Single Source Shortest Path Algorithm – Bellman-Ford

Course: Algorithms

Faculty: Dr. Rajendra Prasath

Autumn 2018

Bellman – Ford Algorithm

This lecture covers the interesting aspects of the single source shortest path algorithm – The popular Bellman – Ford 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

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

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)

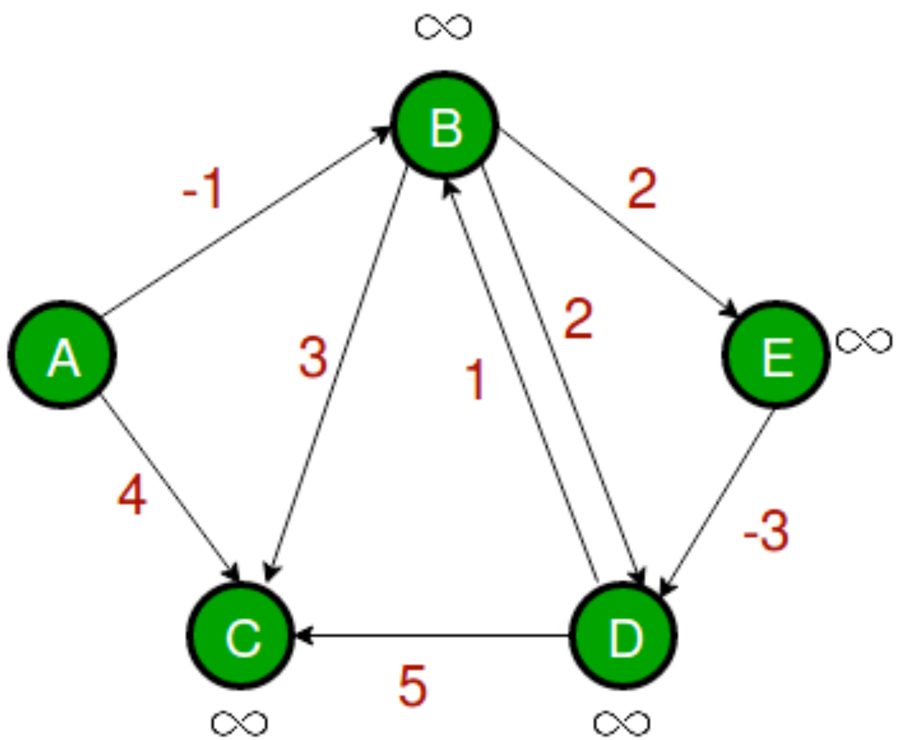
    If (d[u] + c[u,v] < d[v]) then

    d[v] = d[u] + c[u,v]

    How many times do we relax all edges?

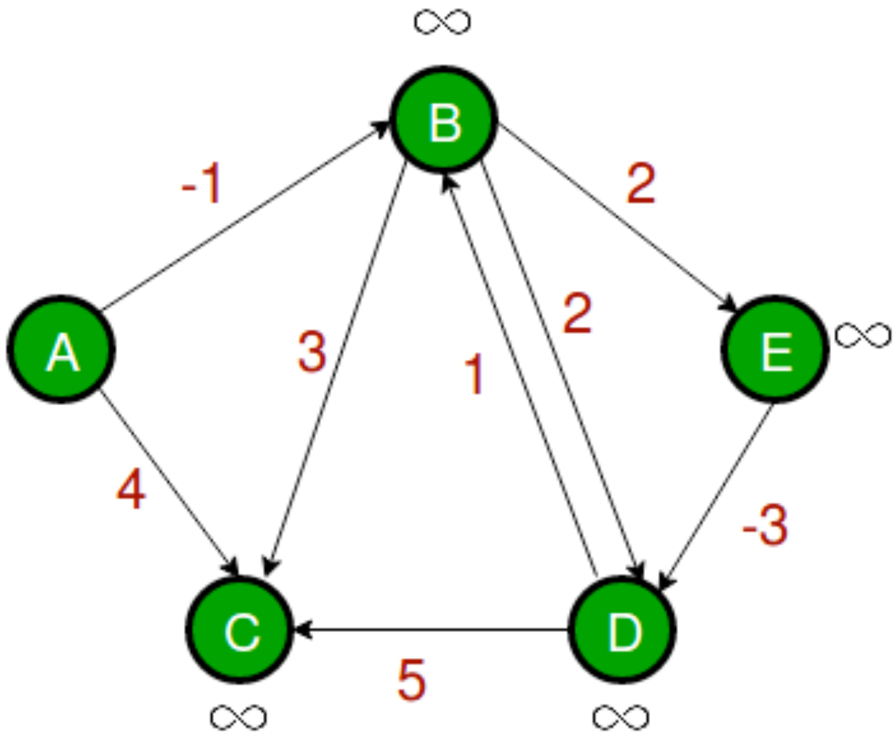
    - (n-1) times (Why?)
    - the longest possible path connecting n edges

