**Introduction** **to** **graph** **And** **shortest** **path** **algorithms**

**Members** **of** **group**

Nguyen Duc Minh, Nguyen Van Loc, Nguyen Van Nam

**Instructors**

Nguyen Thi Hong Minh, Pham Huy Thong

**Hanoi** **University** **of** **Science**

1/33

Lecture outline

▶ Graph definitions

▶ Definitions and terminologies

▶ Types of Graphs

▶ Graph ADT

▶ Data Structure for Graphs

▶ Adjacency Matrix

▶ Adjacency List

▶ Edge List

▶ Graph Algorithms and Applications

▶ Shortest path problem

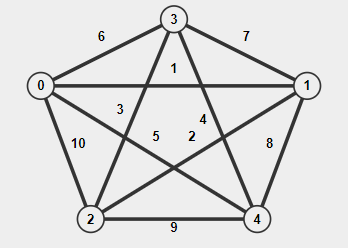
▶ Dijkstra’s Shortest Path Algorithm

▶ Bellman Ford Algorithm

▶ Applications

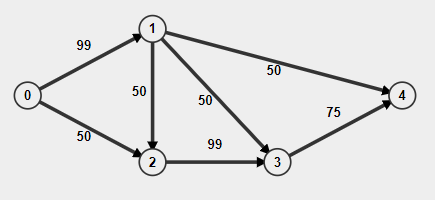
2/33

Graph Definition



– Definition and Terminologies

Example about graphs.



<https://visualgo.net/en/graphds>

3/33

Graph Definition

- Definition and Terminologies

▶ A graph is a collection of **nodes** also called **vertices** which are connected between one another. Each connection between two vertices is called an **edge** (sometimes called a **branch**).

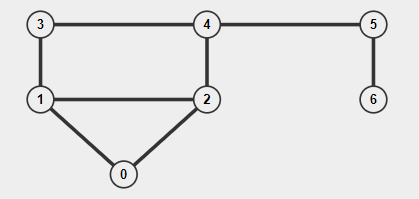
▶ In other words, a graph is an ordered pair **G** **=** **(V,** **E)** where,

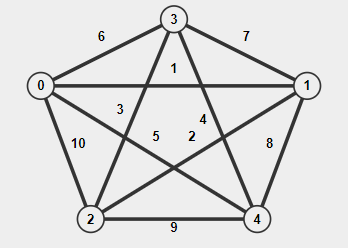
▶ **G** specifies the graph.

▶ **V** is the vertex-set whose elements are called the vertices, or nodes of the graph. This set is often denoted by **V(G)** or just **V**.

▶ **E** is the edge-set whose elements are called the edges, or connections between vertices of the graph. This set is often denoted by **E(G)** or just **E**.

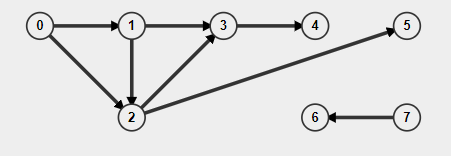
4/33

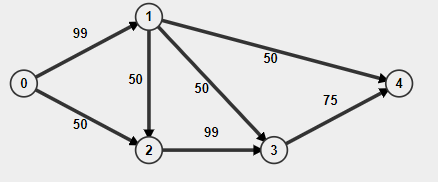
Graph Definition – Types of Graphs



An undirected/unweighted graph An undirected/weighted graph

5/33

Graph Definition – Types of Graphs



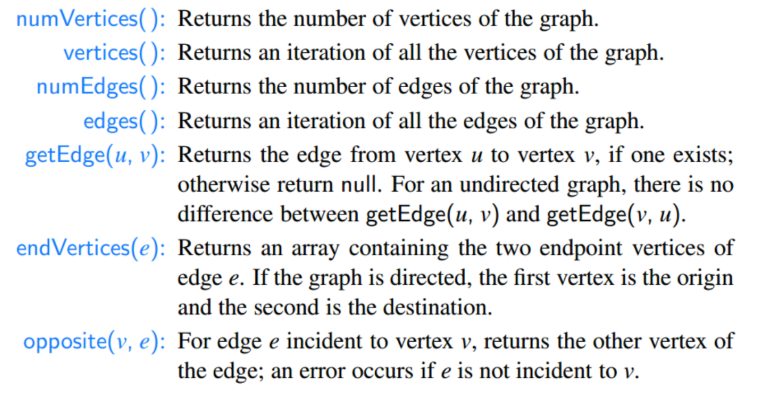
A directed/unweight graph A directed/weight graph

\* The direction of an edge is not always needed. For instance: in a social network like Facebook, there is no need to have directed edges to represent friendship, as if A if a friend of B,

then B is also a friend of A..

