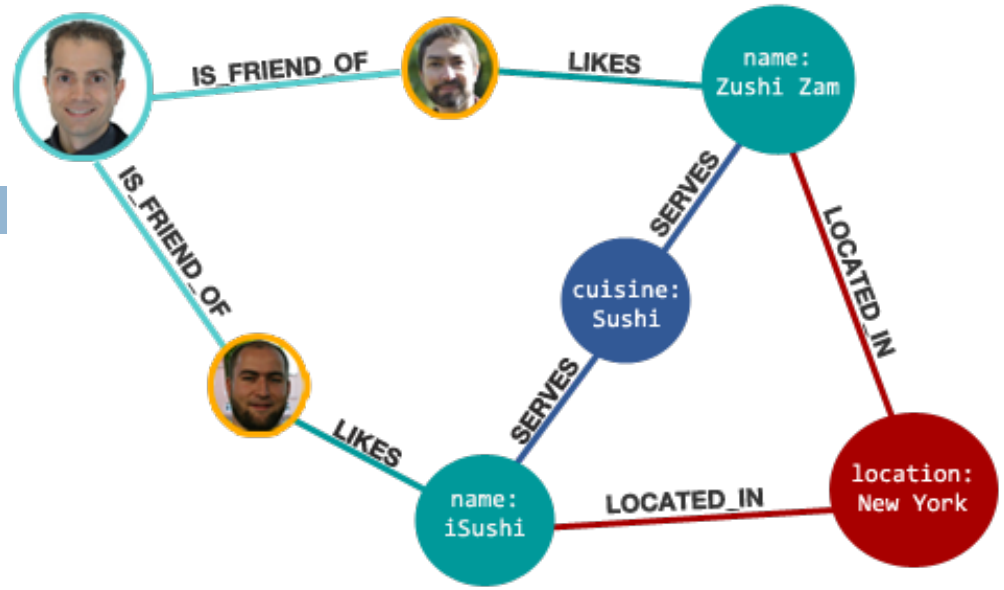
SIT221: DATA STRUCTURES & ALGORITHMS

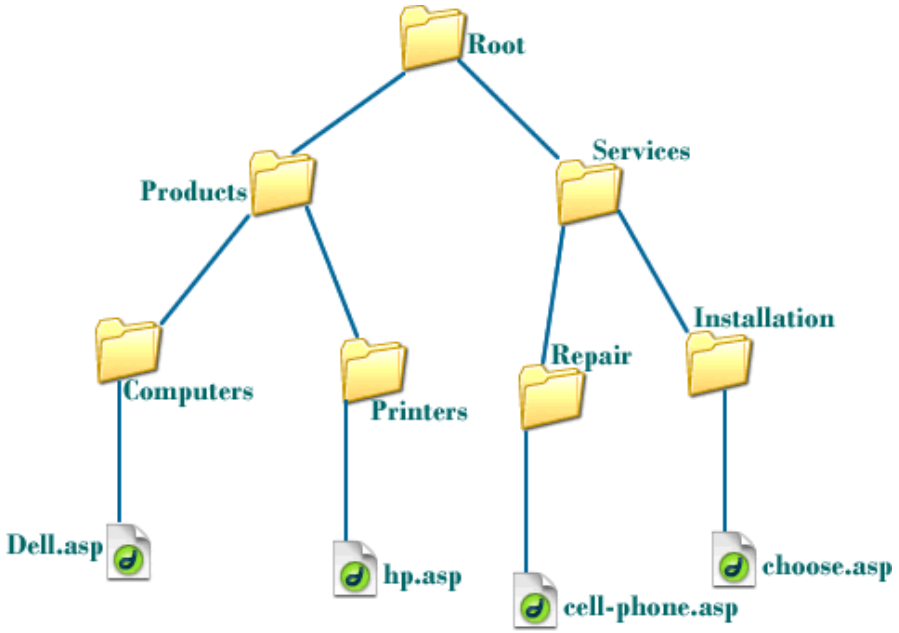
LECTURE #7: GRAPHS

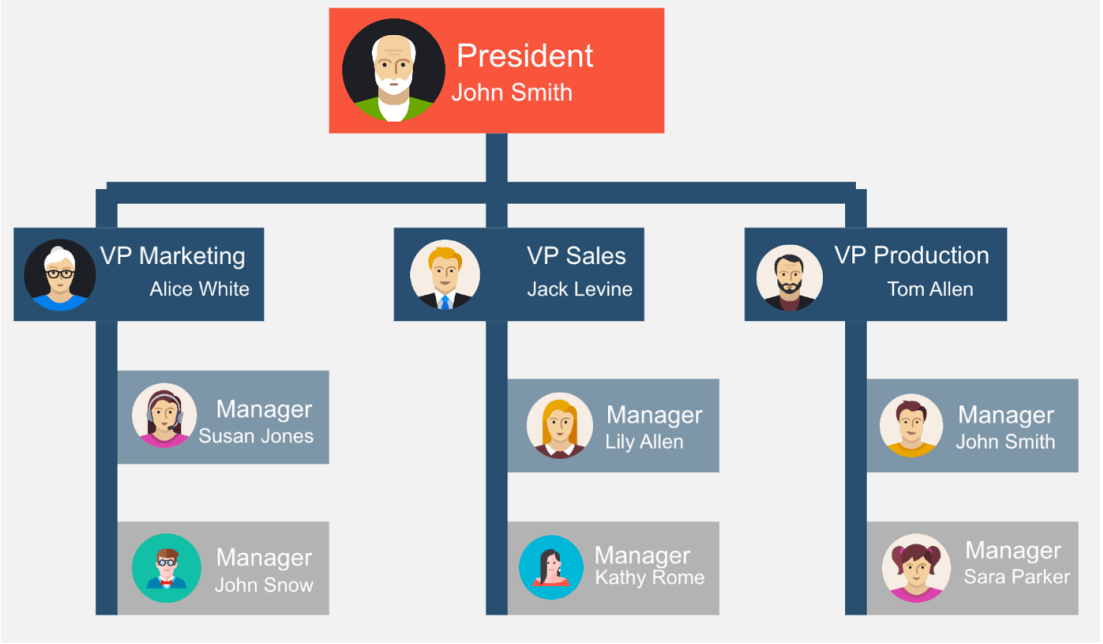
29th August, 2016

Non-linear data structures

- Values are not arranged in order, many next/previous nodes.
  - Hierarchical data (this week)
  - Network data (next week)

