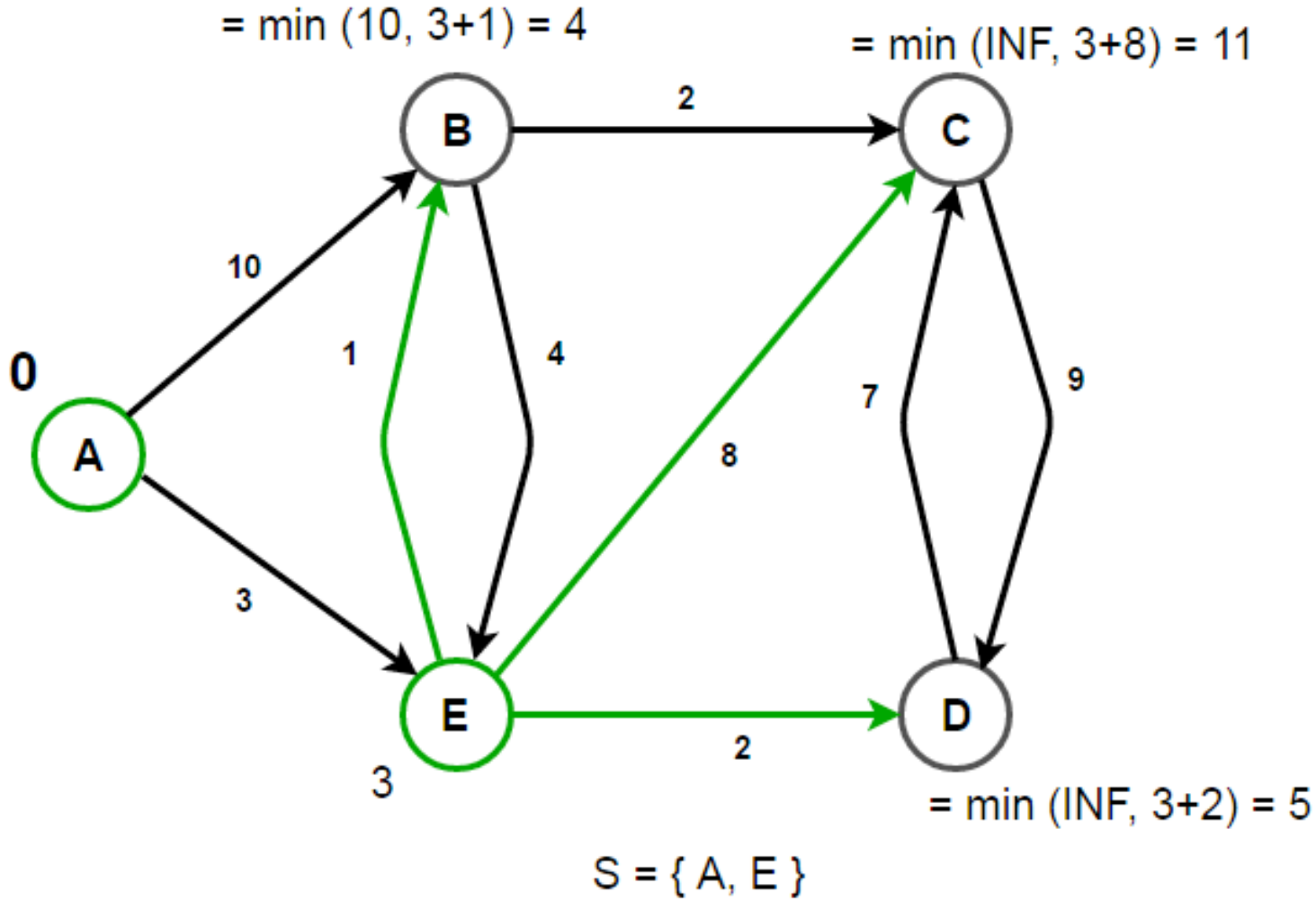


S = { }

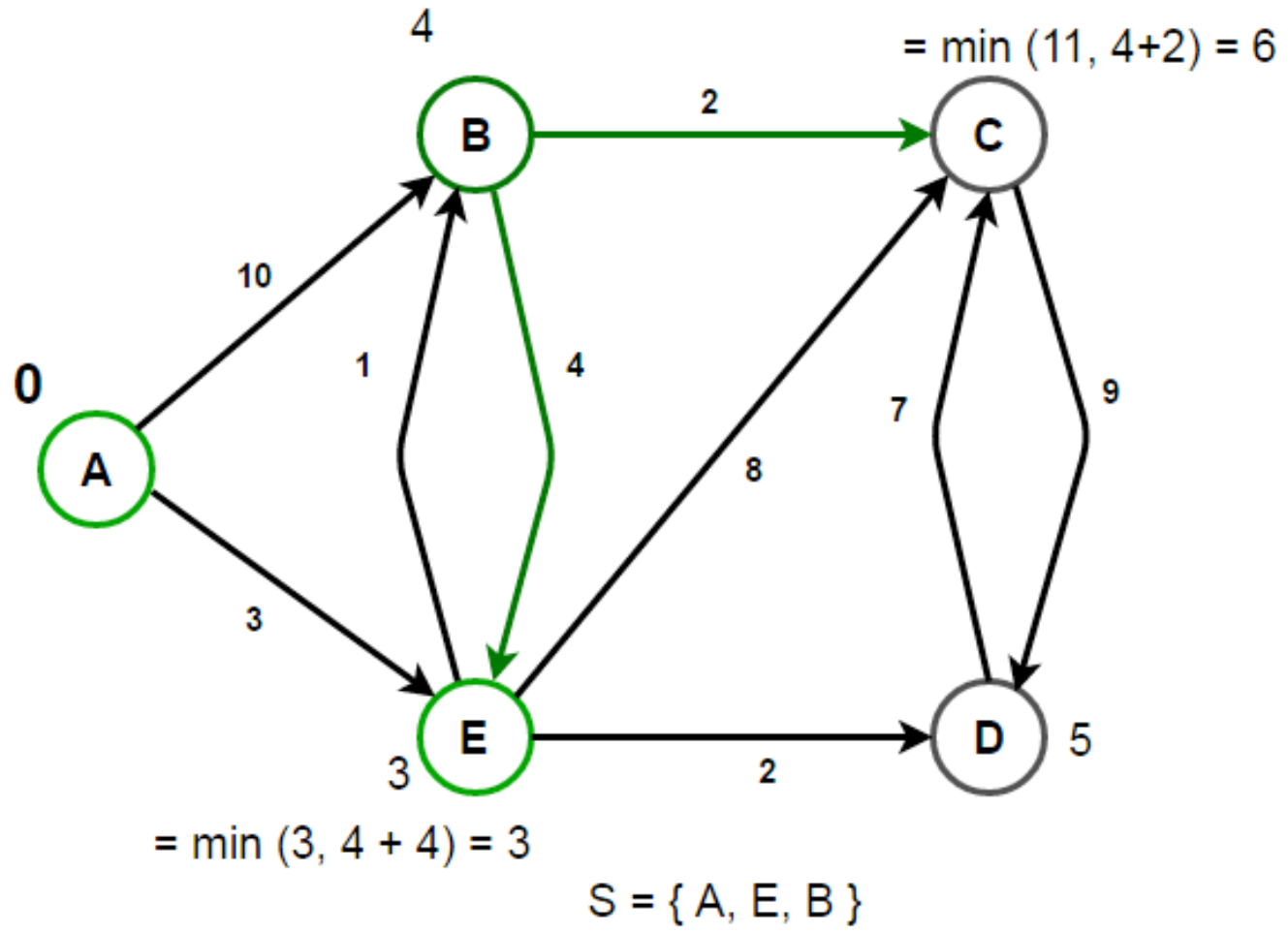
Dijkstra's - Illustration



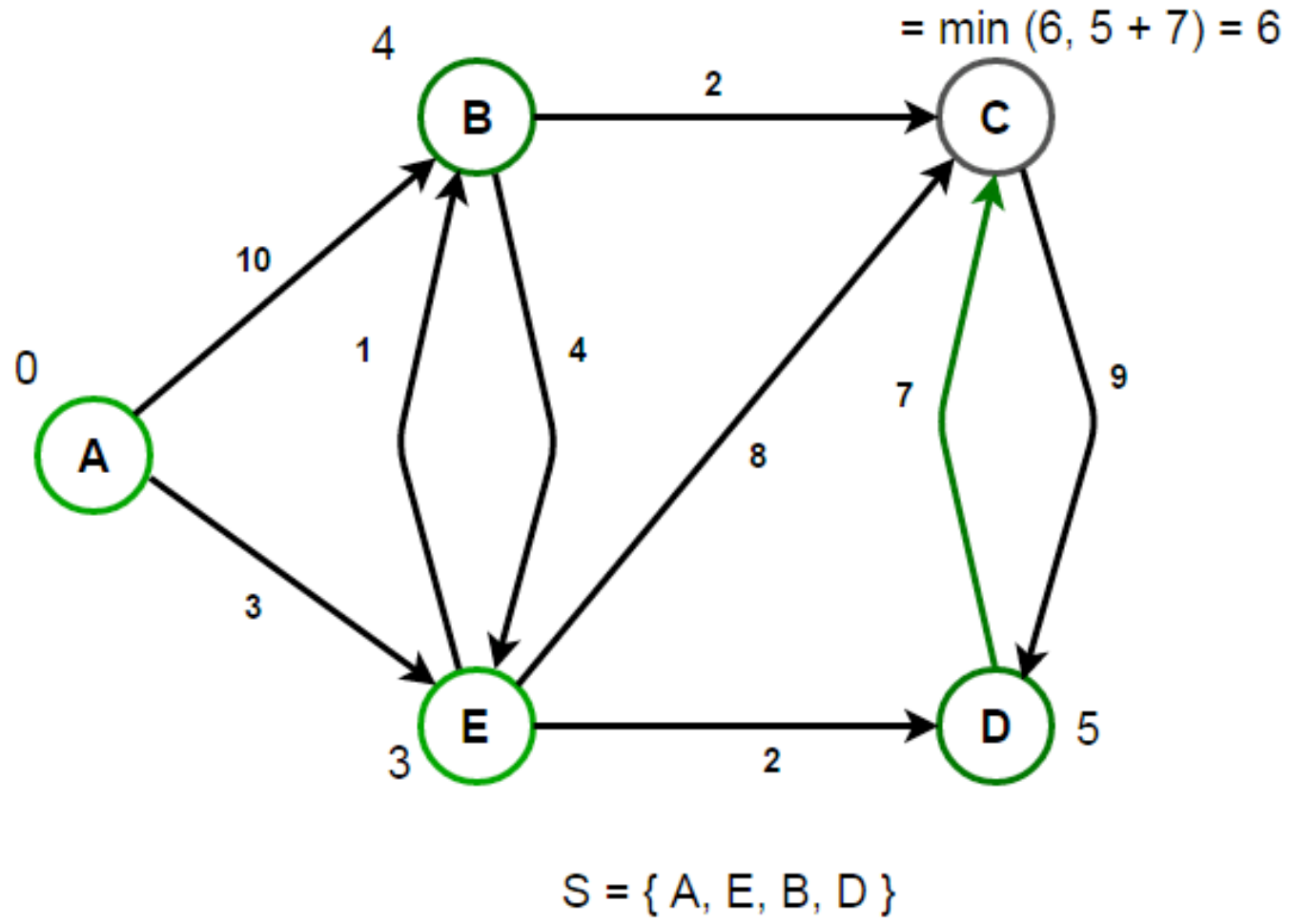
Dijkstra's - Illustration



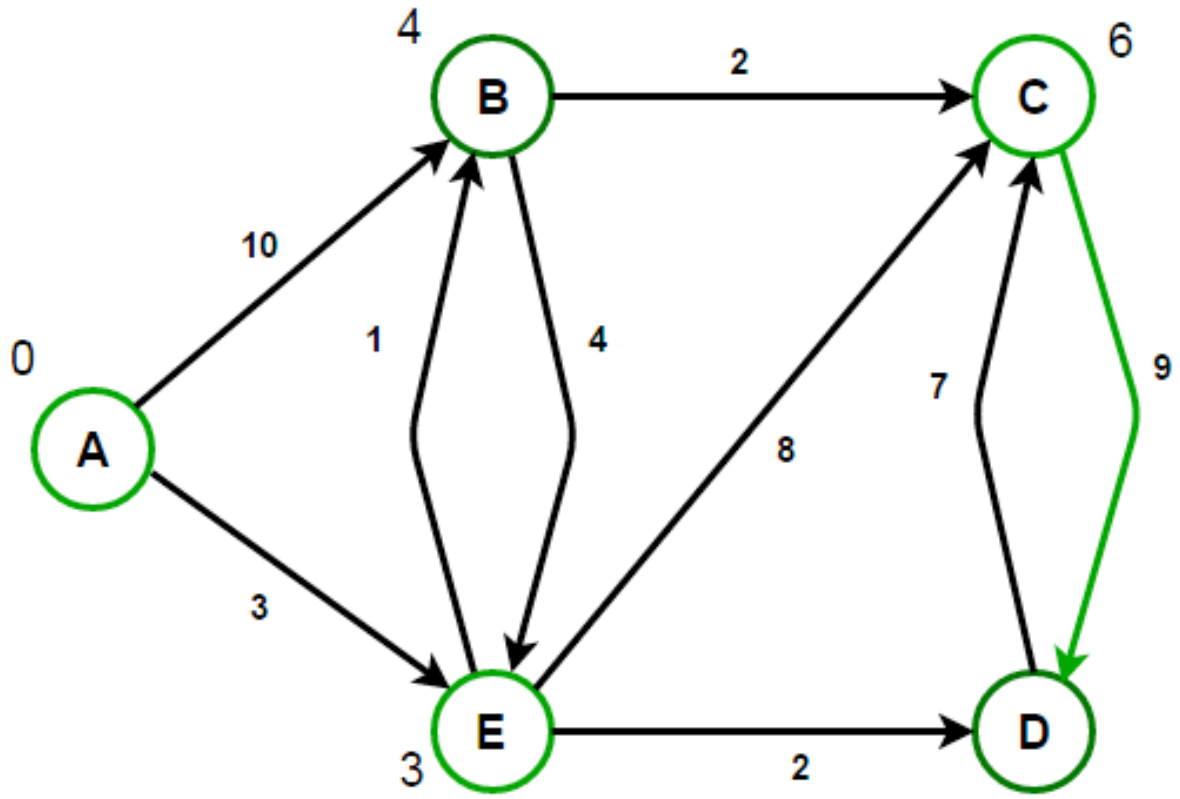
