Graphs

Reading: §10.1, 10.2 10.3 (opt.), 10.4 (opt.), 10.5 (mostly opt.), 10.5.3, 10.6 (mostly opt.)

Basic Definitions

Graphs are useful for describing relationships between objects in more complex ways than can be achieved by functions and relations.

Defns:

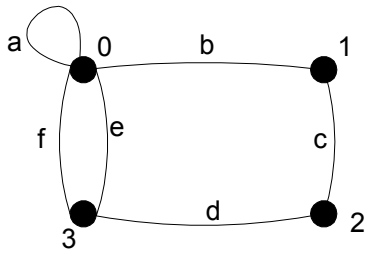
- A **(undirected) graph** $G = (V, E, \phi)$ consists of
  - a set of **vertices** V,
  - a set of **edges** E, and
  - an **incidence function** $\phi : E \to \mathcal{P}(V)$ such that for all $e \in E$, $|\phi(e)| \in \{1, 2\}$
- If $v \in \phi(e)$ we call v an **endpoint** of e.
- So each edge has 1 or 2 endpoints.
- When $|\phi(e)| = 1$ we call e a **loop**.

- Note: Singular of "vertices" is "vertex"
- Vertices are often called **nodes** and edges **arcs**

Example: $G_0 = (V_0, E_0, \phi_0)$ where

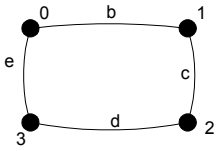
- $V_0 = \{0, 1, 2, 3\}$
- $E_0 = \{a, b, c, e, d, f\}$
- ϕ_0 is defined by

$$\begin{array}{lcl} a & \mapsto & \{0\} \\ b & \mapsto & \{0, 1\} \\ c & \mapsto & \{1, 2\} \\ d & \mapsto & \{2, 3\} \\ e & \mapsto & \{0, 3\} \\ f & \mapsto & \{0, 3\} \end{array}$$



Defn: A graph is called **simple** iff ϕ is one-one (no double edges) and for all $e \in E$, $|\phi(e)| = 2$ (no loops).

Example: G_1 is obtained from G_0 by eliminating edges a and f. G_1 is a simple.



Walks and connectivity

Defns: A **walk** in an undirected graph $G = (V, E, \phi)$ is a sequence of vertices and edges

$$(v_0, e_1, v_1, e_2, ..., e_k, v_k)$$

such that $\phi(e_i) = \{v_{i-1}, v_i\}$ for all $i \in \{1, 2, ..., k\}$.

v_0 and v_k are called the **endpoints** of the walk and the walk is **from** v_0 **to** v_k.

The **length** of the walk is the number of edges, k, and can be 0.

A **trail** is a walk without repeated edges.

A **path** is a walk without repeated vertices.

Example: In G_0 we have many walks, one example is

$$(3, e, 0, a, 0, f, 3, d, 2)$$

This is a walk of length 4 from vertex 3 to vertex 2 and is a trail, but not a path

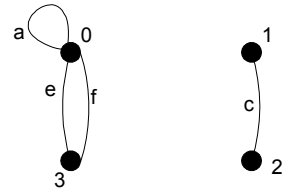
More examples in G_0

	Walk	Trail	Path	Length
(1)	✓	✓	✓	0
$(0, a, 0)$	✓	✓		1
$(0, a, 0, e, 3, f, 0)$	✓	✓		3
$(0, a, 0, e, 3, e, 0)$	✓			3
$(0, f, 1, c, 3)$	×			

A graph G is said to be **connected** if there is a walk from any vertex in G to any vertex in G.

Example: G_0 is connected. The graph G_2 is not.

G_2 :



A **connected region** is a maximal set of vertices such that there is a walk from any vertex in the set to any other. ("Maximal" meaning that adding any other vertex to the region would violate this condition.)

Example: The connected regions of G_2 are $\{0, 3\}$ and $\{1, 2\}$.

Example Applications:

- Consider a power grid. The vertices are the stations (power stations, substations, etc.) The edges are functional transmission lines. The graph is connected if (some) power can be transmitted from any station to any other.
- Face recognition. Consider a digital photograph. The vertices are pixels that meet some specified criterion e.g. colour range is roughly that of human skin. There is an edge between pixels that are distance 1 apart. The connected regions represent potential faces.

Finding connected regions. There are many ways.

