02393 Programming in C++ Module 11 Graphs

Teacher: Alberto Lluch Lafuente

Sebastian Mödersheim (slides author)

May 2, 2016

Lecture Plan

#	Date	Topic	Chapter *
1	1.2	Introduction	1
2	8.2	Basic C++	1
3	15.2	Data Types	2
4	22.2	Libraries and Interfaces	3
5	29.2		
6	7.3	Classes and Objects	4.1, 4.2 and 9.1, 9.2
7	14.3	Templates	4.1, 11.1
		Påskesferie	
8	4.4	Inheritance	14.3, 14.4, 14.5
9	11.4	Recursive Programming	5
10	18.4	Linked Lists	10.5
11	25.4	Trees	13
12	2.5	Graphs	16.1-16.3,16.5
13	9.5	Summary	
	17.5	Exam	

* Recall that the book uses sometimes ad-hoc libraries that are slightly different with respect to the standard libraries (e.g. strings and vectors).

Some nice websites that visualise algorithms.

- http://www.algomation.com
- http://visualgo.net

Summary of previous lecture

- Main topics
  - ★ Checking equality: abstract data type vs concrete structure;
  - ★ Binary Search Trees, Balanced trees, etc.
- Live programming:
  - ★ Class `Tree` of binary search trees:
    - ▶ Insertion and search;
    - ▶ Traversals: in-, pre- and post-order;
    - ▶ Cost analysis: insert/find/delete in $O(\log n)$;
    - ▶ Tree sort: insert all elements in the tree, then in-order;
  - ★ Class `Set` implemented with binary search trees:

Today

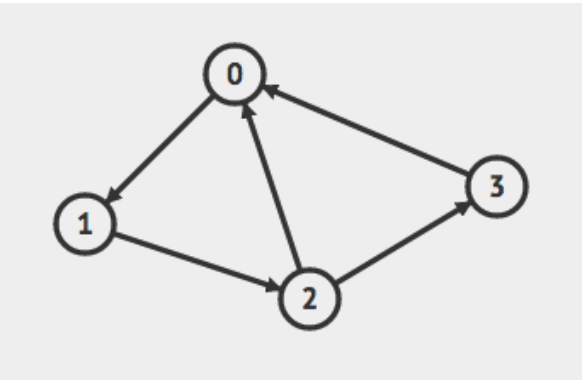
- Main topics
  - ★ Class `Graph`;
  - ★ Different *concrete* representations
    - ▶ Adjacency matrix
    - ▶ Adjacency list/set
  - ★ Operations: insertion, checking reachability, traversals.
  - ★ recursive vs iterative DFS.

What is a graph?

A *directed* graph is a pair $\langle N, E \rangle$, where

- N is a set of nodes;
- $E \subseteq N \times N$ is a set of edges (i.e. pairs of nodes);

Example:
$N = 0, 1, 2, 3, 4$
$E = (0, 1), (1, 2), (2, 0), (2, 3), (3, 0).$

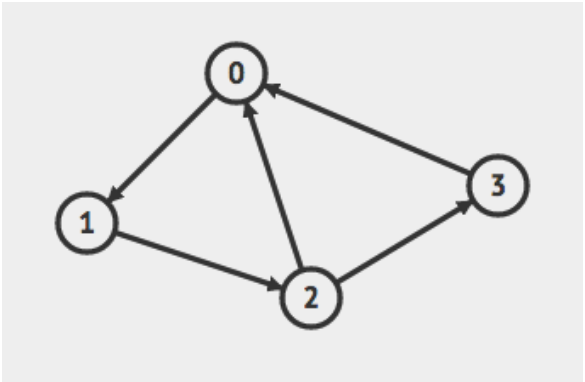


An *undirected* graph is a graph $\langle N, E \rangle$ where we ignore the direction of edges.

There are a lot of classes of other classes of graphs: weighted (costs associated to edges), hypergraphs (edges are n-ary), etc.

Note: graphs are relations.

Main problem under consideration




```
Graph g;

g.insert(0,1); g.insert(1,2); g.insert(2,0);
g.insert(2,3); g.insert(3,0);
...

if (g.reachable(0,3)) ...
```

Is node 3 reachable from node 0?

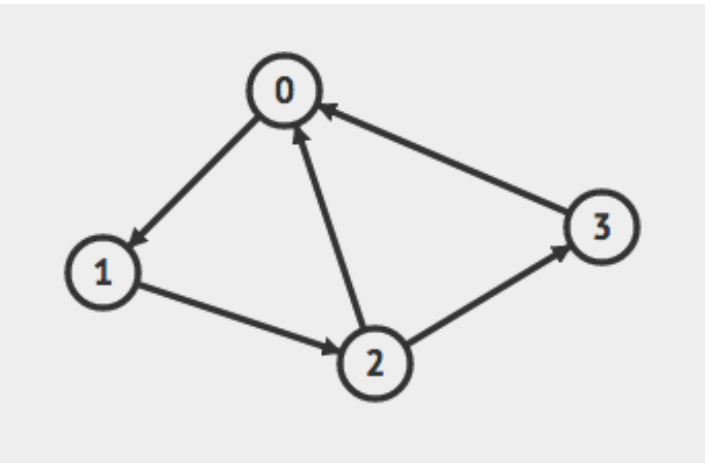
See http://visualgo.net/dfsbfs

Data structures for graphs

Data structures for graphs:

- Adjacency matrix

	0	1	2	3
0	0	1	0	0
1	0	0	1	0
2	1	0	0	1
3	1	0	0	0



  - ★ In general good for *dense* graphs ($|E|$ is $O(|N|^2)$)
- Adjacency list

  0 $\mapsto$ 1
  1 $\mapsto$ 2
  2 $\mapsto$ 0,3
  3 $\mapsto$ 0

  - ★ In general good for *sparse* graphs ($|E|$ is smaller than $O(|N|^2)$, e.g. $O(|N|)$ or $O(|N| \cdot \log |N|)$)

See some examples here: http://visualgo.net/graphds

Class `Graph`: Live Programming