Dijkstra's - Illustration



Dijkstra's - Illustration



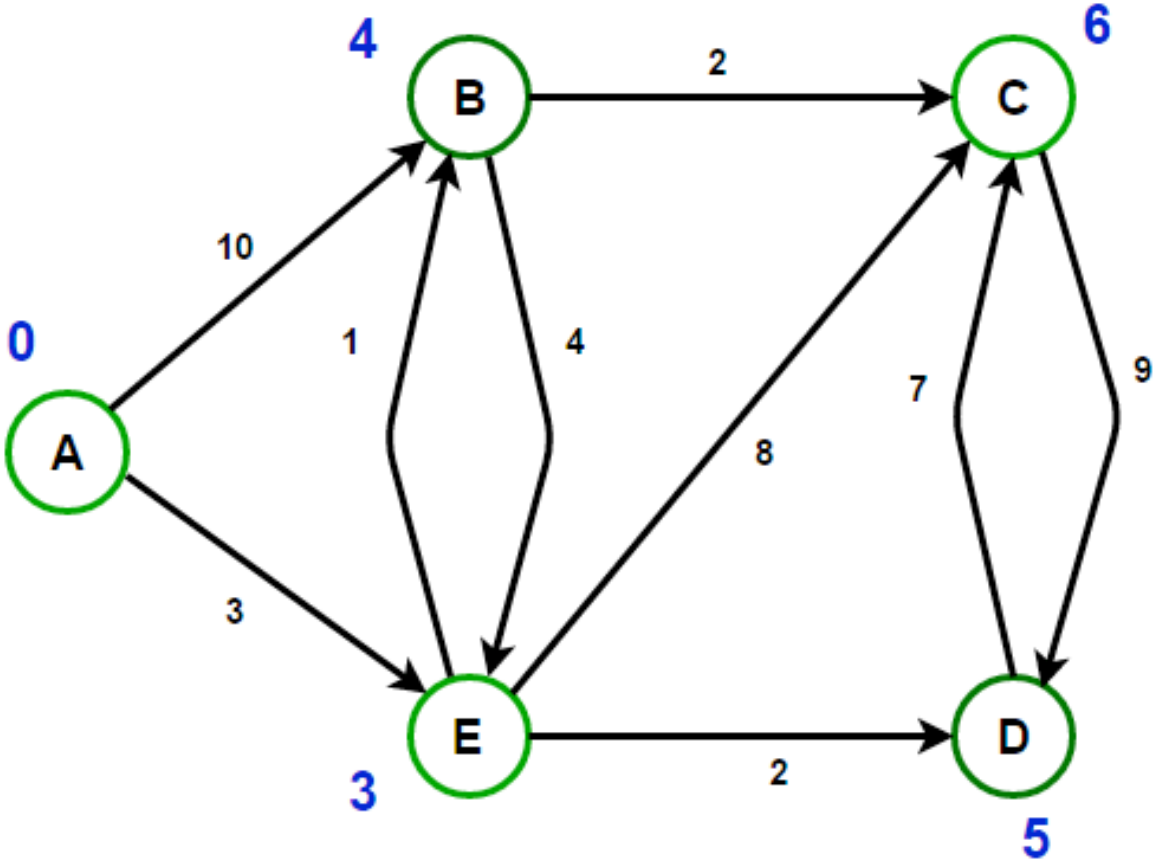
Dijkstra's - Illustration



= min (5, 6 + 9) = 5

S = { A, E, B, D, C }

Dijkstra's - Illustration



Important Points

