

DATA STRUCTURE AND ALGORITHMS

LECTURE 8

Graph

Reference links:

<https://cs.nyu.edu/courses/fall17/CSCI-UA.0102-007/notes.php>

<https://www.comp.nus.edu.sg/~stevenha/cs2040.html>

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

<https://graphonline.ru/>

[M.Goodrich, chapter 14]

Lecture outline

- ❑ Graph definitions
 - Definitions and terminologies
 - Graph ADT
- ❑ Data Structure for Graphs
 - Adjacency Matrix – Ma trận kề
 - Adjacency List – Danh sách kề
 - Edge List – Danh sách cạnh
- ❑ Graph Algorithms and Applications
 - *So many*

Lecture outline

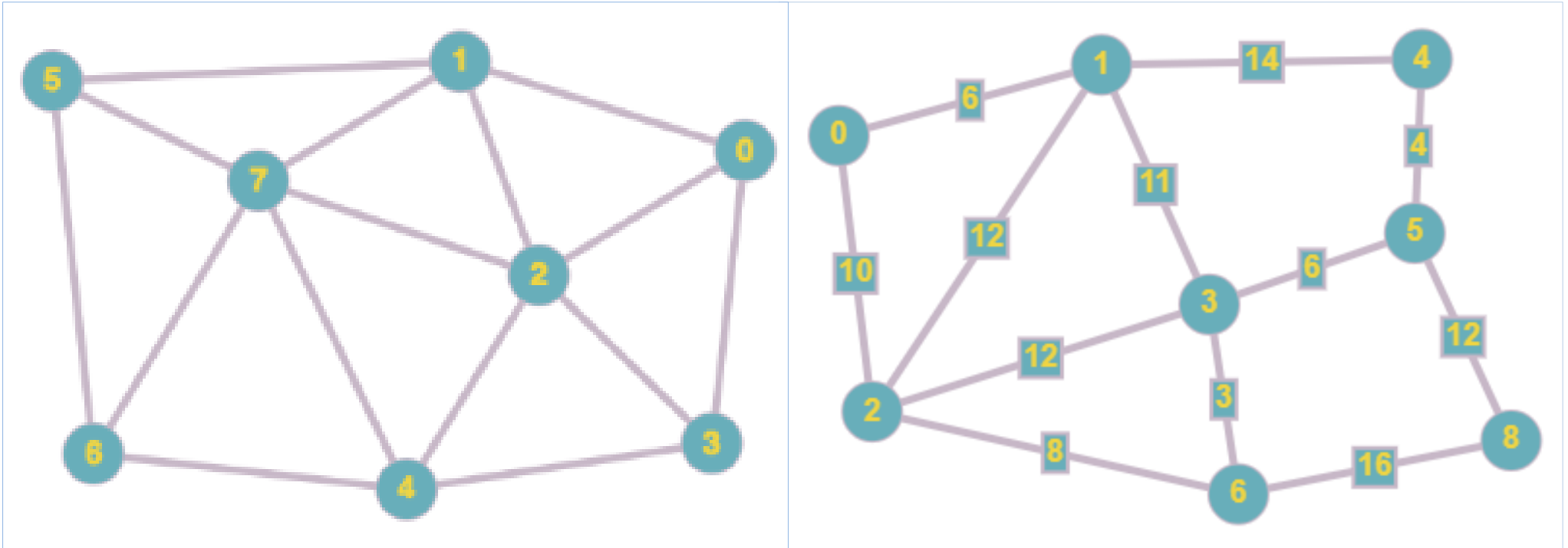
□ Method to study

- Listen to the skim lecture in class
- Read book chapter and provided documents
- Run illustration programs
- Choose favorite algorithms and applications
- Code

Graph

- Definition and terminologies
 - Graph ADT
-

Graph definitions & terminologies



Planar graph examples

<https://visualgo.net/en/graphds?slide=1>

Graph ADT

`numVertices()`: Returns the number of vertices of the graph.

`vertices()`: Returns an iteration of all the vertices of the graph.

`numEdges()`: Returns the number of edges of the graph.

`edges()`: Returns an iteration of all the edges of the graph.

`getEdge(u, v)`: Returns the edge from vertex u to vertex v , if one exists; otherwise return null. For an undirected graph, there is no difference between `getEdge(u, v)` and `getEdge(v, u)`.

