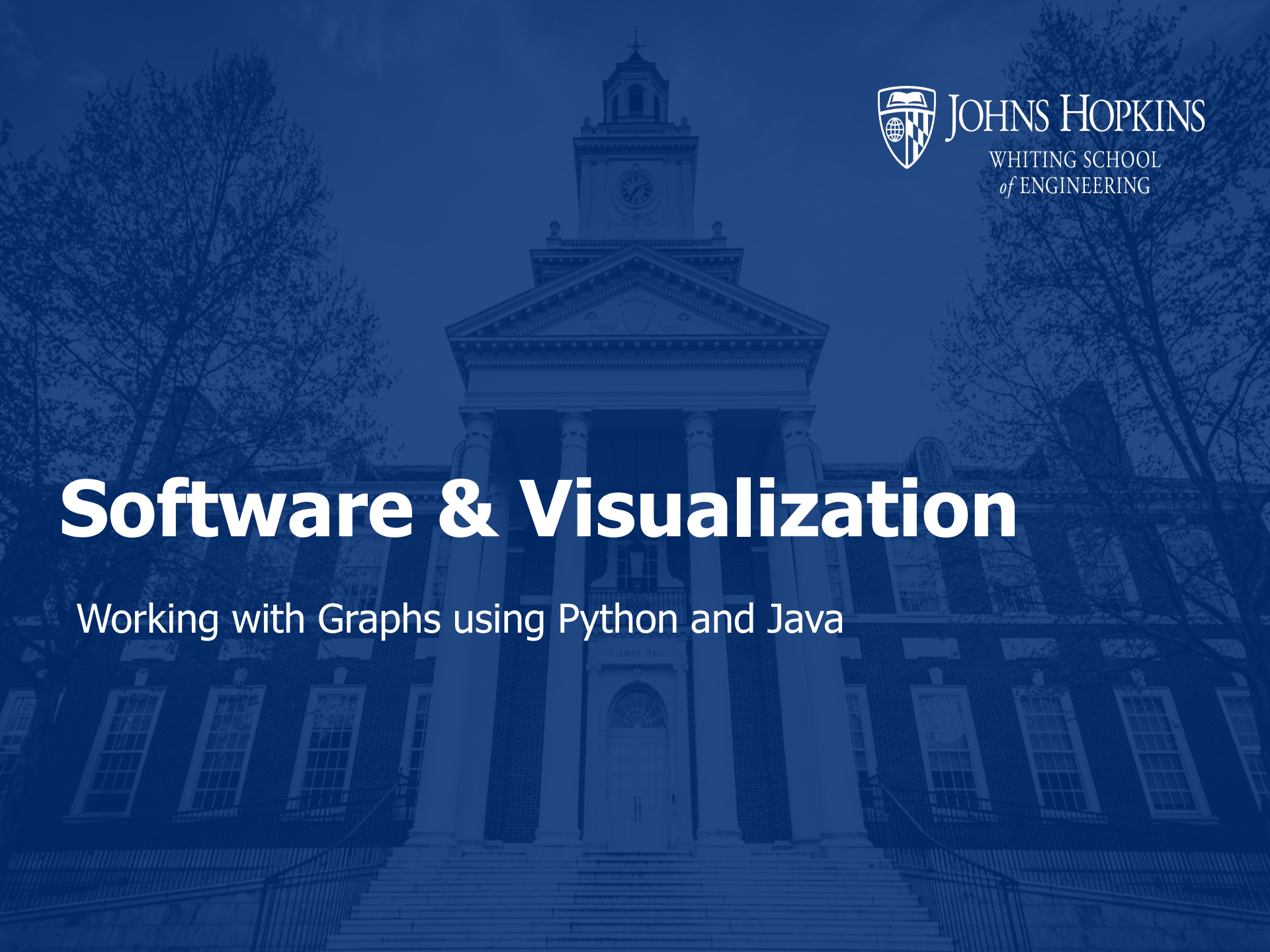


Software & Visualization

Working with Graphs using Python and Java

Goal

Goal of these slides is to give you a brief orientation to some useful software that you can use to work with graphs.

This is optional material and if you already have libraries or tools you prefer to use feel free to use those.

Those interested in large scale property graph processing may find Tinkerpop discussion useful as it facilitates handling such graphs.

Example code package

- You will not be tested on any language specific knowledge in this class. Our focus is on graph algorithms and analysis.
- To help you get started with working with graphs class website provides some code examples using Java and Python.
- If you prefer another language or software library, feel free to use that.
- All graphs in the class will be shared either in graphml or graphson formats. So at a minimum you should be able to open and work with a graph file in one of these formats. Provided Java and Python examples show how to do that.