- Dijkstra's algorithm can be used for directed graphs as well
- This algorithm finds shortest distances from source to all vertices
- Time Complexity of the implementation is $O(V^2)$
  - If the input graph is represented using adjacency list, it can be reduced to O(E log V) with the help of binary heap
- Dijkstra's algorithm doesn't work for graphs with negative weight edges
  - For graphs with negative weight edges:

    - Bellman–Ford algorithm

Complexity

- Worst-case performance

$$(|E| + |V| \log |V|)$$

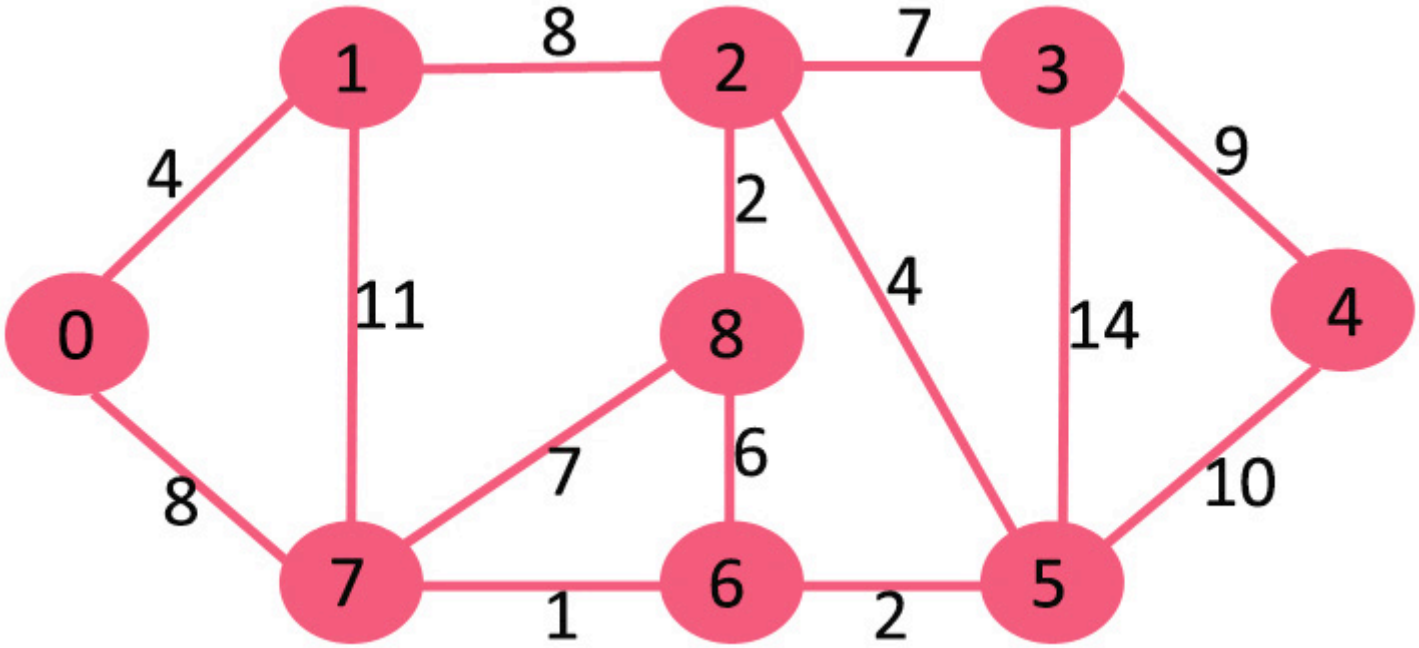
- Why??
  - Visiting all nodes
  - In every chosen node, checking all nodes satisfying the relaxation principle.