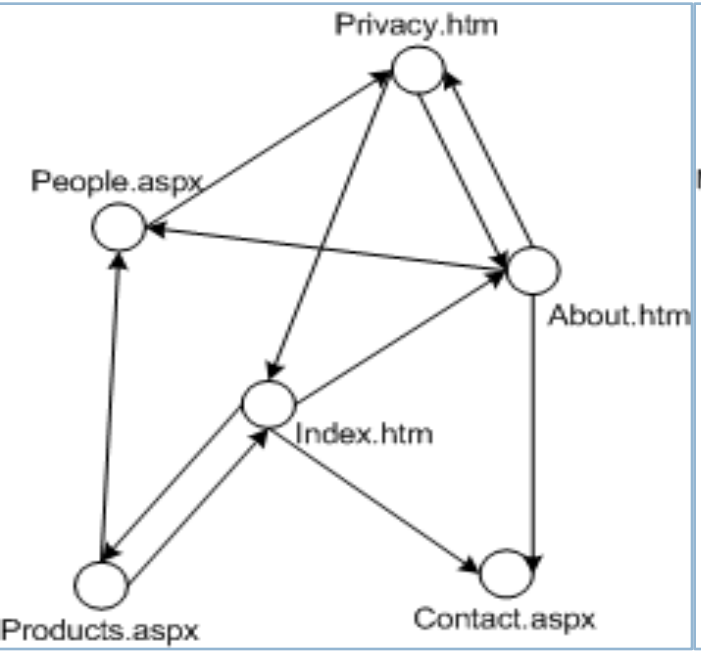


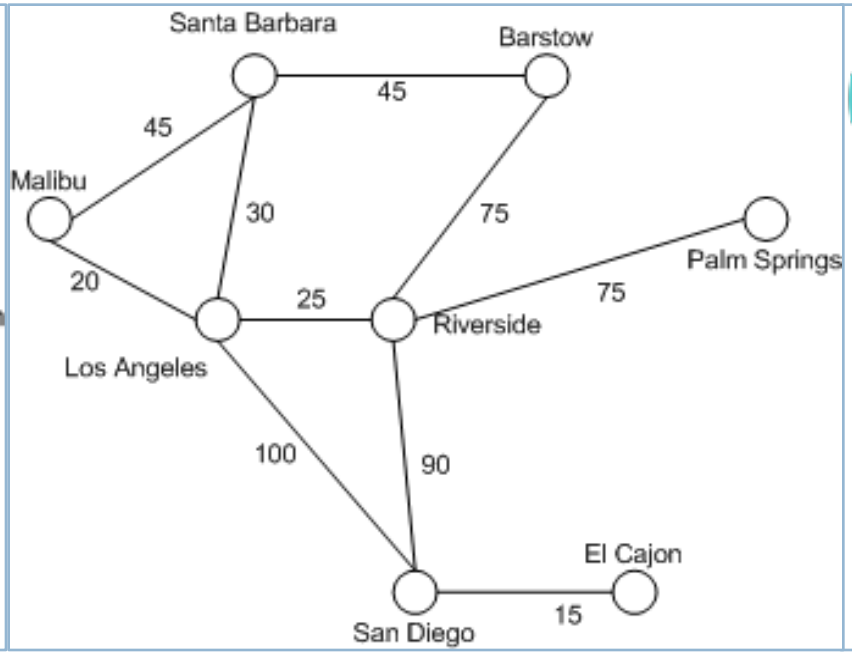


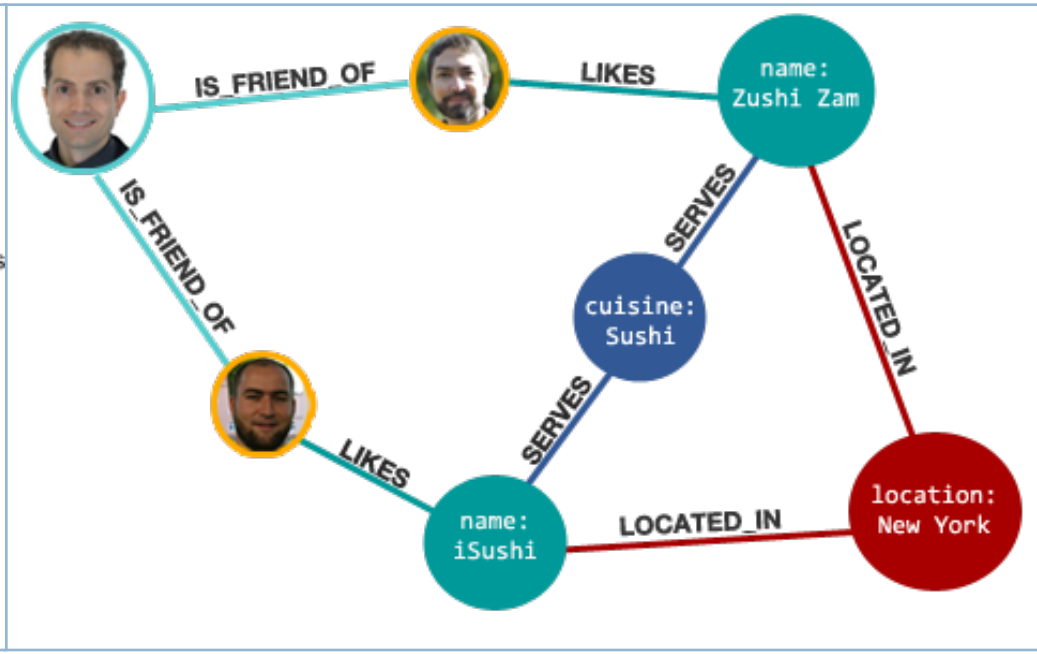


Motivation

- How can we store data of this type in memory?



