Chapter 1

Graphs

1.1 Definitions

Graph properties

Definition 1. *(Graph)*
A graph is a tuple $G = (V, E)$ where V is a finite set of vertices and E is a finite collection of edges. Each edge is either an one or two element subset of V.

Definition 2. *(Self-Loop)*
If $G = (V, E)$ is a graph and $v \in V$ and $e = \{v\}$, then edge e is called a self-loop. That is, any edge that is a single element subset of V is called a self-loop.

We specify that E is a collection of one and two element subsets of V rather than to say that E was, itself, a set. This allows us to have duplicate edges in the edge set and thus to define multigraphs.

Definition 3. *(Multigraph)*
A graph $G = (V, E)$ is a multigraph if there are two edges e_1 and e_2 in E such that e_1 and e_2 are equal as sets. That is, there are two vertices v_1, v_2 in V so that $e_1 = e_2 = \{v_1, v_2\}$.

Definition 4. *(Simple) A graph $G = (V, E)$ is called simple if it is not a multigraph and it has no self-loops.*

Definition 5. *(Vertex adjacency)*
Let $G = (V, E)$ be a graph. Two vertices v_1 and v_2 are said to be adjacent if there exists an edge $e \in E$ so that $e = \{v_1, v_2\}$. A vertex v is self-adjacent if $e = \{v\}$ is an element of E.

Definition 6. *(Edge adjacency)*
Let $G = (V, E)$ be a graph. Two edges e_1 and e_2 are said to be adjacent if there exists a vertex v so that v is an element of both e_1 and e_2 as sets.

Definition 7. *(Neighborhood)*
Let $G = (V, E)$ be a graph and let $v \in V$. The neighbors of v are the set of vertices that are adjacent to v. Formally

$$N(u) = \{v \in V : \exists e \in E \quad s.t \quad e = \{v, u\} \quad or \quad u = v \quad and \quad e = \{u\}\}.$$

Definition 8. *(Degree) Let $G = (V, E)$ be a graph and let $v \in V$. The degree of $v \in V$ is the number of non-self-loop edges adjacent to v plus two times the number of self-loops defined at v. Formally*

$$deg(v) = |\{e \in E : \exists u \in V \quad s.t \quad e = \{u, v\}\}| + 2|\{e \in E : e = \{v\}\}|.$$

Example You are given the graph $V = \{1, 2, 3, 4\}$ and $E = \{\{1, 2\}, \{2, 3\}, \{3, 4\}, \{4, 1\}, \{1\}\}$ represent the graph and give the degree of each vertex.

Theorem 1. *Let $G = (V, E)$ be a general graph. Then*

$$2|E| = \sum_{V \in V} deg(v).$$

Proof. Consider two vertices v_1 and v_2 in V. If $e = \{v_1, v_2\}$ is an edge then $+1$ is added to the sum $\sum_{V \in V} deg(v)$ for both v_1 and v_2. Hence, every non self-loop is counted twice to the vertex sum and every self-loop vertex has degree $+2$. So each edge contributes exactly $+2$ to the sum. The result follows. □

Definition 9. *(Directed graph)*
A directed graph (digraph) is a tuple $G = (V, E)$ where V is a finite set of vertices and E is a collection of elements contained in $V \times V$. That is, E is a collection of ordered pairs of vertices. The edges in E are called directed edges. The source of the directed edge $e = (v_1, v_2)$ is v_1 while the destination of the edge is v_2.

Remark *It is worth noting that the ordered pair (v_1, v_2) is distinct from the pair (v_2, v_1). Thus if a digraph $G = (V, E)$ has both (v_1, v_2) and (v_2, v_1) in its edge set, it is not a multi-digraph.*

Definition 10. *(Underlying Graph) If $G = (V, E)$ is a digraph, then the underlying graph of G is the (multi) graph (with self-loops) that results when each directed edge (v_1, v_2) is replaced by the set $\{v_1, v_2\}$ thus making the edge non-directional. Naturally if the directed edge is a directed self-loop (v, v) then it is replaced by the singleton set $\{v\}$.*