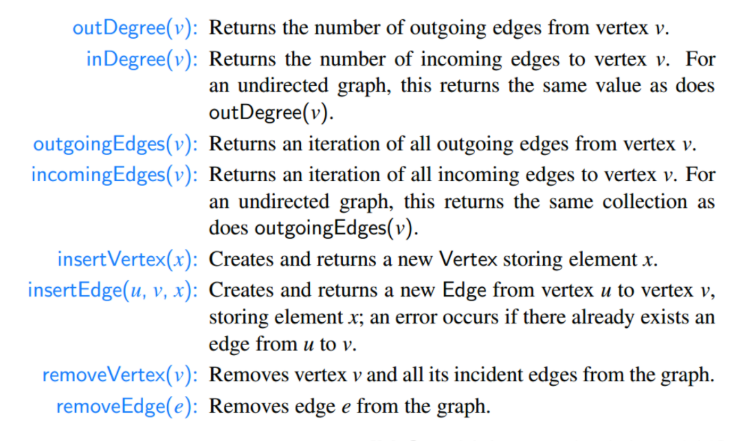
6/33

Graph Definition - Graph ADT



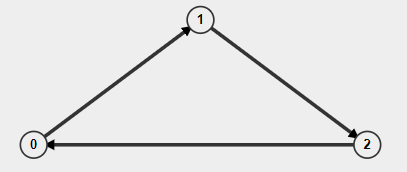
[M.Goodrich, sec. 14.1.1, p. 618] 7/33

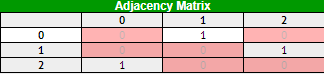
Graph Definition - Graph ADT



8/33

Data Structure for Graphs - Adjacency Matrix

Adjacency Matrix is a 2D array of size V x V where V is the number of vertices in a graph. Adjacency matrix for undirected graph is always symmetric. Adjacency Matrix is also used to represent weighted graphs. If adj[i][j] = w, then there is an edge from vertex i to vertex j with weight w, if w = 0 there is no edge.

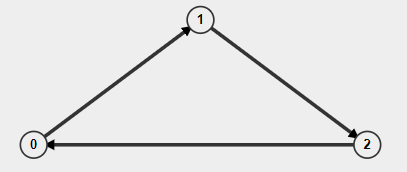
Pros: Representation is easier to implement and follow. Removing an edge takes O(1) time. Queries like whether there is an edge from vertex ‘u’ to vertex ‘v’ are efficient and can be done O (1).

Cons: Consumes more space O(V2). Even if the graph is sparse (contains less number of edges), it consumes the same space. Adding a vertex is O(V2) time.

<https://visualgo.net/en/graphds>

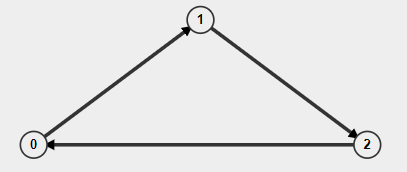
<https://en.wikipedia.org/wiki/Adjacency_matrix>9/33

Data Structure for Graphs - Adjacency List

In graph theory and computer science, an adjacency list is a collection of unordered lists used to represent a finite graph. Each list describes the set of neighbors of a vertex in the graph. This is one of several commonly used representations of graphs for use in computer programs.

<https://en.wikipedia.org/wiki/Adjacency_list>10/33

Data Structure for Graphs - Edge List

An edge list is a data structure used to represent a graph as a list of its edges. An (unweighted) edge is defined by its start and end vertex, so each edge may be represented by two numbers. The entire edge list may be represented as a two-column matrix. If weighted edges, add a column is weight of edges. An edge list may be considered a variation on an adjacency list which is

represented as a length |V| array of lists. Since each edge contains just two or three numbers.

<https://en.wikipedia.org/wiki/Edge_list>

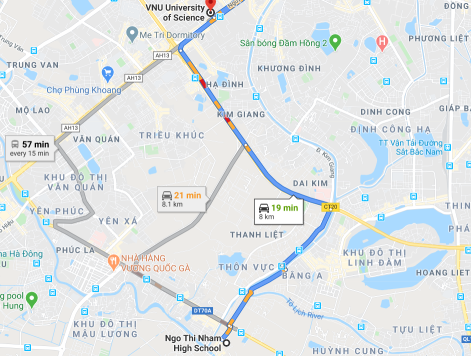
11/33

Graph Algorithms and Applications - Shortest Path Problem

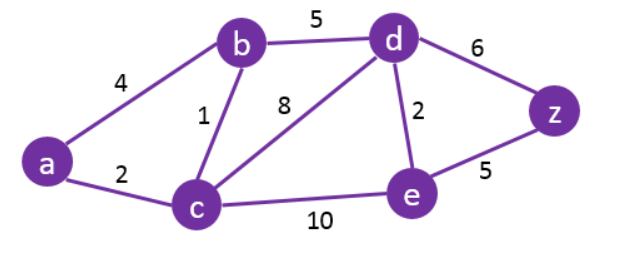
▶ Definication: the **shortest** **path** **problem** is the problem of finding

a **path** between two **vertices** (or nodes) in a graph such that the sum of the **weights** of its constituent edges is **minimized**.