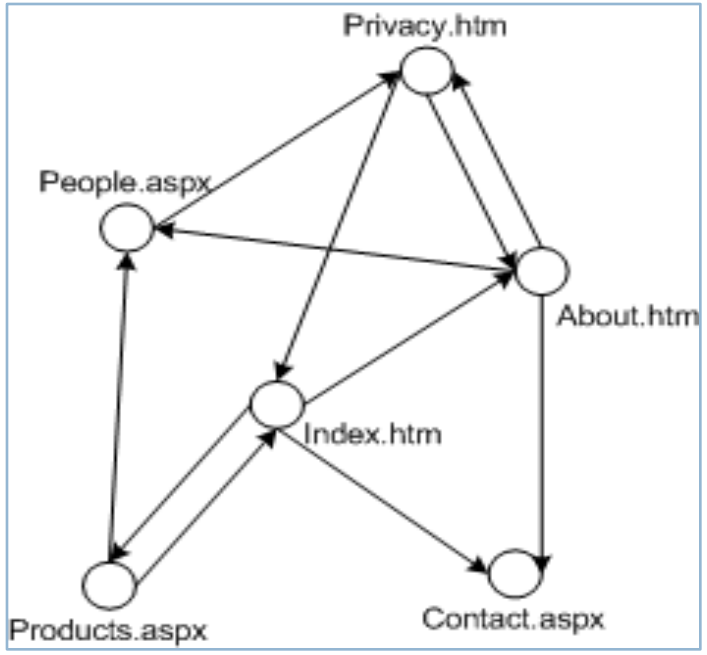


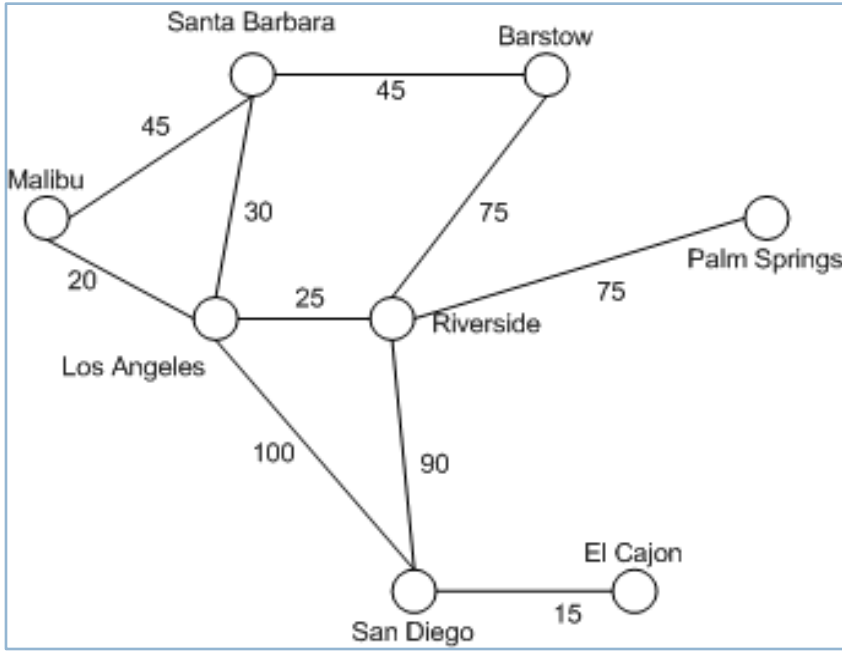


Graph data (1)

- What are the key elements in this diagram?
  - Circles [Pages / Nodes / Individuals / …] ==> we call these circles as vertices
  - Links [Navigation from page to another, route from a city to another, …] ==> we call these links as edges

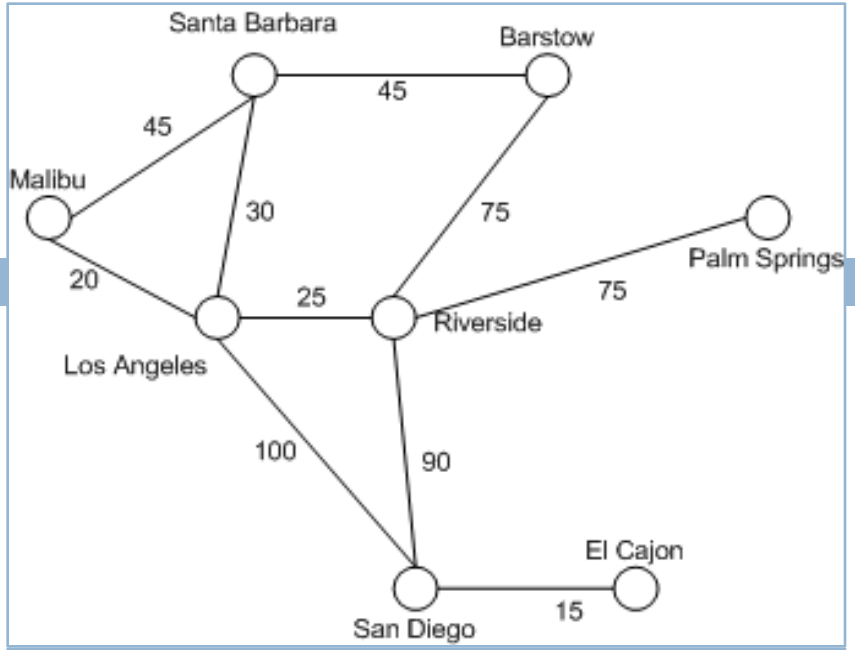
- Thus, any graph can be described as: Vertices & edges
- **G=(V,E)**