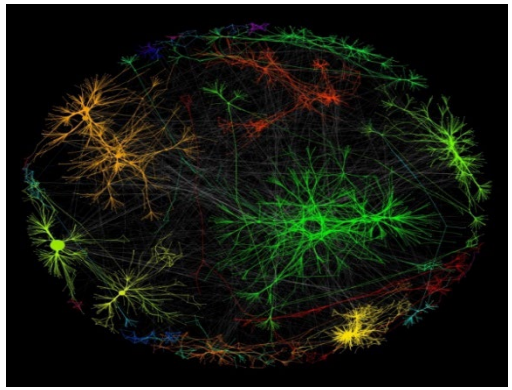
Visualization

Some Libraries for Graph Visualization

Below are two of the freely available tools for graph visualization that you may find useful:

Cytoscape: http://www.cytoscape.org/

Gephi: https://gephi.org/



For many real world graphs no tool can visualize the entire graph. Often you will want to "filter" what you want to look at based on some calculated properties. This is related to "graph query" topic which we touch on in Tinkerpop discussion.

Python

Using Python to create and save a Graph

- Those who would like to use Python, easiest way to get Python is to download Anaconda distribution which comes with useful libraries:

  https://www.anaconda.com/distribution/

- NetworkX library provides a simple interface and tools for working with graphs.

- You may also find the online NetworkX tutorial useful

Example:

Create a simple graph, save, and read it

```
1   import networkx as nx
2
3   # Create an empty graph:
4
5   G = nx.Graph()
6
7   # Add some nodes with properties:
8
9   n1 = G.add_node(1, name="john")
10  n2 = G.add_node(2, name="mary")
11  n3 = G.add_node(3, name="mike")
12  n4 = G.add_node(4, name="lisa")
13
14  # Add some edges:
15
16  e1 = G.add_edge(1,2, w="2")
17  e2 = G.add_edge(1,3, w="3")
18  e3 = G.add_edge(2,3, w="2")
19  e4 = G.add_edge(3,4, w="2")
20
21  G.nodes.data()
22
23  G.edges.data()
24
25  # Save network:
26
27  nx.write_graphml_lxml(G, "C:/workspace/graphs/data/simple/s
28
29  # Read it back:
30
31  G2 = nx.read_graphml("C:/workspace/graphs/data/simple/simpl
```

Using Python to get and visualize Graph Matrices

```python
import numpy as np
from scipy import linalg
import matplotlib.pyplot as plt
import networkx as nx
```

Example:

Generate 2 built in random graphs, get and visualize their adjacency matrices

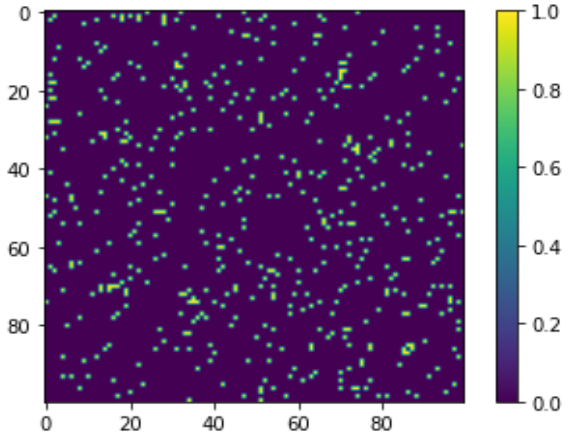
```python
### Generate an Erdos Renyi graph:

er = nx.generators.random_graphs.fast_gnp_random_graph(100, 0.05, seed=1, directed=False)

A = nx.adjacency_matrix(er)

Ad = A.todense()

plt.imshow(Ad)
plt.colorbar()
plt.savefig("C:/workspace/graphs/python/er.png", format="png")
plt.show()
```



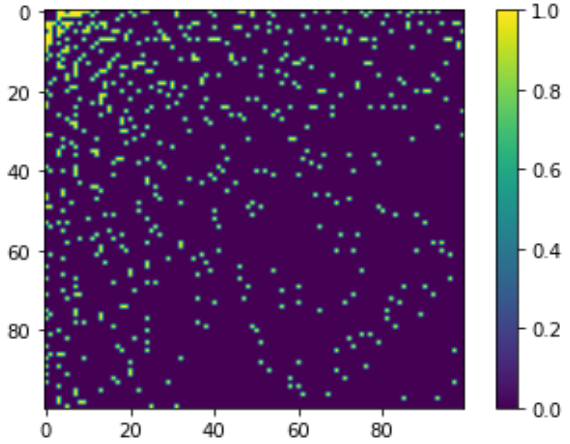

```python
### Generate a Barabasi Albert graph:

ba = nx.generators.random_graphs.barabasi_albert_graph(100,3,1)

A = nx.adjacency_matrix(ba)

Ad = A.todense()

plt.imshow(Ad)
plt.colorbar()
plt.savefig("C:/workspace/graphs/python/ba.png", format="png")
plt.show()
```