Bellman-Ford: Illustration



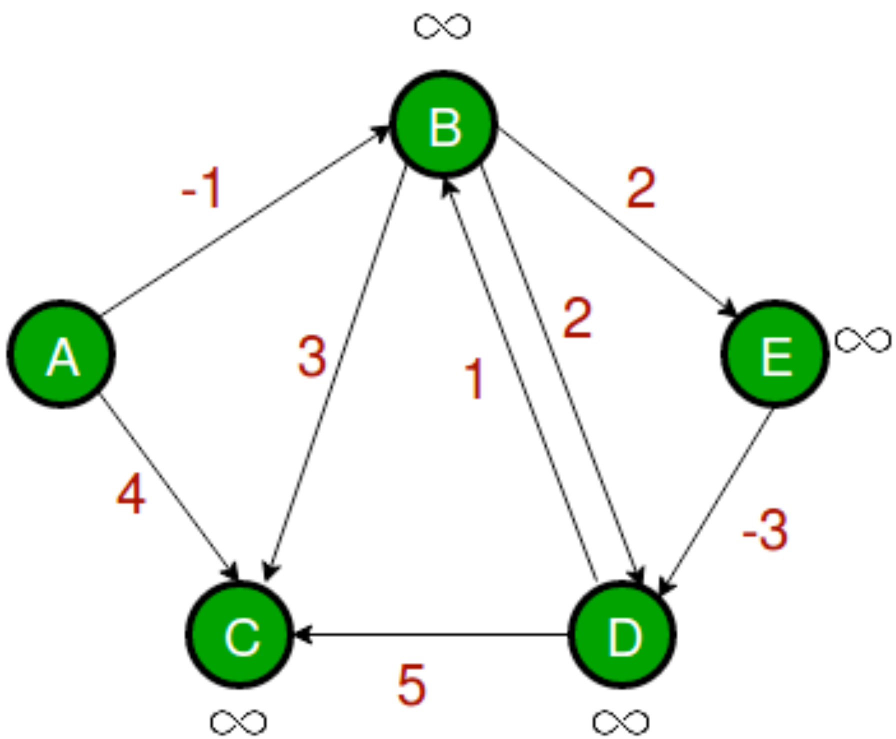
Bellman-Ford: Illustration

A	B	C	D	E
0	∞	∞	∞	∞



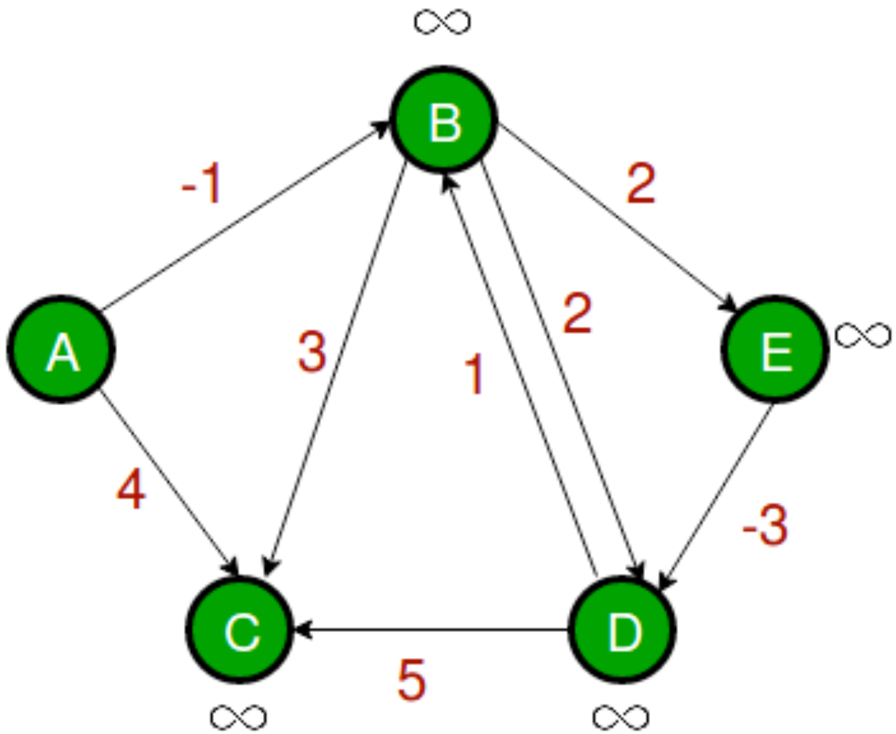
Bellman-Ford: Illustration

A	B	C	D	E
0	∞	∞	∞	∞
0	-1	∞	∞	∞
0	-1	4	∞	∞