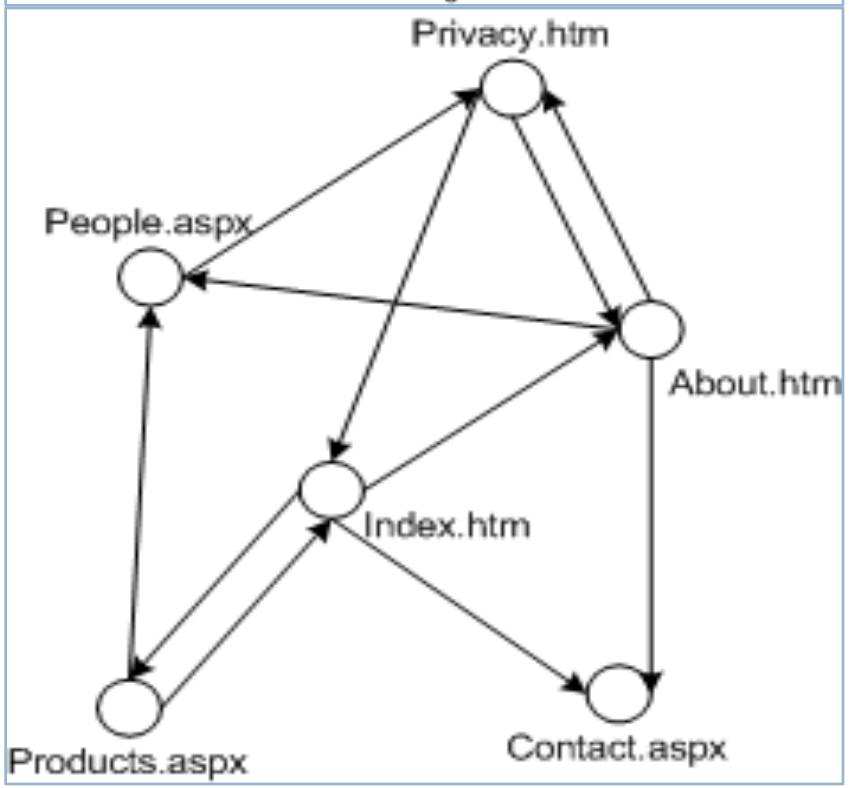




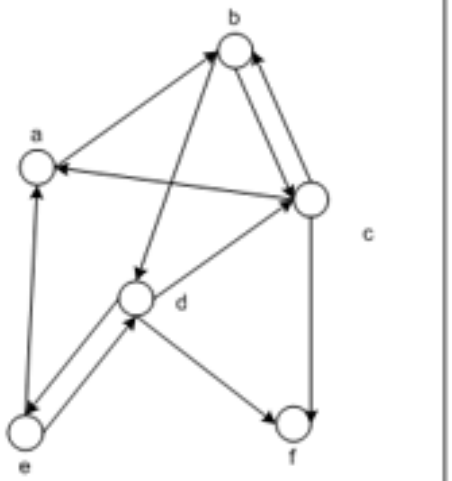
Graph data (2)

- Are all graphs the same? No!
- Some graphs are directed (edges have direction on the arrow from - to)
- Others are undirected (edges do not have directions).
- Some edges might have weights / distance / travel time / etc.
- Some graphs are sparse (few links/edges), while others are dense (too many edges).

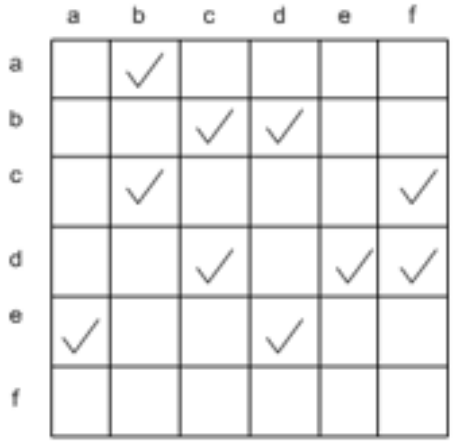




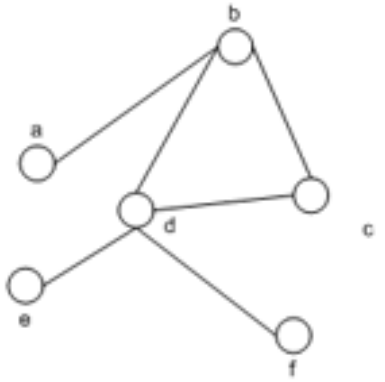
How to represent a graph?



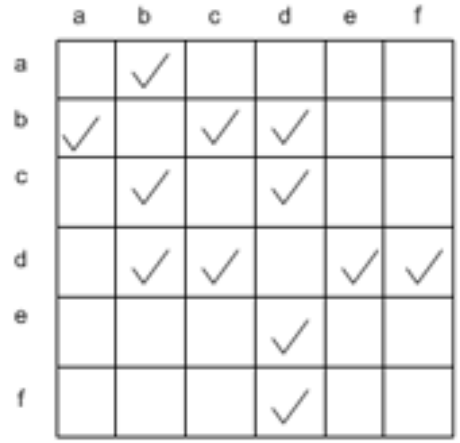
Directed Graph (a)



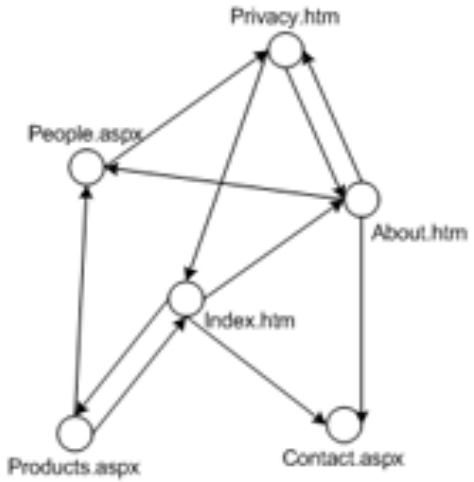
Adjacency Matrix Representation (a)