`endVertices(e)`: Returns an array containing the two endpoint vertices of edge e . If the graph is directed, the first vertex is the origin and the second is the destination.

`opposite(v, e)`: For edge e incident to vertex v , returns the other vertex of the edge; an error occurs if e is not incident to v .

[M.Goodrich, sec. 14.1.1, p. 618]

Graph ADT

`outDegree(v)`: Returns the number of outgoing edges from vertex v .

`inDegree(v)`: Returns the number of incoming edges to vertex v . For an undirected graph, this returns the same value as does `outDegree(v)`.

`outgoingEdges(v)`: Returns an iteration of all outgoing edges from vertex v .

`incomingEdges(v)`: Returns an iteration of all incoming edges to vertex v . For an undirected graph, this returns the same collection as does `outgoingEdges(v)`.

`insertVertex(x)`: Creates and returns a new Vertex storing element x .

`insertEdge(u, v, x)`: Creates and returns a new Edge from vertex u to vertex v , storing element x ; an error occurs if there already exists an edge from u to v .

`removeVertex(v)`: Removes vertex v and all its incident edges from the graph.

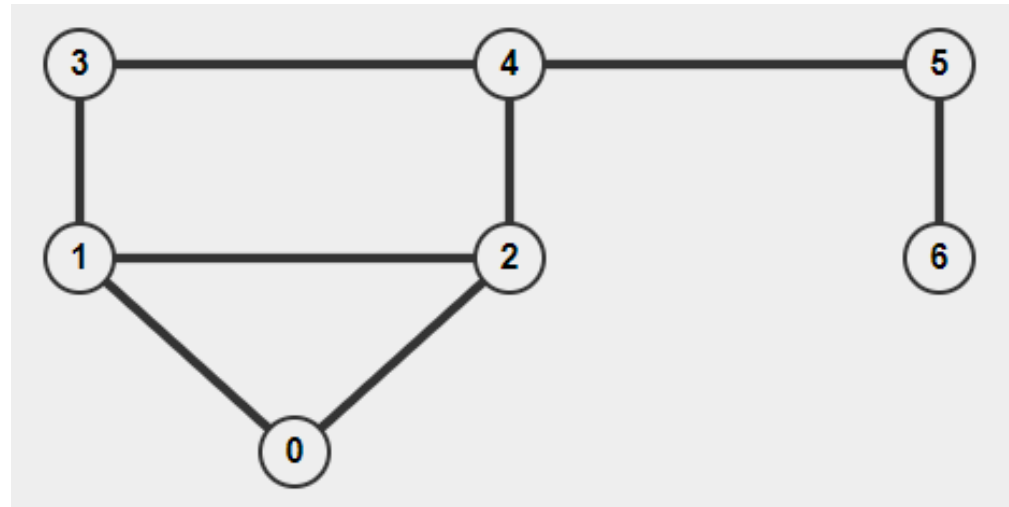
`removeEdge(e)`: Removes edge e from the graph.

[M.Goodrich, sec. 14.1.1, p. 618]

Data Structures for Graph

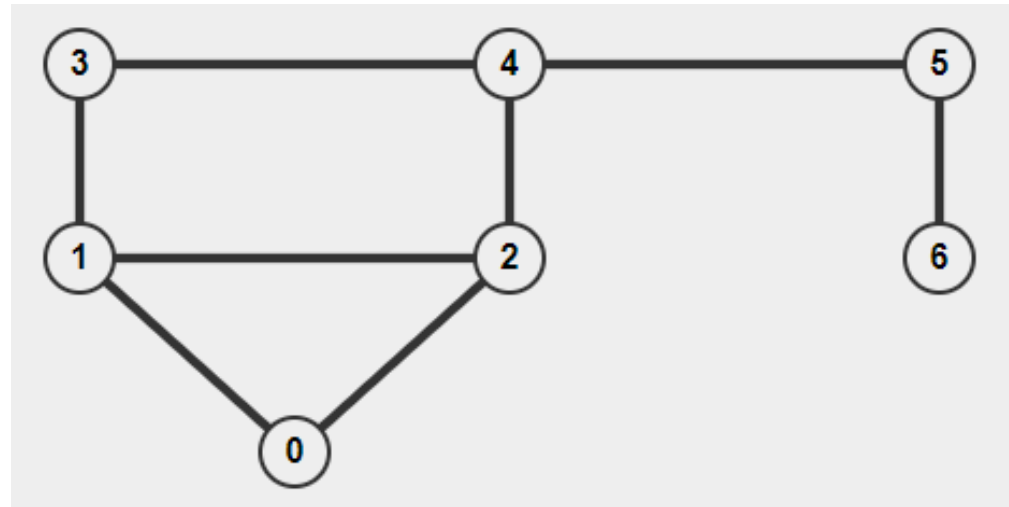
- Adjacency Matrix
 - Adjacency List
 - Edge List
-