Defn: An **adjacency matrix** A for a graph $G = (V, E, \phi)$ with $V = \{v_0, v_1, ...v_{n-1}\}$ is an n by n matrix such that

$$a_{i,j} = \begin{cases} 0 & \text{if there is no edge } e \text{ with } \phi(e) = \{v_i, v_j\} \\ 1 & \text{if there is 1 edge } e \text{ with } \phi(e) = \{v_i, v_j\} \\ 2 & \text{if there are 2 edges } e \text{ with } \phi(e) = \{v_i, v_j\} \\ \dots & \dots \end{cases}$$

$a_{i,j} = |\{e \in E \mid \phi(e) = \{v_i, v_j\}\}|$, for all $0 \leq i, j < n$

Example: For graph G_0 we have

$$A_0 = \begin{bmatrix} 1 & 1 & 0 & 2 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 2 & 0 & 1 & 0 \end{bmatrix}$$

For graph G_2 we have

$$A_2 = \begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 2 & 0 & 0 & 0 \end{bmatrix}$$

Note A is a symmetric matrix.

Note that loops are counted only once. [Contrary to the example in the book.]

Note that if $\deg(v)$ is the number of edges incident with v (counting each loop twice) and if $\mathrm{loop}(v)$ is the number of loops incident with v then the sum of row (or column) i is $\deg(v_i) - \mathrm{loop}(v_i)$.

Note that I (the identity matrix) gives the number of walks of length 0 between pairs of vertices and A gives the number of walks of length 1 between pairs of vertices and $A^2 = AA$ gives the number of walks of length 2 between vertices.

Example: $A_0^2 = \begin{bmatrix} \underline{1} & \underline{1} & \underline{0} & \underline{2} \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 2 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 & \underline{0} & 2 \\ 1 & 0 & \underline{1} & 0 \\ 0 & 1 & \underline{0} & 1 \\ 2 & 0 & \underline{1} & 0 \end{bmatrix} = \begin{bmatrix} 6 & 1 & \boxed{3} & 2 \\ 1 & 2 & 0 & 3 \\ 3 & 0 & 2 & 0 \\ 2 & 3 & 0 & 5 \end{bmatrix}$.

Consider the ensquared entry, representing the number of walks from 0 to 2.

This is the dot product the underlined row and column

$$[1, 1, 0, 2] \cdot [0, 1, 0, 1]$$

These entries represent the following sets of length one walks.

k	0	1	2	3
Walks from 0 to k	$[\{(0, a, 0)\},$	$\{(0, b, 1)\},$	$\emptyset,$	$\{(0, e, 3), (0, f, 3)\}]$
Walks from k to 2	$[\emptyset,$	$\{(1, c, 2)\},$	$\emptyset,$	$\{(3, d, 2)\}]$

Combining these we get the following sets of walks of length 2

k	0	1	2	3
Walks fr 0 to 2 via k :	$[\emptyset,$	$\{(0, b, 1, c, 2)\},$	$\emptyset,$	$\{(0, e, 3, d, 2), (0, f, 3, d, 2)\}]$

Counting all these length two walks, we get 3.

In general:

Theorem: If A is an adjacency matrix for $G = (V, E, \phi)$ with $V = \{v_0, v_1, ...v_{n-1}\}$, $k \in \mathbb{N}$ and $B = A^k$ then $b_{i,j}$ is the number of walks from v_i to v_j with length k.

Proof: By induction on k.

Corollary: v_i and v_j are in the same connected region iff entry i, j in $A^* = \sum_{k=0}^{|V|-1} A^k$ is not 0.

Corollary: G is connected iff A^* has no 0 entries.

Calculating A^*

$B := I$;
$A^* := I$;
int $j := 0$;
// Invariant: $B = A^j$ and $A^* = \sum_{k=0}^{j} A^k$
while($j < n - 1$) {
    $B := AB$;
    $A^* := A^* + B$;
    $j := j + 1$ }

Example:

$$A_0^* = I + A + A^2 + A^3 = \begin{bmatrix} 19 & 11 & 6 & 19 \\ 11 & 4 & 6 & 5 \\ 6 & 6 & 3 & 9 \\ 19 & 5 & 9 & 10 \end{bmatrix}$$

Application:

- Consider a computer network with point-to-point bidirectional links. Can every computer communicate with every other in less than 6 hops?
  - Calculate $\sum_{k=0}^{6} A^k$
- Consider a graph where V is a set of actors and we have an edge with endpoints $\{u, v\}$ if u and v have appeared in the same movie. For a given actor, u, can u be connected to Kevin Bacon in 6 hops or less?