▶ For instance: The problem of finding the shortest path between two intersections on a road map may be modeled as a special case of the shortest path problem in graphs, where the vertices correspond to intersections and the edges correspond to road segments, each weighted by the length of the segment.

12/33

Shortest Path Problem - Dijkstra Algorithm

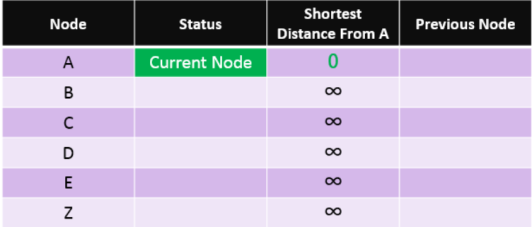
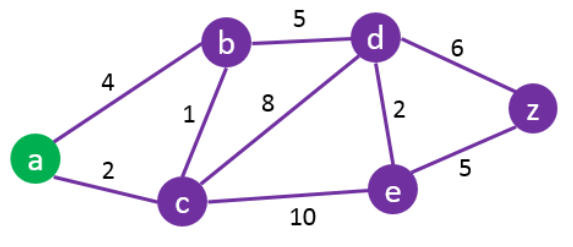


▶ Method: **Dijkstra's** **algorithm** to find the shortest path between vertex in a graph(purple). It picks the unvisited vertex with the lowest distance

(green), calculates the distance through it to each unvisited neighbor(blue), and updates the neighbor's distance if smaller. Mark visited (gray) when done with neighbors.

What is the shortest path to travel from A to Z ? 13/33

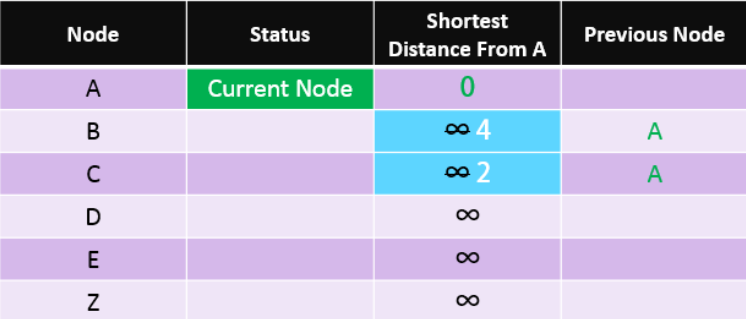
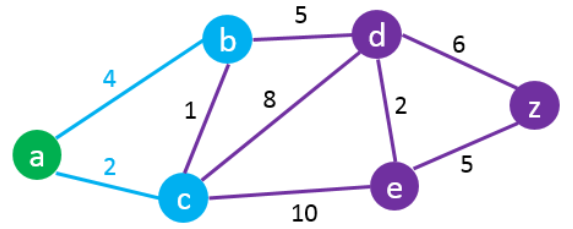
Shortest Path Problem - Dijkstra Algorithm



Start by setting the starting node (A) as the current node.

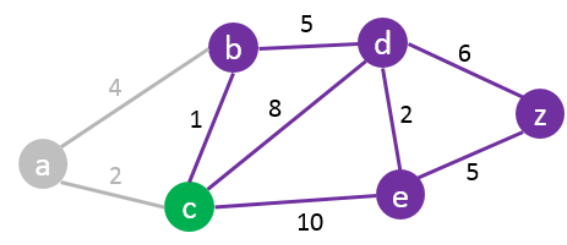
14/33

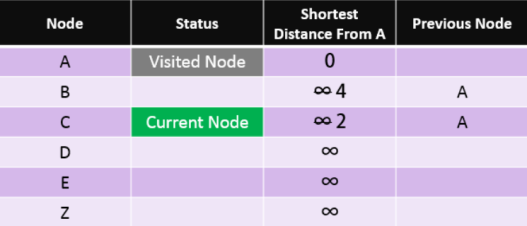
Shortest Path Problem - Dijkstra Algorithm



Check all the nodes connected to A and update their **“Distance** **from** **A”** and set their **“previous** **node”** to “A”.