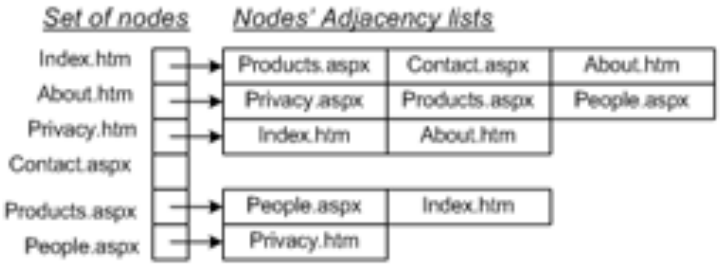
Undirected Graph (b)



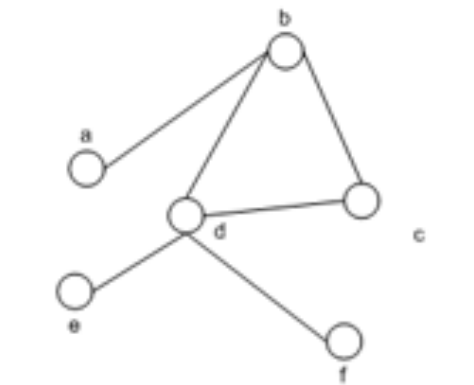
Adjacency Matrix Representation (b)



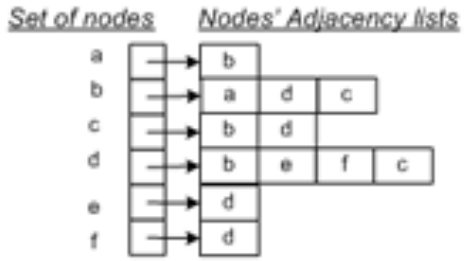
Directed Graph (a)



Adjacency List Representation (a)



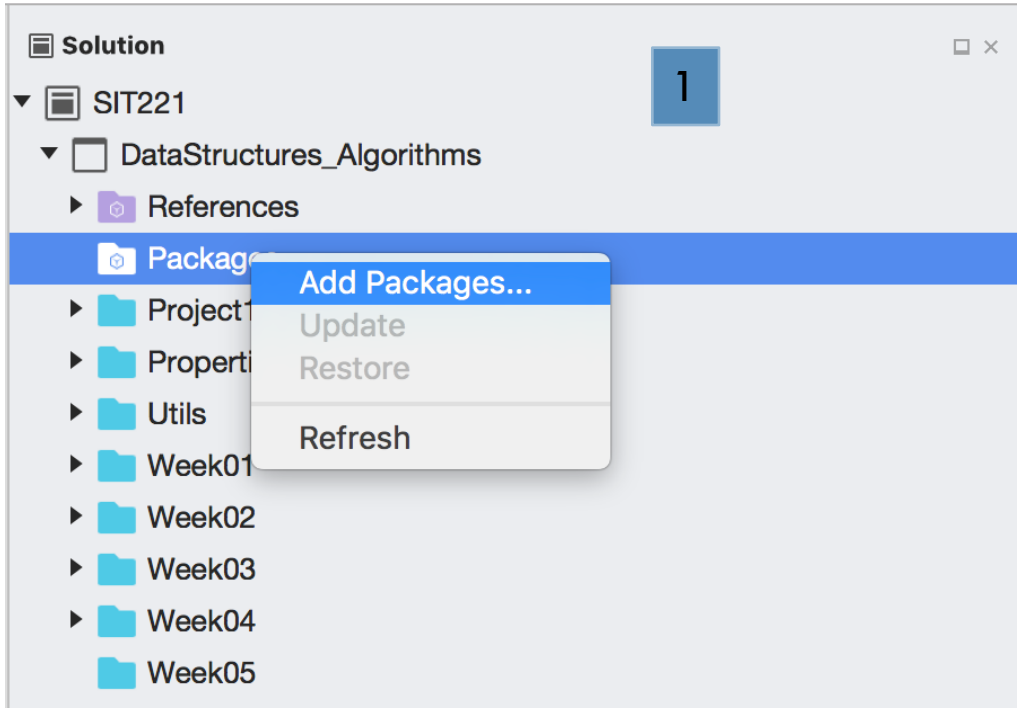
Undirected Graph (b)