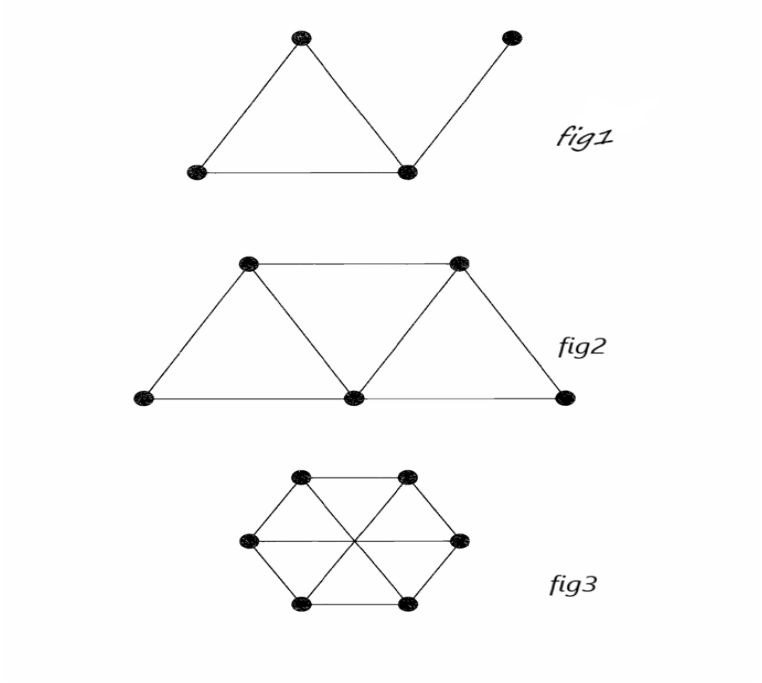
Remark Whether the underlying graph of a digraph is a multi-graph or not has to be specified by the author who has to state whether two directed edges (v_1, v_2) and (v_2, v_1) are combined into a single set $\{v_1, v_2\}$ or not.

Definition 11. *(In-degree, out-degree)*
Let $G = (V, E)$ be a digraph. The in-degree of a vertex v in G is the total number of edges in E with destination v. The out-degree of v is the total number of edges in E with source v. We will denote the in-degree of v by $deg_{in}(v)$ and the out-degree by $deg_{out}(v)$.

Theorem 2. *Let $G = (V, E)$ be a digraph. Then the following holds*

$$|E| = \sum_{v \in V} deg_{in}(v) = \sum_{v \in V} deg_{out}(v).$$

Definition 12. *(Subgraph) A subgraph of a graph G is a graph, each of whose vertices belongs to V and each of whose edges belongs to E. Thus the graph in Fig.1 is a subgraph of the graph in Fig.2, but is not a subgraph of the graph in Fig. 3, since the latter graph contains no triangle.*



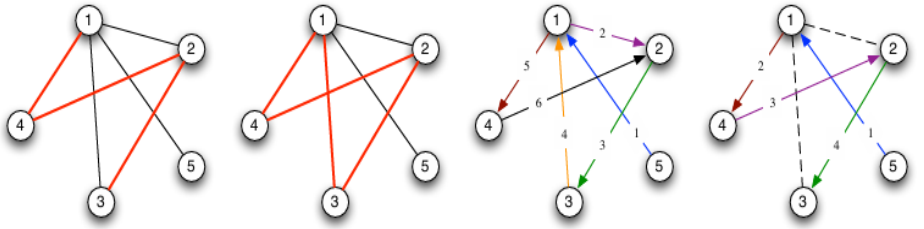
Definition 13. *(Walk) Let $G = (V, E)$ be a graph. A walk $w = (v_1, e_1, v_2, e_2, ..., e_n, v_{n+1})$ in G is an alternating sequence of vertices and edges in V and E respectively so that for all $i \in \{1, ..., n\}$, $e_i = \{v_i, v_{i+1}\}$. A walk is called closed if $v_1 = v_{n+1}$ and open otherwise. A walk consisting of only one vertex is called trivial.*

Definition 14. *(Length of a walk)*
The length of a walk w is the number of edges contained in it.