Java

Using Java Tinkerpop Graph API

Advantages:

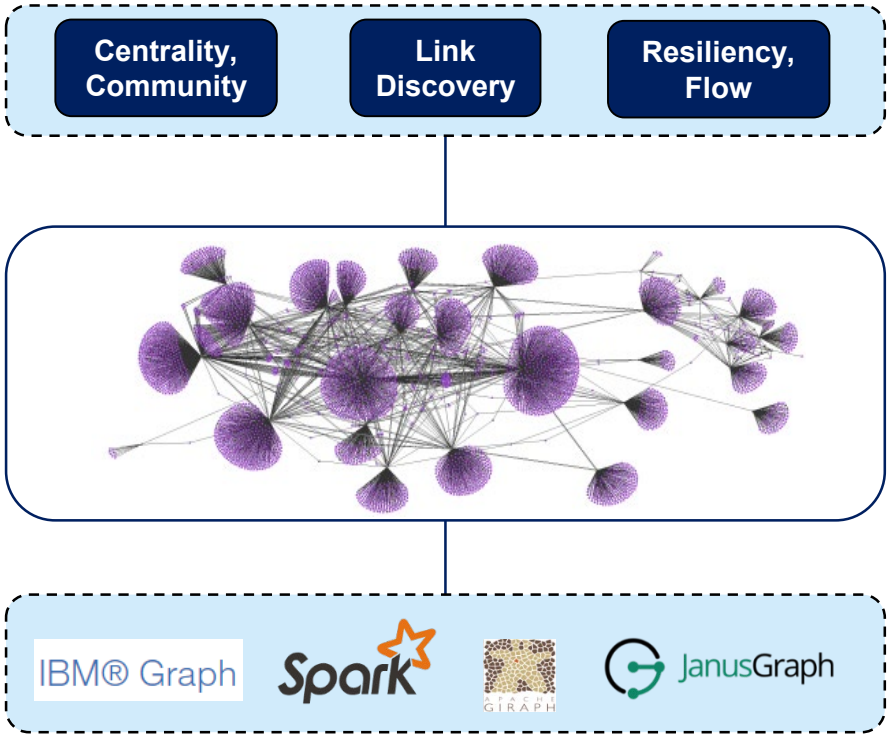
- A standard interface that unifies access to data
- Suitable for large scale processing
- Decouples analytics from specifics of back-end.
- Makes code portable to other platforms.

Disadvantage:

- Potential performance impact compared to a native implementation

Tinkerpop Graph API :

http://tinkerpop.apache.org/



An Evolving List of Back-Ends:

Using Java Tinkerpop to Create a Graph

Create a graph[*]:

```
Graph graph = TinkerGraph.open();
```

Add 2 vertices with properties:

```
Vertex john = graph.addVertex("name", "john", "age", 41);
Vertex sarah = graph.addVertex("name", "sarah", "age", 47);
```

Add an edge with properties:

```
Edge edge = john.addEdge("knows", sarah, "type", "friend");
```

Iterate over vertices & print names:

```
GraphTraversalSource g = graph.traversal();

GraphTraversal<Vertex, Vertex> t = g.V().unfold();

while(t.hasNext())
{
    Vertex v = t.next();
    System.out.println(v.value("name"));
}
```

[*] A variety of code examples are provided in a Java Project under your Module 1 in Blackboard for those interested in using Java. These are just a few snippets.

Simple Graph Query Examples with Java Tinkerpop

Query using multiple equality constraints, save results to a list, and iterate:

```
ArrayList<Vertex> list = new ArrayList<Vertex>();

GraphTraversalSource g = graph.traversal();

g.V().has("sex","1").has("school","0").fill(list);

for(Vertex v:list)
{
        System.out.println(v.value("sex")+ " " + v.value("school"));
}
```

Query using inequality constraint and iterate over results:

```
GraphTraversalSource g = graph.traversal();

GraphTraversal<Vertex, Vertex> t = g.V().has("grade", P.between("7","9")).unfold();

while(t.hasNext())
{
        Vertex v = t.next();
        System.out.println(v.value("grade"));
}
```