Graphs and Trees

Lecturer: Nataliia Stulova
Teaching assistant: Mohammadreza Hazirprasad
Graphs
Graph data structure

A data structure to store a collection of elements (vertices) and relations between them (edges):

- 5 vertices: A, B, C, D, E
- 5 edges: (A,B), (A,E), (A,C), (B,D), (B,C)

Use whenever you need to study a network:

- city map (public transport routes and stops)
- social network (“share with friends of friends”)
- program call graph (which method calls which)
- .... and many more
Graph properties

Edges can have additional properties:
● direction (=> “directed graph”)
● weight (=> “weighted graph”)

Most common tasks:
● find a path between two vertices
● find cycles (paths that begin and end at the same vertex)
Graph data structure implementation

```java
public class Graph {
    List<Edge> vertices;
}

public class Edge {
    public Vertex start;
    public Vertex end;
    public int weight;
}

public class Vertex {
    public String label;
    public int weight;
}
```

Adjacency matrix of a graph

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>5</td>
<td>1</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>B</td>
<td>5</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Java and Graphs

Java doesn't have a default implementation of the graph data structure.

But there are several public libraries to use:

- [JGraphT](#)
- [Google Guava](#)
- Apache Commons [Graph](#)
Tree data structure

A data structure to store a collection of (usually, ordered) elements in a structured way:

- A is a tree root
- B and C are regular nodes, children of A
- D and E, and F are leaf nodes
- ...but we do not have branches: we have sub-trees!
- each tree has depth: the number of levels
- unlike a graph, never has a cycle
Trees and Lists

Single-linked list:
- each node holds a value
- each node has one child

Tree:
- each node holds a value
- each node has several children
Trees and Arrays

Any tree can be converted to an array:

- need to account for number of children of each node
- need to account for tree traversal order:
  - breadth-first (here), or
  - depth-first
Tree properties

Tree elements are typically ordered and it is reflected in the tree structure:

- Binary search tree
- Min/Max Heap

On top of that, many trees are balanced to minimize the difference in length of the branches (for performance reasons):

- AVL tree
- B-tree
- Red–black tree
Trees for search

Most commonly tree data structures are used to store data sorted according to some order and make the search of elements with specific values faster compared to data structures with linear lookup, such as arrays and lists.

**Binary search tree:**
- child on the left has smaller value than parent
- child on the right has larger value than parent

Insertion order:
12, 15, 20, 9, 13, 3

Steps (=cost) to check if 13 is present:
- list: 5 (linear)
- tree: <3 (logarithmic, at most its depth)
More trees use

Other tasks where it is convenient to store data in trees:

- parsing
- autocomplete
- indexing

A parse tree of an arithmetic expression 
(a+b)*c+7

A trie (prefix tree) for keys "A", "to", "tea", "ted", "ten", "i", "in", and "inn"
Java and Trees

Java doesn't have a default implementation of any tree data structures as general purpose collection classes or interfaces, but some data structures are using them internally:

- **TreeMap (Java SE 11 & JDK 11)** - A Red-Black tree based NavigableMap implementation
- **TreeSet (Java SE 11 & JDK 11)** - A NavigableSet implementation based on a TreeMap

Set and map data structures are coming in the next lecture!
Practice
Exercise

Building graph adjacency matrix

- Write the 3 classes implementing the graph data structure (as in slide 7)
- attributes:
  - as provided
- methods:
  - in Vertex and Edge: constructors
  - in Graph: constructor to create an empty graph, and Matrix toMatrix() that returns an adjacency matrix of the graph

I/O

- read a graph from a CSV file where each row contains edges as triplets: a, 5, b
- print the adjacency matrix to System.out

Tests

Reuse the Matrix implementation from the previous class and add the toString() method to it for pretty-printing the matrix