Adjacency Matrix



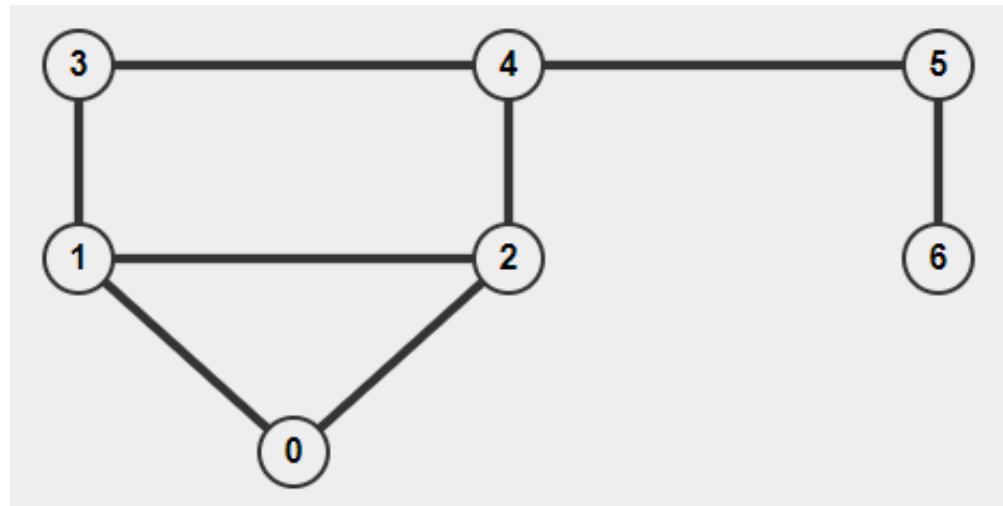
Adjacency Matrix							
	0	1	2	3	4	5	6
0	0	1	1	0	0	0	0
1	1	0	1	1	0	0	0
2	1	1	0	0	1	0	0
3	0	1	0	0	1	0	0
4	0	0	1	1	0	1	0
5	0	0	0	0	1	0	1
6	0	0	0	0	0	1	0

Adjacency List



Adjacency List			
0:	1	2	
1:	0	2	3
2:	0	1	4
3:	1	4	
4:	2	3	5
5:	4	6	
6:	5		

Edge List



Edge List		
0:	0	1
1:	0	2
2:	1	2
3:	1	3
4:	2	4
5:	3	4
6:	4	5
7:	5	6



Graph Algorithms and Applications



Algorithms

The screenshot shows the Graph Online website interface. The browser's address bar displays graphonline.ru/en/. The website's navigation bar includes links for Home, Create Graph, Help, News, Contacts, and Language. The main heading is "Find shortest path", followed by the instruction "Create graph and find the shortest path. On the Help page you will find tutorial video." Below this, there is a toolbar with buttons for Graph, View, Default, Add vertex, Connect vertices, Algorithms, Remove object, and Settings. The "Algorithms" button is highlighted, and its dropdown menu is open, listing various graph algorithms. To the right of the dropdown, there is a "Vertex enumeration" section with a dropdown menu showing "1, 2, 3...".

<https://graphonline.ru/en/>

- Find shortest path using Dijkstra's algorithm
- Breadth-first search
- Depth-first search
- Search of minimum spanning tree
- Graph coloring
- Arrange the graph
- Find Maximum flow
- Find Eulerian cycle
- Find Eulerian path
- Find Hamiltonian cycle
- Find Hamiltonian path
- Find connected components
- Floyd-Warshall algorithm
- Visualisation based on weight
- Search graph radius and diameter
- Calculate vertexes degree

Algorithms and pplications

<https://www.sanfoundry.com/java-programming-examples-graph-problems-algorithms/>

<https://www.geeksforgeeks.org/applications-of-graph-data-structure/>

Summary