0	-1	2	∞	∞



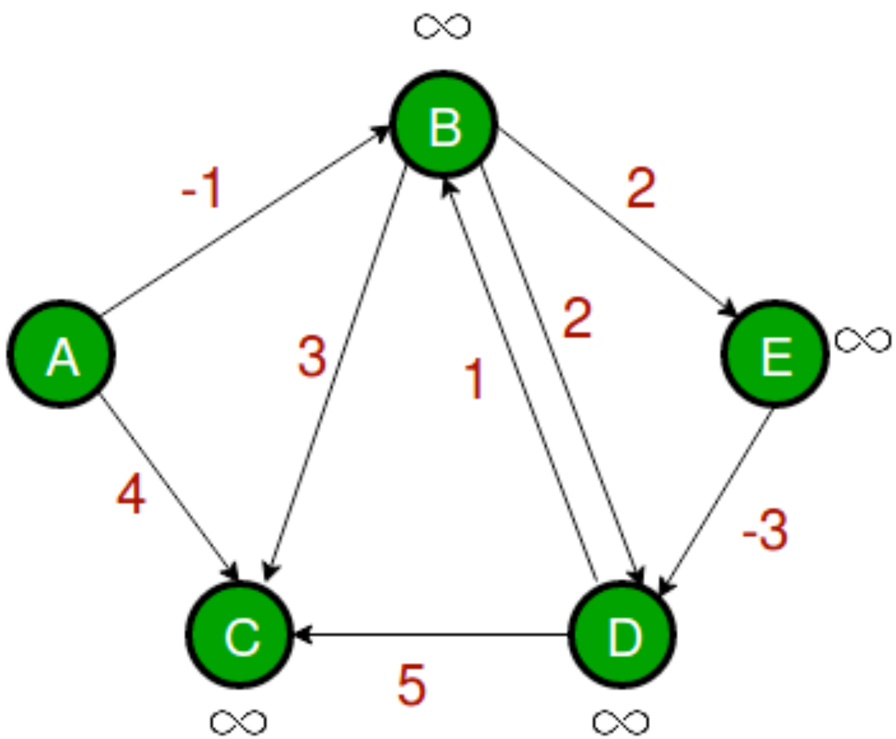
Bellman-Ford: Illustration

A	B	C	D	E
0	∞	∞	∞	∞
0	-1	∞	∞	∞
0	-1	4	∞	∞
0	-1	2	∞	∞
0	-1	2	∞	1
0	-1	2	1	1
0	-1	2	-2	1



Bellman-Ford: Illustration

A	B	C	D	E
0	∞	∞	∞	∞
0	-1	∞	∞	∞