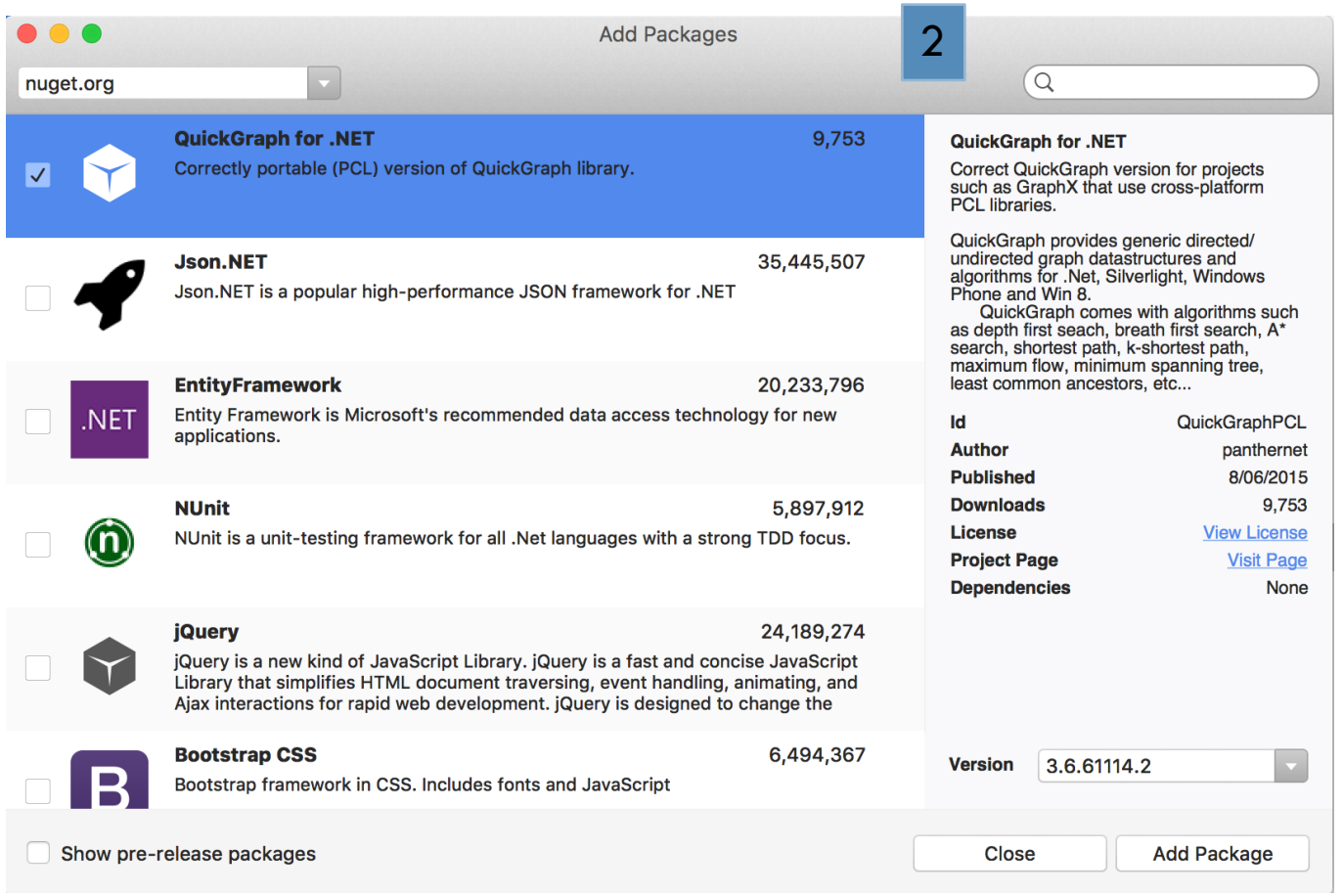


Adjacency List Representation (b)

How to represent a graph? – Our way!

- We will use a 3rd-part (developed by someone else) library/package.
- How can we start using it?

