notes for advanced graph theory lecture on april 16th

Machine model: unit-cost RAM

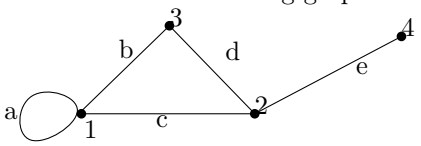
This means:
memory access in constant time
basic arithmetic operations in constant time
words are of unbounded size (this allows to cheat by hiding several operations in arithmetic operations; convention: "cheating not allowed")
the input size is given in numbers of objects (that means out input size is $\Theta(n+m)$, where n is the number of nodes and m the number of edges)

Graph definitions

An edge v, v that connects a node to itself is called a self-loop.
Multiple edges connecting the same two nodes are called parallel edges.
Neither of these is allowed in a simple graph. If either of these occur we speak of a multigraph.
$G = (V, E)$, $n = |V|$, $m = |E|$, $V = V(G)$, $E = E(G)$

Representations of graphs

We will use the following graph as running example:



Edge list

This contains all edges in one list, e.g. $\{1,1\}, \{1,3\}, \{1,2\}, \{2,3\}, \{3,4\}$
size $O(m)$, but impractical to use

Adjacency matrix

	1	2	3	4
1	1	1	1	0
2		0	1	0
3			0	1
4				0

size $O(n^2)$

Incidence matrix

	a	b	c	d	e
1	2	1	1	0	0
2	0	1	0	1	0
3	0	0	1	1	1
4	0	0	0	0	1

size $O(n^2)$

Observation

$\sum_{\text{column sums}} = 2m = \sum_{\text{row sums}}$
Since row sums are degrees, this yields $2m = sum_{v \in V} deg(v)$
This yields

lemma 1. *(Handshake lemma)*
In any graph, the number of odd vertices (i.e. vertices with odd degree) is even.

Adjacency lists

For each node we save a (doubly-linked) list of the nodes that it is adjacent to.

1 $1 \leftrightarrow 1 \leftrightarrow 2 \leftrightarrow 3$
2 $1 \leftrightarrow 3$
3 $1 \leftrightarrow 2 \leftrightarrow 4$
4 3

size: $O(n+m)$

Incidence lists

For each node we save a (doubly-linked) list of the edges it is incident to. Each node stores a link to its incidence list, the nodes are also saved in a double-linked list. Every edge e stores pointers to its endpoints and to all locations of e in the incidence lists.

1 $a \leftrightarrow a \leftrightarrow b \leftrightarrow c$
2 $b \leftrightarrow d$
3 $c \leftrightarrow d \leftrightarrow e$
4 e

size: $O(n+m)$ some basic operations and their costs:

neighbour query(v,w): are v,w neighbours?	$O(\min(\deg(v), \deg(w)))$
find incident edge(v): give an arbitrary edge incident to v	$O(1)$
add edge / delete edge	$O(1)$
add isolated vertex	$O(1)$
delete vertex v (not nec. isolated)	$O(\deg(v))$

Bipartiteness

definition 1. *A graph G is bipartite if V can be partitioned into two sets A and B such that every edge has one endpoint in A and one endpoint in B.*

definition 2. *A graph is k-colorable if its vertices can be colored with k colors s.t. every edge has different colors in its end vertices.*

definition 3. *G is connected if there exists a path between any two vertices; otherwise G is disconnected.*

definition 4. *A (connected) component of a graph is a maximal connected subgraph.*

theorem 1. *G is bipartite iff all cycles in G are even (i.e. of even length).*

Proof. A graph is bipartite iff all its components are bipartite, so we assume wlog that G is connected.
$\Rightarrow$
G is bipartite, so $\exists A, B$ s.t. $V = A \dot\cup B$.
Take an arbitrary cycle, wlog its first vertex is in A. Then its next vertex is in B, the one after that in A, .. When the start vertex is reached again we are back in A (as the start vertex is in A) and so have even length.

$\Leftarrow$
Fix an arbitrary vertex $v \in V$. Let $A = \{w \in V | d(v,w) \text{ even}\}$ (d means distance, i.e. the length of a shortest path connecting v and w) and $B = V \setminus A$.
To show: no edges join two vertices from A (or B).
Assume the contrary, an edge $e = (x,y)$ joining two vertices of wlog B. $d(v,x)$ is odd, as is $d(v,y)$. Furthermore $|d(v,x) - d(v,y)| \leq 1$ since x and y are connected. This yields $d(v,x) = d(v,y)$. Let z be the last vertex both paths have in common. With the same argument as above we have $d(z,x) = d(z,y)$. Thus the cycle z, x, y has odd length. □

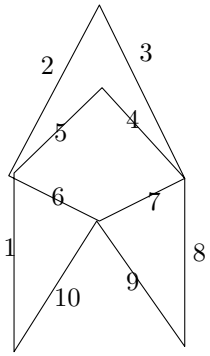


Figure 1: a Eulerian graph with an example tour

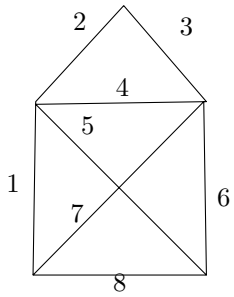


Figure 2: a graph that is not Eulerian but admits a Euler trail

Algorithm for checking bipartiteness

Run DFS to get components and spanning trees of components. Consider the spanning tree T with root r of a component: color r black and $T \setminus r$ accordingly. For all edges (not contained in the spanning tree) check wether its endpoints have distinct colors. If that is the case, we have a 2-coloring (a proof of bipartiteness), otherwise an odd cycle proving that G is not bipartite (compare proof of the above theorem).

Eulerian graphs

definition 5. *A walk in a graph is an alternating sequence of vertices and edges* $v_1 e_1 v_2 e_2 \dots e_{k-1} v_k$ *s.t.* $e_i = \{v_i, v_{i+1}\}$.

definition 6. *A trail is a walk that contains no edge twice.*

definition 7. *A circuit is a closed trail, i.e. a trail where start and end are identical.*

definition 8. *A circuit is Eulerian if it visits every edge.*

definition 9. *A graph is Eulerian if it admits an Eulerian circuit.*

definition 10. *A Eulerian trail is a trail that visits every edge once (it is not necessarily closed).*