- Try Dijkstra's algoithm

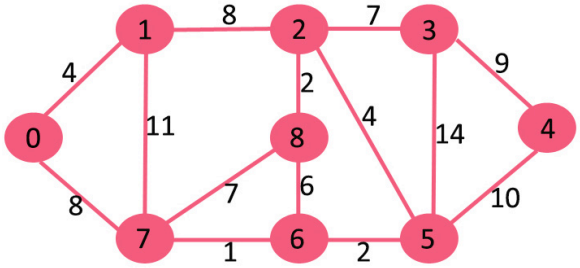
  with a Priority Queue

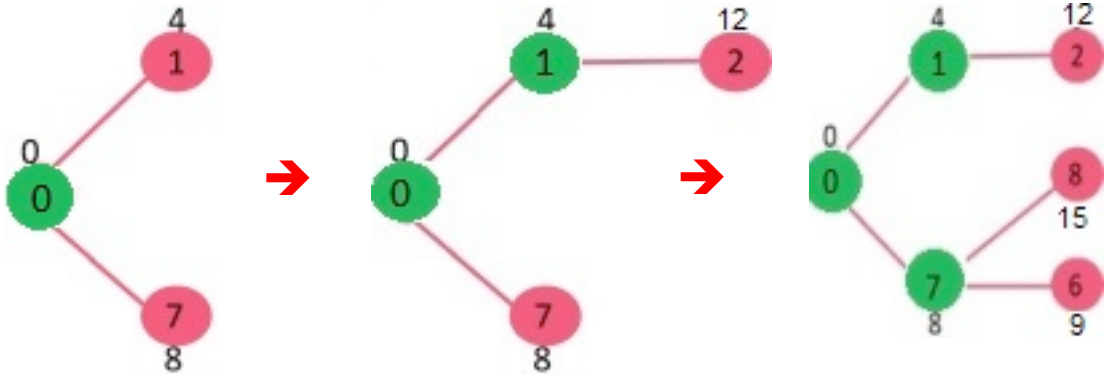
Exercise

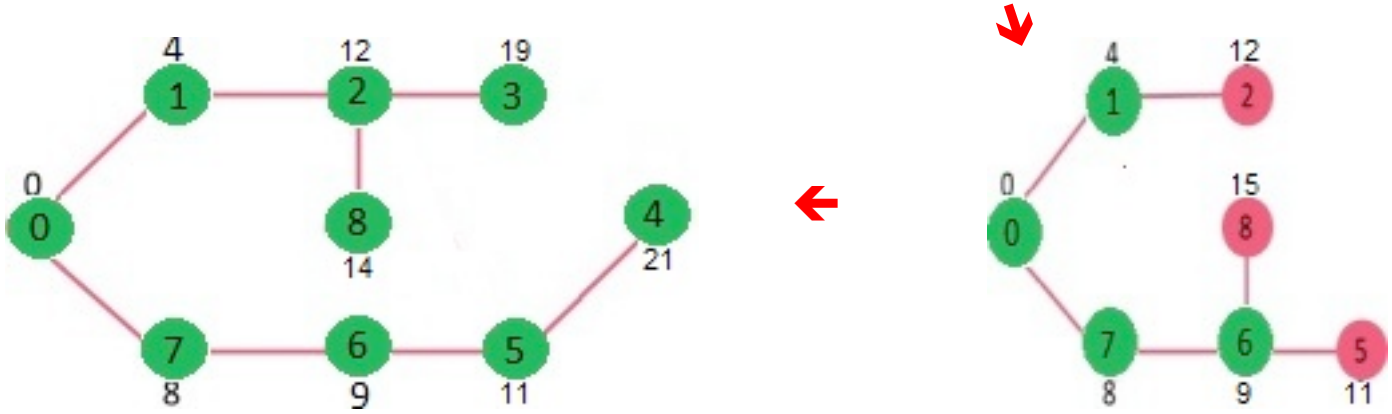
- Apply Dijkstra's Algorithm



Dijkstra's - Exercise







Help among Yourselves?