0	-1	4	∞	∞
0	-1	2	∞	∞
0	-1	2	∞	1
0	-1	2	1	1
0	-1	2	-2	1



Bellman-Ford: Complexity

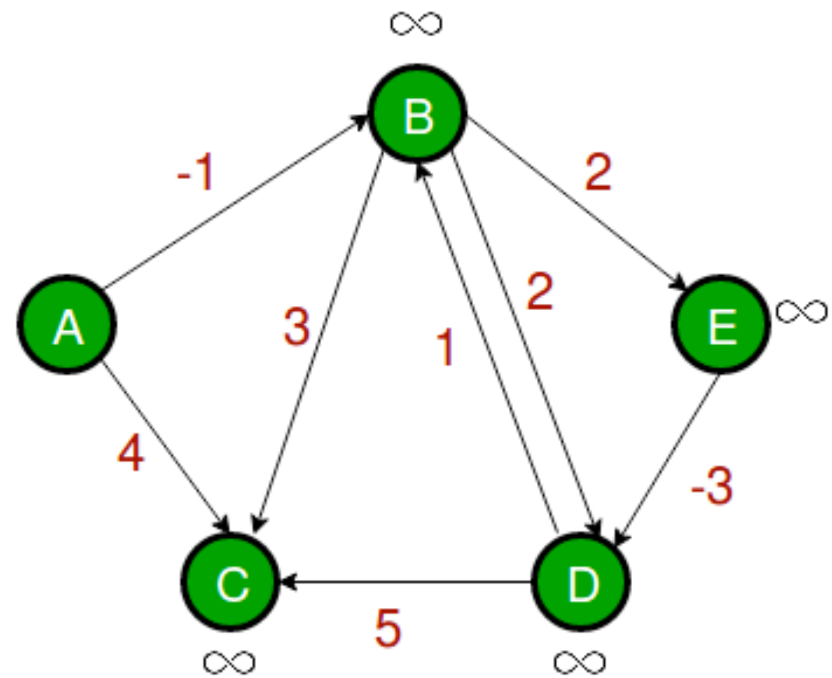
- There are |V| nodes and |E| edges.
- All edges are relaxed upto (|V| - 1) times.
- If |V| = n and |E| = n then