Colouring graphs

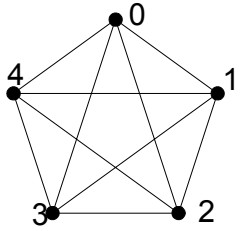
Defn. A k**-colouring** of a graph. $G = (V, E, \phi)$ is any function f from V to a set of size $k \in \mathbb{N}$ such that for all $e \in E$ and $v, u \in V$,

$$\text{if } \phi(e) = \{u, v\} \text{ then } f(u) \neq f(v)$$

Informally, a k-colouring assigns each vertex one of k colours such that the endpoints of no edge are coloured the same.

Example: For G_1 the colouring function $f(0) = f(2) = red$, $f(1) = f(3) = green$ is a 2-colouring.

Example: Consider K_5 which is a graph with 5 vertices where each pair of vertices is connected by an edge.



It has a 5 colouring $f(i) = i$ for each $i \in \{0, 1, ..., 4\}$

A k-colouring of G is optimal if there is no $(k-1)$-colouring of G.

Both the examples above are optimal.

Defn. The **chromatic number,** $\chi(G)$, of a graph, G, is the smallest k such that G has a k-colouring.

Applications of colouring

- Assigning frequencies (or time slots) to stations on a wireless network.
- Grouping traces in PCB testing (see slide set 0).
- Scheduling exams
- Assigning registers to variables when compiling computer programs.
- In general: allocating scarce resources.

Bad news: We know of no algorithm that

- can find an optimal k-colouring for any finite graph, and
- can do so quickly for many large graphs.

Good news:

- A planar graph is one that can be drawn on a sheet of paper such that no two edges cross each other.
- A planar graph can be coloured with only four colours. (Proved 1976 after about 80 years of investigation.)

Directed graphs

Defns: A **directed graph**, $D = (V, E, \phi)$, consists of

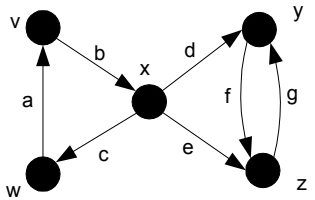
- A set of vertices, V
- A set of edges, E
- A function $\phi : E \to (V \times V)$

If $\phi(e) = (u, v)$ then u is called the **initial vertex** of e, and v is called the **terminal vertex** of e, and if $u = v$ then e is called a **loop**.

D is called **simple** if it has no loops and ϕ is one-one.

Example: D_0 has

- $V_0 = \{v, w, x, y, z\}$
- $E_0 = \{a, b, c, d, e, f, g\}$
- $\phi_0$is defined by $a \mapsto (w, v)$
  $b \mapsto (v, x)$
  $c \mapsto (x, w)$
  $d \mapsto (x, y)$
  $e \mapsto (x, z)$
  $f \mapsto (y, z)$
  $g \mapsto (z.y)$



- D_0 is simple.

Defns:

- A **directed walk** in $D = (V, E, \phi)$ is an alternating sequence of vertices and edges
  $$(v_0, e_1, v_1, ..., v_{i-1}, e_i, v_i, ..., e_k, v_k)$$
  such that $\phi(e_i) = (v_{i-1}, v_i)$ for all $i \in \{1, 2, ..., k\}$.
- The **length** of a directed walk is the number of edges.
- The walk is said to be **from** v_0 **to** v_k .

Example: In D_0 we have walks

$$(v)$$
$$(v, b, x)$$
$$(v, b, x, d, y, f, z, g, y)$$

Defn. An **adjacency matrix**, A, for a directed graph, $D = (V, E, \phi)$, where $V = \{v_0, v_1, ..., v_{n-1}\}$, is defined by

$$a_{i,j} = \begin{cases} 0 & \text{if there is no edge } e \text{ with } \phi(e) = (v_i, v_j) \\ 1 & \text{if there is 1 edge } e \text{ with } \phi(e) = (v_i, v_j) \\ 2 & \text{if there are 2 edges } e \text{ with } \phi(e) = (v_i, v_j) \\ ... & ... \end{cases}$$

I.e.

$$a_{i,j} = |\{e \in E \mid \phi(e) = (v_i, v_j)\}| \text{, for all } 0 \leq i, j < n$$

Example: If we order the vertices of D_0 alphabetically, its adjacency matrix is

$$A_0 = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Theorem: If A is an adjacency matrix for $D = (V, E, \phi)$, the number of directed walks of length k from v_i to v_j is entry i, j in A^k, for all $k \in \mathbb{N}$, $v_i, v_j \in V$.

Proof: Induction on k.