15/33

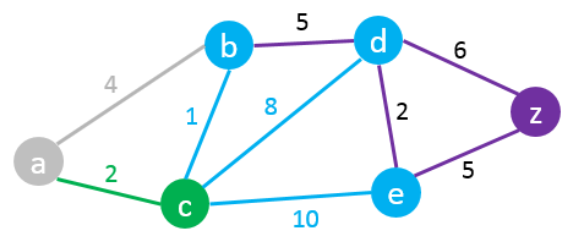
Shortest Path Problem - Dijkstra Algorithm

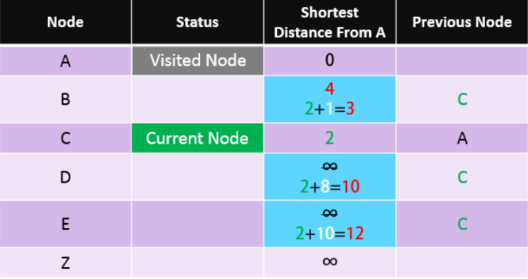


Set the current node (A) to **“visited”** and use the closest unvisited node to A as the **current** **node**

(e.g. in this case: Node C).

16/33

Shortest Path Problem - Dijkstra Algorithm

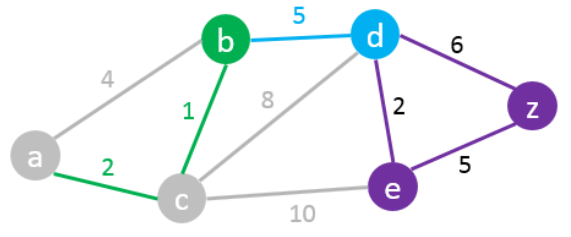


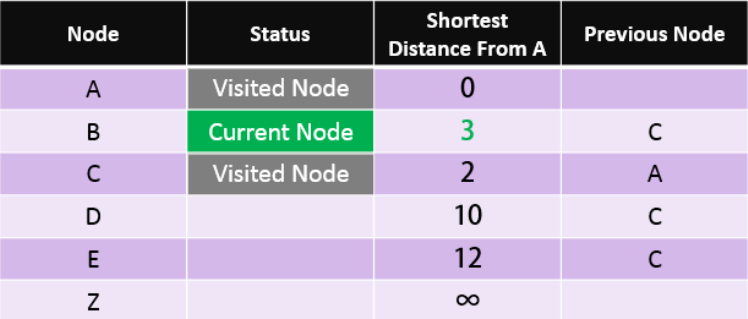
Check all unvisited nodes connected to the current node and add the distance from A to C to all distances from the connected nodes. Replace their values only if the new distance is lower than the previous one.

C -> B: 2 + 1 = 3 < 4 – Change Node B C -> D: 2 + 8 = 10 < ∞ – Change Node D

C -> E: 2 + 10 = 12 < ∞ – Change Node E

17/33

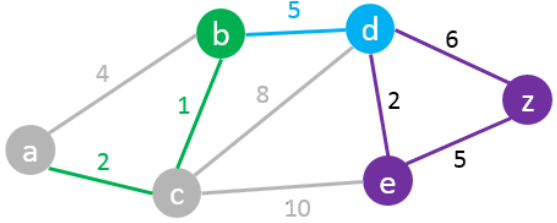
Shortest Path Problem - Dijkstra Algorithm

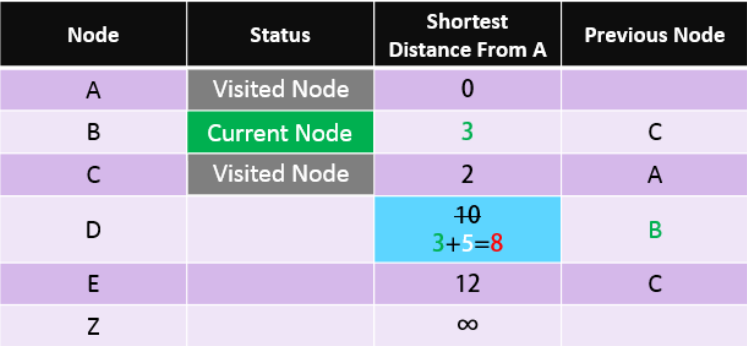


Set the current node C status to Visited. We then repeat the same process always

picking the closest unvisited node to A as the current node. In this case node B becomes the current node.

