Java Graphs and Trees

Lecturer: Nataliia Stulova
Teaching assistant: Mohammadreza Hazirprasad

Software Composition Group
University of Bern
30 September 2020
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)
- star example: PageRank (in 2 slides)
- .... 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)
Graphs and Google PageRank

PageRank works by counting the number and quality of links to a page to determine a rough estimate of how important the website is. The underlying assumption is that more important websites are likely to receive more links from other websites.
Adjacency matrix of a graph

A  B  C  D  E
A  0  1  0  0  1
B  0  0  0  0  0
C  1  1  0  0  0
D  0  1  0  0  0
E  0  0  0  0  0

A  B  C  D  E
A  0  5  1  0  -1
B  5  0  2  3  0
C  1  2  0  0  0
D  0  3  0  0  0
E  -1  0  0  0  0
Graph data structure implementation

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

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

```java
public class Vertex {
    public String label;
    public int weight;
}
```
Java and Graphs

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

But there are several public libraries to use:

- JGraphT [https://jgrapht.org/](https://jgrapht.org/)
- Apache Commons [https://commons.apache.org/sandbox/commons-graph/](https://commons.apache.org/sandbox/commons-graph/)
Trees
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 use

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)
Trees use

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 trie for keys "A", "to", "tea", "ted", "ten", "i", "in", and "inn"

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