➔ Complexity = $O(n^2)$

- In a complete graph, there are n(n-1)/2 edges
- This leads to

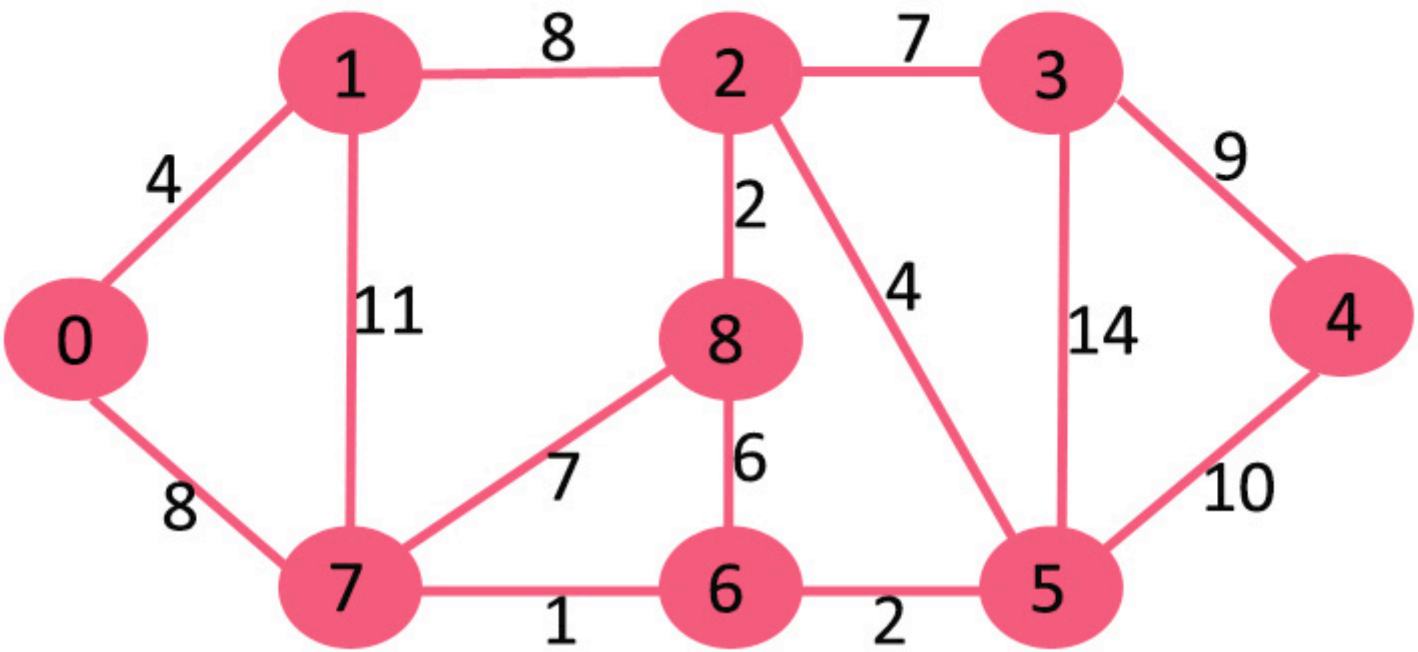
  (n(n-1) / 2) * (n-1)

➔ Complexity = $O(n^3)$



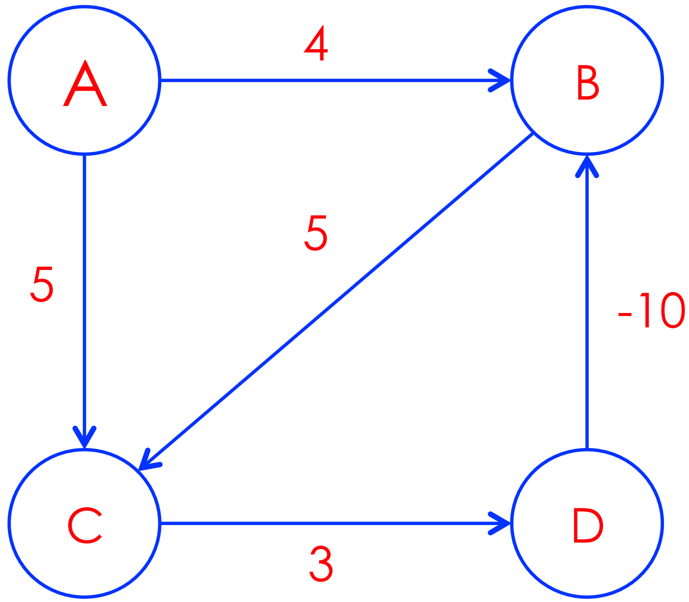
Exercise

- Apply Bellman-Ford Algorithm



Exercise – One More …

- Apply Bellman-Ford Algorithm



Bellman-Ford: Issues

- Bellman-Ford works better (better than Dijksra's for distributed systems

- Dijksra's: finds minimum value of all vertices

  Bellman – Ford: Edges are considered one by one

- Dijksra's: No Negative edge weight considered

  Bellman – Ford: works with Negative edge weights

- Dijksra's: Does not

  Bellman – Ford: Finds cycles in a given graph

Help among Yourselves?