18/33

Shortest Path Problem - Dijkstra Algorithm

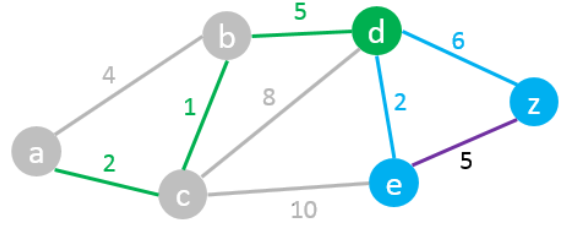


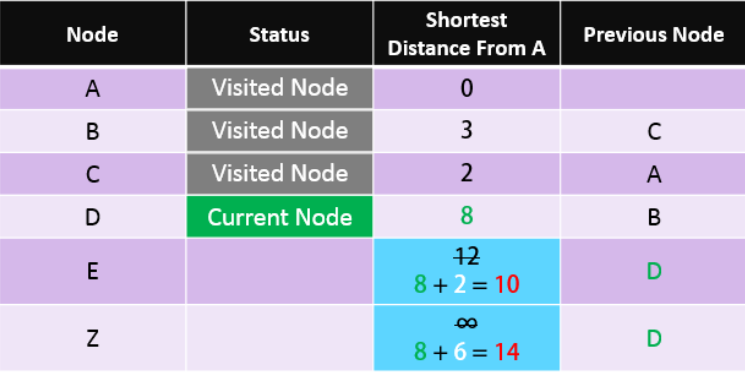
B -> D 3+5 = 8 < 10 – Change Node D

Next “Current Node” will be D as it has the

shortest distance from A amongst all unvisited nodes.

19/33

Shortest Path Problem - Dijkstra Algorithm

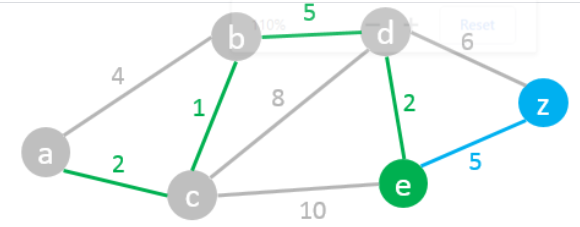


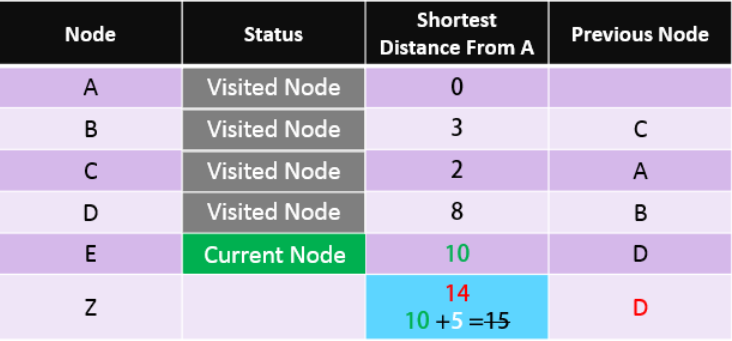
D -> E 8+2 = 10 < 12 – Change Node E D -> Z 8+6 = 14 < ∞ – Change Node Z

We found a path from A to Z but it may not be the shortest one yet. So we need to carry on the process.

Next **“Current** **Node”**: E

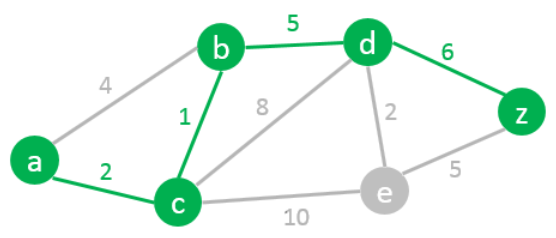
20/33

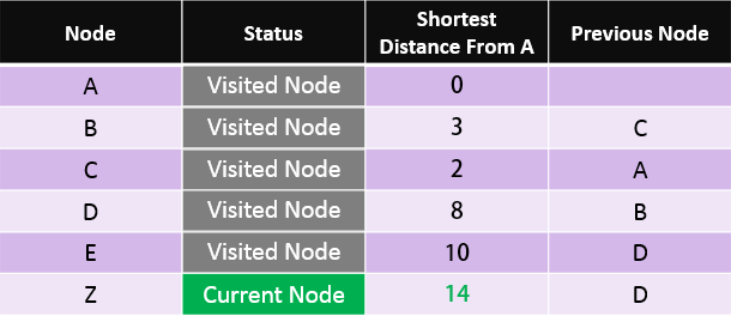
Shortest Path Problem - Dijkstra Algorithm



E -> Z 10+5 = 15 > 14 – We do **not** change node Z.