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

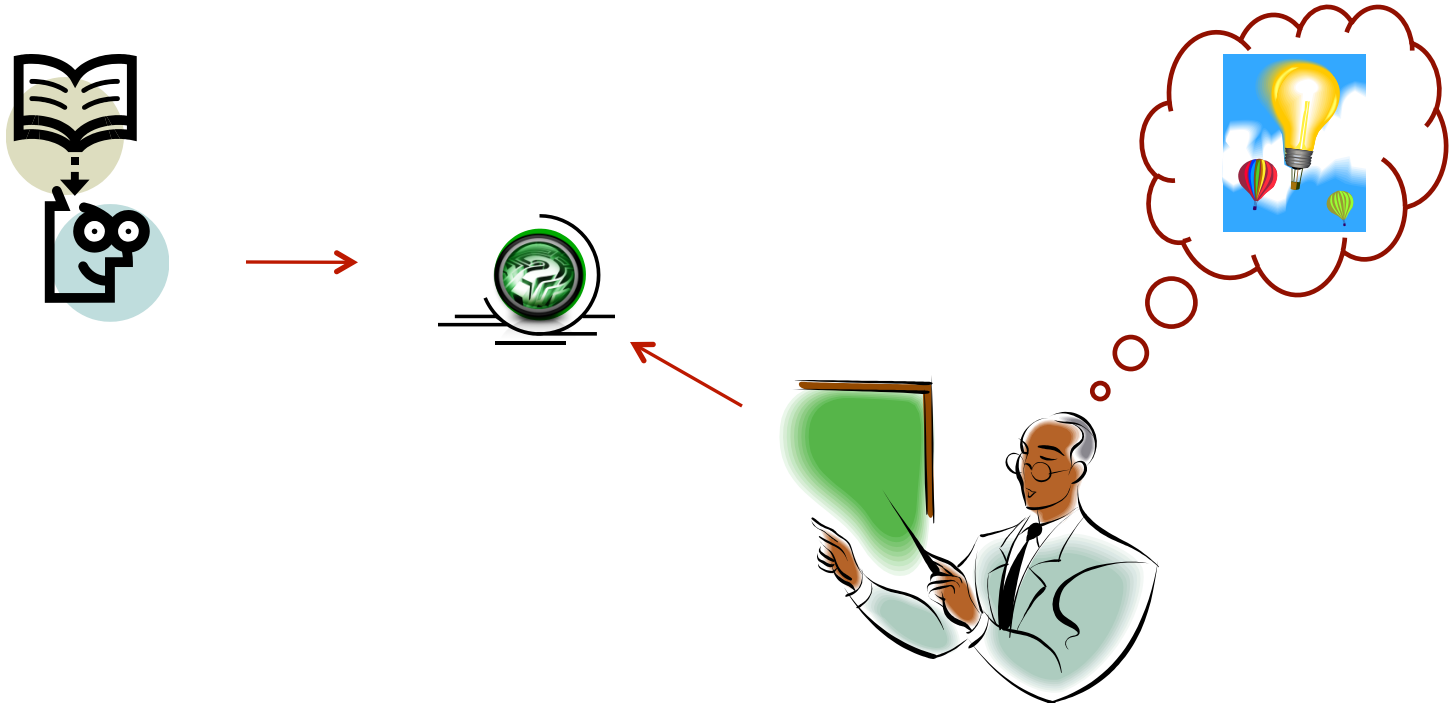
- **Needy Students** (having CGPA less than 6.5)
  - Can the above group help these students? (Your work will also be rewarded)

- 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 ???