Defn A directed graph, $D = (V, E, \phi)$, is **strongly connected** if there is a walk from every vertex to every other vertex.

Example: C_5 is a graph with 5 vertices $\{v_0, v_1, ..., v_4\}$ and 5 edges $\{e_0, e_1, ..., e_4\}$ with each e_i from v_i to $v_{(i+1) \bmod 5}$.

Counter-example: D_0 is not strongly connected.

A directed graph is strongly connected iff $A^{|V|-1}$ has no zero entries.

Weighted directed graphs

If $D = (V, E, \phi)$ is a directed graph, an edge-weighting of D is a function, λ, from E to some range set.

Shortest path problem

Suppose that weights represent distances.

And we want to know the minimal distance from one vertex, a, to each other vertex.

We will assume all distances are real and positive.

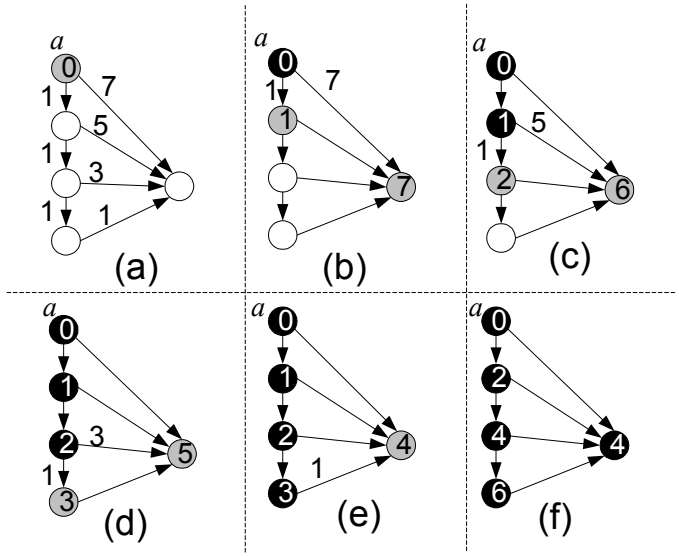
Define the weight of a directed walk $p = (v_0, e_1, v_1, ..., v_{i-1}, e_i, v_i, ..., e_k, v_k)$ as

$$\lambda(p) \triangleq \sum_{i=1}^{k} \lambda(e_k)$$

We want to calculate, for each z,

$$d(z) = \min_{p \text{ is a directed walk from } a \text{ to } z} \lambda(p)$$

- Start at a and work outward discovering shortest walks in the order of their size.
- Black vertices: we know $d(v)$.
- Grey vertices are one edge away from a black vertex. They are labeled by the weight of the best path we know that is all black up to the last vertex.
- At each step we pick the grey vertex that has the lowest label and turn it black.



Dijkstra's shortest path algorithm

---

for each $v \in V$ do $s(v) := \infty$
$s(a) := 0.0$; $Black := \emptyset$; $Grey := \{a\}$
// Invariant: $Black, Grey \subseteq V$ and $Black \cap Grey = \emptyset$
// and $\forall v \in Black, \forall w \in Grey, d(v) = s(v) \leq s(w)$
// and $\forall w \in Grey, s(w) =$ the length of the shortest path
// from a to w by a path that includes only Black
// vertices and w.
// and $\forall v \in V - Black - Grey, s(v) = \infty$
while $Grey \neq \emptyset$ do
  let v be a member of $Grey$ such that $s(v)$ is minimal
  for each vertex w s.t. there is an edge from v to w
    if $w \notin Black$ then
      $s(w) := s(w) \min(s(v) + \lambda(v, w))$
      $Grey := Grey \cup \{w\}$
  $Grey := Grey - \{v\}$
  $Black := Black \cup \{v\}$
// At this point $s(v) = d(v)$ for all $v \in V$.

Applications:

Dijkstra's shortest path algorithm is easily extended to find an optimal walk from a to each other vertex.

- Create a second graph with the same vertices and originally no edges.
- When a shorter path is found to a vertex w, (i.e. when $s(w) > (s(v) + \lambda(v, w))$) delete any edge that is to w and add a new edge from v to w.
- When done, the only path from a to any v in the new graph is a shortest path in the original

Thus it is very useful for plotting optimal routes

- In computer networks, we can find the fastest route.
- Robots can plot the shortest route to a goal while avoiding obstacles.
- Ships can plot courses that avoid sea ice.

It may seem wasteful to find the shortest route to all places, when one only wishes to go to one. One can stop when the destination becomes black. And there is a variant called A* that is more efficient, but requires more information.