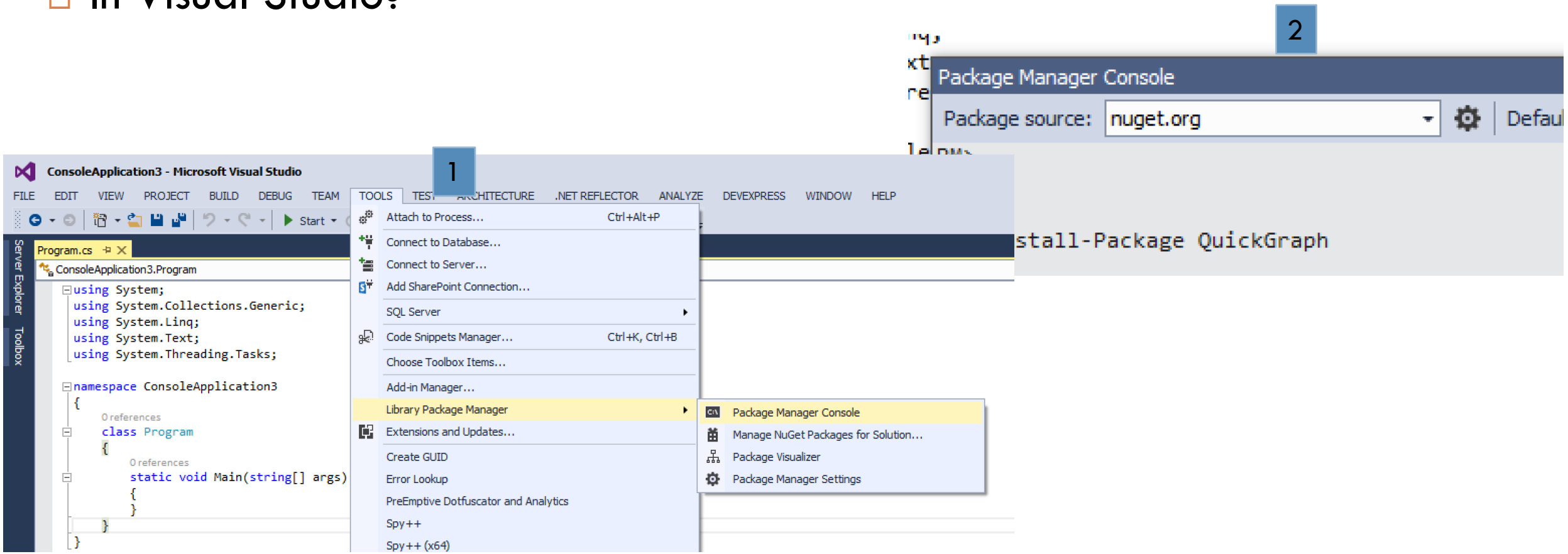




https://quickgraph.codeplex.com

How to represent a graph? – Our way!

- In Visual Studio?



https://quickgraph.codeplex.com

1. Declaring a graph of cities – Example1


```
// Let's create a bi-direction graph
// What do you need to know? Node type and edge type
//Var graph = new BidirectionalGraph<NODETYPE, EDGE<NODETYPE>>();

// In this example, we define a bidirectional graph of city name and
// distances between them - thus nodes of type string and edges of type double

var trafficNetwork = new BidirectionalGraph<string, Edge<string>>();
```

1. Declaring a graph of integers – Example2


```
BidirectionalGraph<int, Edge<int>> IntegersGraph =
        new BidirectionalGraph<int , Edge<int>>();
```

1. Declaring a graph of POIs – Example 3

// I have create a new GraphEdge class that enables you to add description and distance/time to any edge

```
BidirectionalGraph<PointOfInterest, GraphEdge<PointOfInterest>> POIsGraph
= new BidirectionalGraph<PointOfInterest , GraphEdge<PointOfInterest>>();
```

2. Let's add edges – Cities graph

//graph.AddVertex, you give it an object of the type you used in graph declaration – e.g. string, PointOfInterest, etc.

```
//Let's add Australia big cities
trafficNetwork.AddVertex("Melbourne");
trafficNetwork.AddVertex("Sydney");
trafficNetwork.AddVertex("Brisbane");
trafficNetwork.AddVertex("Adelaide");
trafficNetwork.AddVertex("Perth");
trafficNetwork.AddVertex("Melbourne"); //what would happen here?
```

2. Let's integer vertices – Integers graph


```
//Let's add integer nodes
trafficNetwork.AddVertex(2);
trafficNetwork.AddVertex(4);
trafficNetwork.AddVertex(3);
trafficNetwork.AddVertex(6);
```

2. Let's add POI nodes - POsGraph


```
//Let's add POI nodes
var poi = … // you can get it from any source or create a new POI obj
POIsGraph.AddVertex(poi);
```

3. Let's add edges between cities


```
//new Edge<string>(Source, Target)
trafficNetwork.AddEdge(new Edge<string>("Melbourne", "Sydney"));
trafficNetwork.AddEdge(new Edge<string>("Melbourne", "Brisbane"));
trafficNetwork.AddEdge(new Edge<string>("Sydney", "Brisbane"));
```

3. Let's add edges between POIs?

// I have create a new GraphEdge class that enables you to add description and distance/time to any edge

```
//Let's add Edges between two POIs
POIsGraph.AddEdge(new GraphEdge<PointOfInterest> { Source = OBJ_POI1 , Target =  OBJ_POI2 , Description = "Description of route from POI1 to POI2", Distance = 100 });
```

4. Let's loop/display graph nodes and edges


```
// Let's display all graph nodes
foreach (var vertex in trafficNetwork.Vertices)
     Console.WriteLine(vertex);

 // Let's display all edges
foreach (var edge in trafficNetwork.Edges)
     Console.WriteLine(edge);


// Let's display each node and it's edges
foreach(var vertex in trafficNetwork.Vertices)
   foreach (var edge in trafficNetwork.OutEdges(vertex)) //you can use InEdges
         Console.WriteLine(edge);
```

Graph Applications

- Traffic / Navigation
- PageRanking – Search Engines
- Social media – Facebook / Twitter
- Computer networks / routing
- Dependencies – in your code / unit pre-requisites
- Project management – critical path analysis

Next week

- Trees and Graphs
  - Traversal & Searching
    - Breadth first vs Depth first
  - Shortest Path
    - Dijkstra
  - Minimum spanning tree