Graphs

Motivation

Use:

- Computation of distances
- Finding of cycles in dependencies
- Detection of connections
- Time management

Concept:

- Generalization of trees
- Special two-values relation

Definition:

A **Graph** consists of:

- Set *N* of vertices (nodes)
- 2-values relation *R: A→N*, whereas A … set of edges (lines)

consequence: Edges are given by pairs of nodes

Items

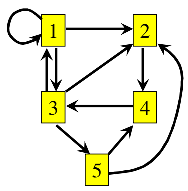
- **path:** $(v_1, v_2, \ldots, v_n)$, $(v_i, v_j) \in A$
- **path length** (=Number of edges): n - 1, n… Number of nodes
- **loops:** $(v, v_1, v_2, \ldots, v_n, v)$
- **loop length:** n + 1
- **simple loop:** each node occurs only once
- **multiple loops:** nodes occur multiple times
- **cyclic graph:** contains at least 1 loop
- **acyclic graph:** contains not one loop
- **acyclic path:** not one node occurs multiple times
- **order** of a graph the number of vertices (nodes)

Directed graph

Edges are oriented: $(u,v) \neq (v,u)$

Notation: $(u,v) \equiv \underset{\substack{tail \\ predecessor}}{u} \rightarrow \underset{\substack{head \\ successor}}{v}$

Example:



N={1,2,3,4,5}

A={(1,1),(1,2),(1,3),(2,4)
(3,1),(3,2),(3,5),(4,3),(5,2),(5,4)}

Directed graph: Definitions

- **Designation of nodes and angles**

Dog $\xrightarrow{\text{bites}}$ Cat

Nodes are **unique** serially numbered, but they can have the same designation

Vertex (node)

- **Output degree:** Number of edges **outgoing** from the vertex
- **Input degree:** Number of edges **incoming** to the vertex
- **Degree:** Sum input degree + output degree

Graph

- **Degree of a graph:** $\max_{i \in N}(Grad_i)$

Undirected graph

Definition:

If $(v_i, v_j) \in A$, then $(v_j, v_i) \in A$ as well.
u and *v* are ***adjacent***,
respectively *u* and *v* are ***neighbor-vertices***.

Implementation of graphs

- **Standard list**

  (number of vertices, number of edges, starting point, ending point of every edge)

  Example: $(\underbrace{5}_{n_n}, \underbrace{10}_{n_a}, \underbrace{1, 1}_{a_1}, \underbrace{1, 2}_{a_2}, \underbrace{1, 3}_{a_3}, \underbrace{2, 4}_{a_4}, \underbrace{3, 1}_{a_5}, \underbrace{3, 2}_{a_6}, \underbrace{3, 5}_{a_7}, \underbrace{4, 3}_{a_8}, \underbrace{5, 2}_{a_9}, \underbrace{5, 4}_{a_{10}})$

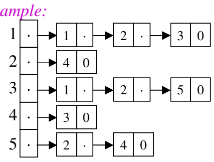
- **Edge-oriented list**

  (number of vertices, number of edges, for every vertex: output degree, targets)

  Example: $(\underbrace{5}_{n_n}, \underbrace{10}_{n_a}, \underbrace{3, 1, 2, 3}_{k_1}, \underbrace{1, 4}_{k_2}, \underbrace{3, 1, 2, 5}_{k_3}, \underbrace{1, 3}_{k_4}, \underbrace{2, 2, 4}_{k_5})$

- **Adjacency list**

  Example:



- **Adjacency matrix**

  Example:

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

Estimation of memory space

- Adjacency list

$(n_n + 2n_a)$ words, n_n … number of vertices
n_a … number of edges

- Adjacency matrix

$n_n^2 / 32$, @ word length = 32 bit

Estimation

$n_n << 2n_a \Rightarrow n_n + 2n_a \approx 2n_a$;
$2n_a < n_n^2 / 32 \Rightarrow$ **$a < n^2 / 64$ … Liste besser**

Graph operations

- **Searching an edge (verify Existence of an edge):**

Matrix: $O(1)$ access 1 element via his indexes
List: $O(1) + O(n_a/n_n)$, worst case $n_a = n_n^2$
Searching in a vector +
average search time in the list

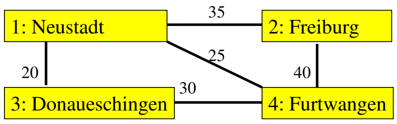
- **Find all successors of a given vertex:**

Matrix: $O(n_n)$ passing the rows
List: $O(1) + O(n_a/n_n)$, worst case $n_a = n_n^2$
the same as at “searching an edge”

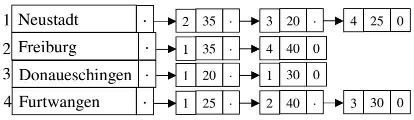
- **Find all predecessors of a given vertex:**

Matrix: $O(n_n)$ passing the columns
List: $O(n_a)$ passing all edges

Marked graphs



Adjacency list



Adjacency matrix

$$\begin{pmatrix} -1 & 35 & 20 & 25 \\ 35 & -1 & -1 & 40 \\ 20 & -1 & -1 & 30 \\ 25 & 40 & 30 & -1 \end{pmatrix}$$

Interrelated components #1

Definition:

Interrelated component: {Vertices K_i | K_i reachable from K_j}

Interrelated graph: has at least 1 interrelated component

Analysis of interrelated components:

G_0 (Stand alone vertex) $\rightarrow$ G_1 (1 connection considered) $\rightarrow$ G_2 $\rightarrow$ ... $\rightarrow$ G_a (all connection evaluated)

Interrelated components #2

Derivation:

Basis:

G_0 consists of vertices from G only, without any edges

Each vertex presents a single component

Induction assumption:

G_i … Graph to exposition of interrelated components, after evaluation of the first *i* edges. Contemplating now (*u*,*v*), the *i*+1 edge in G:

a) $u, v \in$ of the same component of $G_i \Rightarrow$
G_i and G_{i+1} contain the same amount of interrelated components
(the new *i*+1 edge connects not any edges, which are not already connected)

b) u, v $\in$ of different components of $G_i \Rightarrow$
the component including *u* and the component including *v* will be merged

Proof:

Interrelated components #3

Graphical construction:

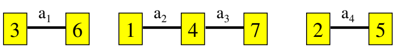
- Beginning: stand-alone vertices as auxiliary graph G_0
- On inspection of every new edge a_i a new auxiliary graph G_i will be drawn
- Interrelated components in G_i will be constructed as trees
- Tree order = max. output degree of the vertex

To which interrelated component affiliates the vertex K_i ?:
- Find K_i in G_i , the root of G_i = interrelated component

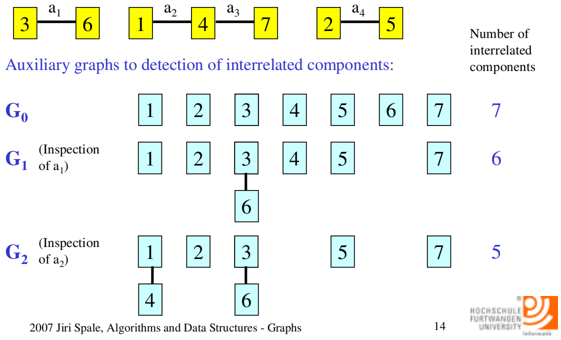
Merging of 2 interrelated components:
- The **root of one component** comes to be **child of the root of another component**

Example

Graph G consisting of 3 interrelated components:



Auxiliary graphs to detection of interrelated components:



Example (continuation)