21/33

Shortest Path Problem - Dijkstra Algorithm



We found the shortest path from A to Z.

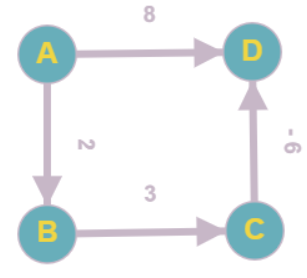
Read the path from Z to A using the previous node column: Z > D > B > C > A

So the Shortest Path is:

**A** **–** **C** **–** **B** **–** **D** **–** **Z** **with** **a** **length** **of** **14**

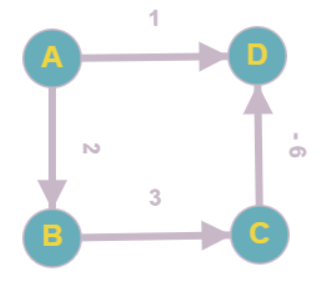
22/33

Dijkstra Algorithm



– Example about Negative Weight Edges

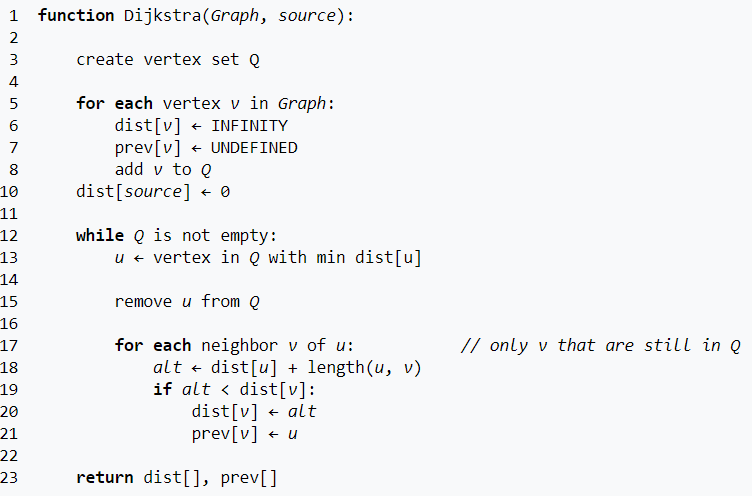
Not work Work



-> Conclusion: Dijkstra Algorithm may not work with negative weight edges

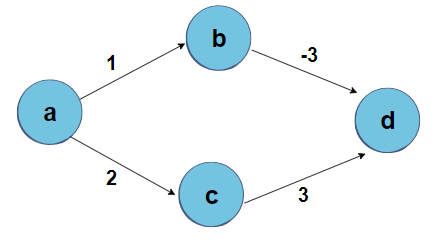
23/33

Shortest Path Problem - Dijkstra Algorithm



Time Complexity: O(n2) (n = V) Time Complexity: O(n2) (n = V)

24/33

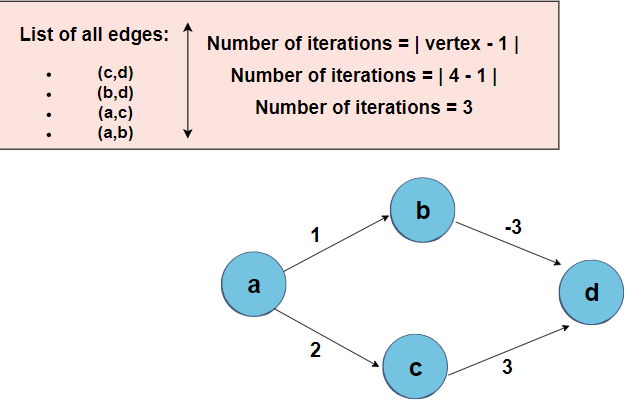
Shortest Path Problem

- Bellman Ford Algorithm

▶ Definication: Like Dijkstra’s shortest path algorithm, the **Bellman-Ford** **algorithm** is guaranteed to find the shortest path in a graph. Though it is slower than Dijkstra's algorithm, Bellman-Ford is capable of handling graphs that contain negative edge weights, so it is more versatile.