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

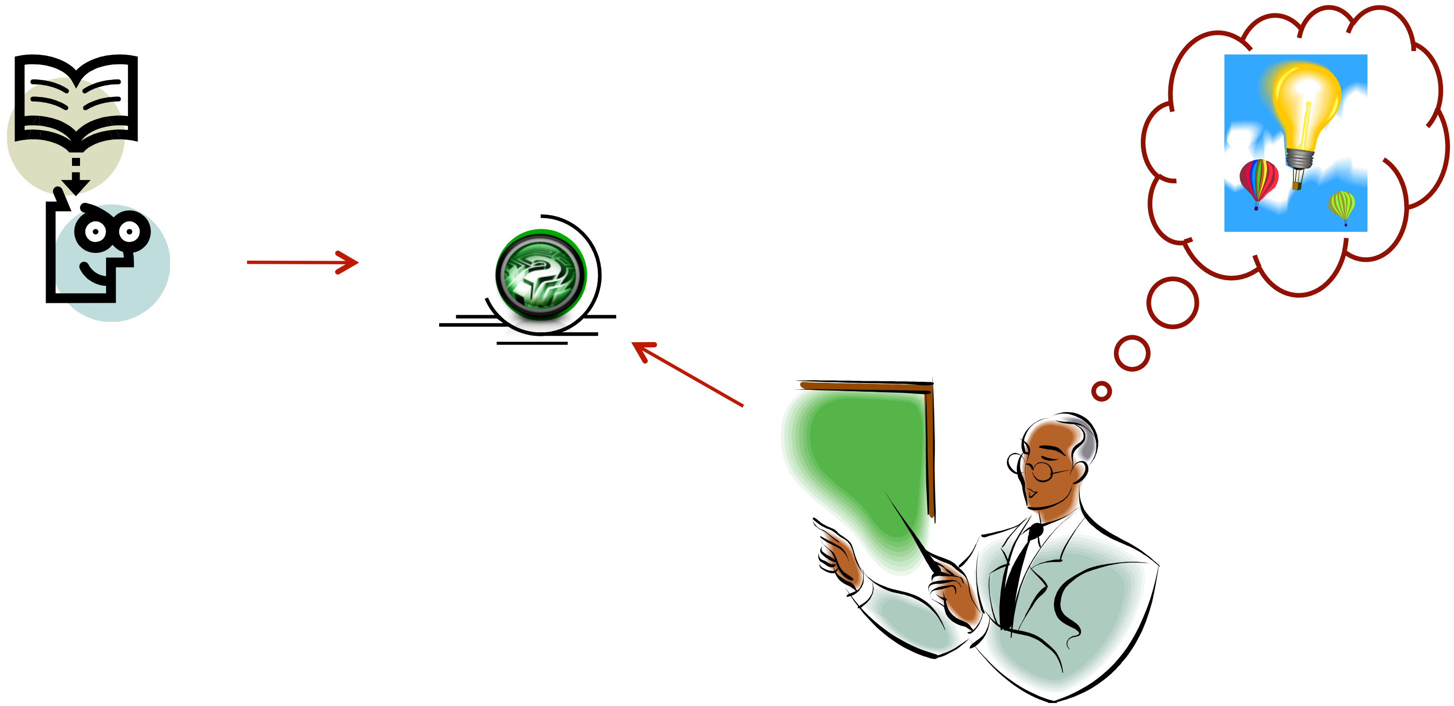
- **Needy Students** (having CGPA less than 6.5)
  - Can the above group help these students? (Your work will also be rewarded)

- 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 ???