25/33

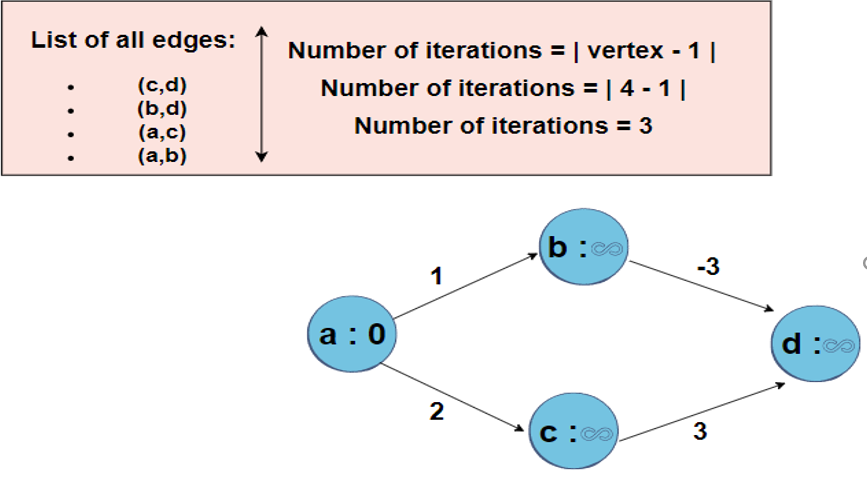
Shortest Path Problem



- Bellman Ford Algorithm

26/33

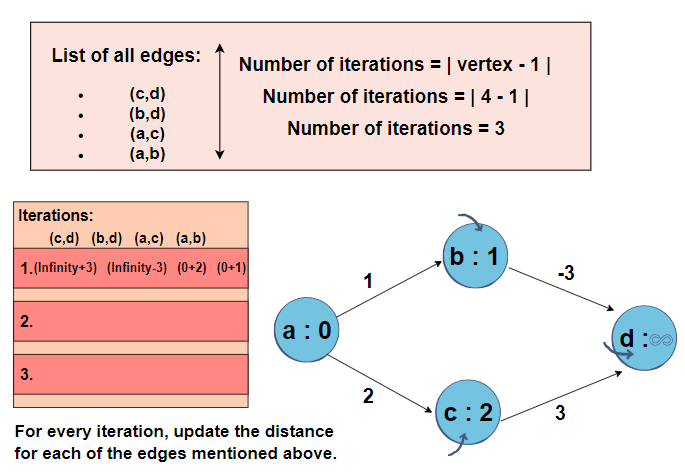
Shortest Path Problem



- Bellman Ford Algorithm

Initially, make the starting point with zero and the other points with infinity 27/33

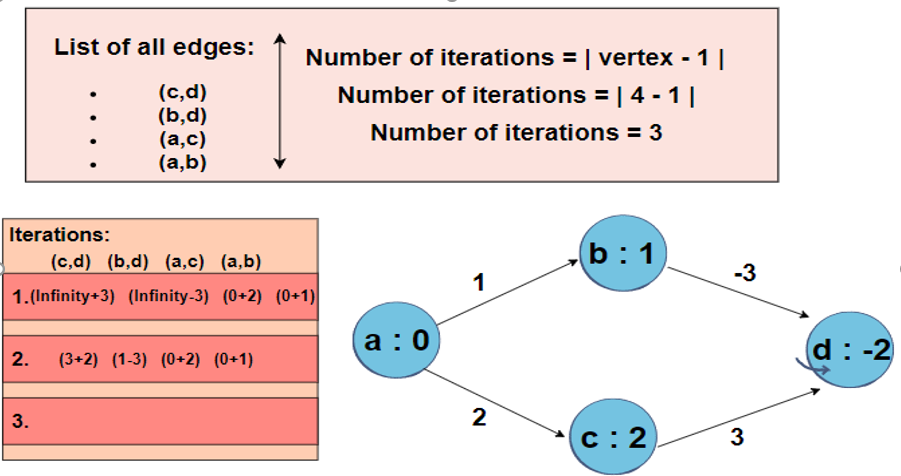
Shortest Path Problem



- Bellman Ford Algorithm

28/33

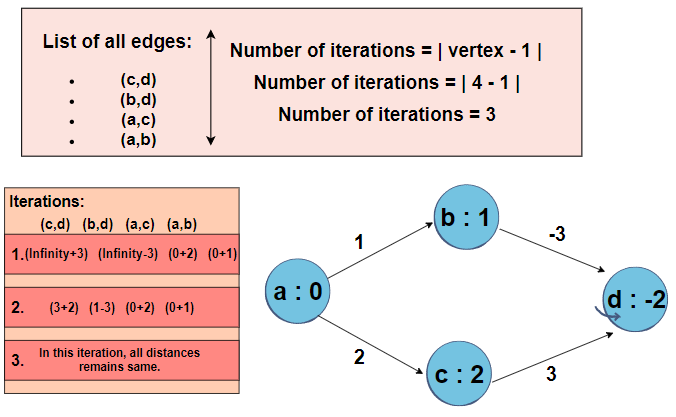
Shortest Path Problem



- Bellman Ford Algorithm

As the shortest distance will be opted for, the distance of D will be equal to -2 instead of 5. 29/33

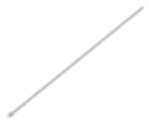
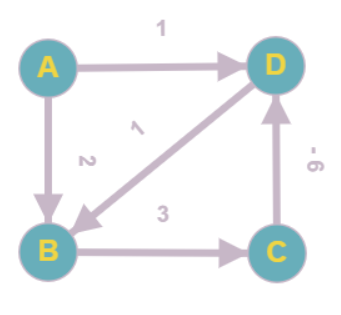
Shortest Path Problem

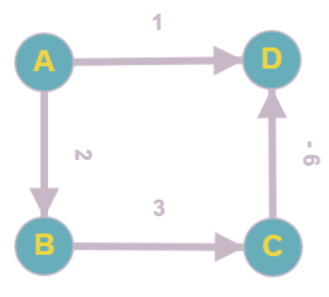


- Bellman Ford Algorithm

30/33

Bellman Ford Algorithm – Example about Negative Weight Edges and Negative Weight Cycles



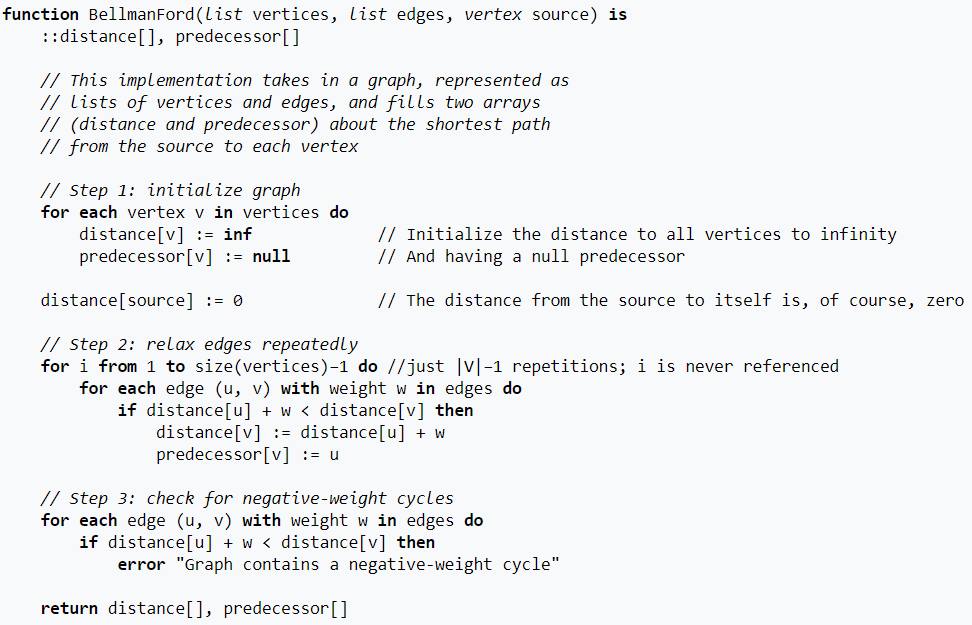
negative weight cycles

Work Not work

-> Conclusion: Bellman Ford Algorithm may work with negative weight edges but may not work with negative weight cycles

<https://www.youtube.com/watch?v=2raV0H9KqY8&list=PLDN4rrl48XKpZkf03iYFl-O29szjTrs_O&index=80>31/33

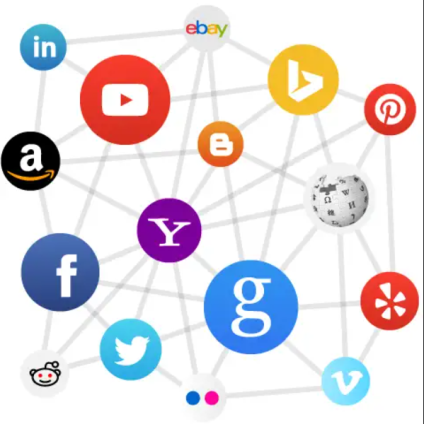
Shortest Path Problem



- Bellman Ford Algorithm

Time Complexity: O(n3) (n = V) 32/33

Graph Algorithms and Applications - Applications



Neural Networks Social Network

**Application** **of** **Graphs:**

**Computer** **Science:** networks of communication, data organization, computational devices etc. **Physics** **and** **Chemistry**: Graph theory is also used to study molecules in chemistry and physics. **Mathematics**: In this, graphs are useful in geometry.

**Biology**: Graph theory is useful in biology … 33/33