Definition 15. *(Path) Let $G = (V, E)$ be a graph. A path in G is a non-trivial walk with no vertex and no edge repeated. A Hamiltonian path is a path that contains exactly one copy of each vertex in V.*

Definition 16. *(Cycle) A closed walk of length at least 3 and with no repeated edges and in which the only repeated vertices are the first and the last is called a cycle. A Hamiltonian cycle is a cycle in a graph containing every vertex. A graph that contains a Hamiltonian cycle is called* ***Hamiltonian****.*

Definition 17. *(Trail, tour)*
*Let $G = (V, E)$ be a graph. A trail in G is a walk in which no edge is repeated. A tour is a closed trail. An Eulerian trail is a trail that contains exactly one copy of each edge in E and an Eulerian tour is a tour that contains exactly one copy of each edge. The graph is called **Eulerian** if it contains a Eulerian tour.*

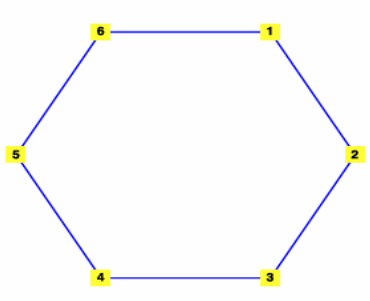


(a) Walk (b) Cycle (c) Eulerian Trail (d) Hamiltonian Path

The walk can be written as $w = (1, \{1, 4\}, 4, \{4, 2\}, 2, \{2, 3\}, 3))$ and the cycle $c = (1, \{1, 4\}, 4, \{4, 2\}, 2, \{2, 3\}, 3, \{3, 1\}, 1))$
Remark Let $G = (V, E)$ be a graph. To any walk $w = (v_1, \{v_1, v_2\}, v_2, ..., v_n, \{n, v_{n+1}\}, v_{n+1})$ we can associate a subgraph $G' = (V', E')$ with

$$\begin{cases} V' = \{v_1, ..., v_{n+1}\} \\ E' = \{\{v_1, v_2\}, ..., \{n, v_{n+1}\}\} \end{cases}$$

Definition 18. *(Cycle graph)*
Let $G = (V, E)$ be a graph with $|V| = n$. If w is a Hamiltonian cycle in G, H is the subgraph induced by w and $H = G$ then G is an $n-$cycle or a cycle graph with n vertices denoted by C_n.



(a) 6-cycle

Theorem 3. *If G is a graph in which the degree of each vertex is at least 2, then G contains a cycle.*

Proof. If G has any loops or multiple edges, the result is trivial. We can therefore suppose that G is a simple graph. Let v be any vertex of G. We construct a walk $v \to v_1 \to v_2 \to ...$ inductively by choosing v_1 to be any vertex adjacent to v and, for each $i > 1$, choosing v_{i+1} to be any vertex adjacent to v_i except v_{i-1} the existence of such a vertex is guaranteed by our hypothesis. Since G has only finitely many vertices, we must eventually choose a vertex that has been chosen before. If v_k is the first such vertex, then that part of the walk lying between the two occurrences of v_k is the required cycle. □

Remark The inverse of this theorem is not true.

Types of graphs

Definition 19. *(Empty and Trivial/Null Graphs)*
A graph $G = (V, E)$ in which $V = \emptyset$ is called the empty graph (or null graph). A graph in which $V = \{v\}$ and $E = \emptyset$ is called the trivial graph.

Definition 20. *(Isolated vertex)*
Let $G = (V, E)$ be a graph and let $v \in V$. If $deg(v) = 0$ then v is said to be isolated.

Definition 21. *(Complete graph) Let $G = (V, E)$ be a graph with $|V| = n$ with $n \geq 1$. A graph is complete, denoted K_n if each vertex is connected to every other vertex by an edge.*

Corollary 1. *In a complete graph of n vertices*

$$|E| = \frac{n(n-1)}{2}$$

Proof. Is left as an exercice. □

Definition 22. *(Regular graphs) A graph in which each vertex has the same degree is a regular graph. If each vertex has degree r, the graph is regular of degree r or r-regular. Of special importance are the cubic graphs, which are regular of degree 3, an example of a cubic graph is the Petersen graph. Note that the null graph is regular of degree 0, the cycle graph C_n is regular of degree 2, and the complete graph K_n is regular of degree $n - 1$.*

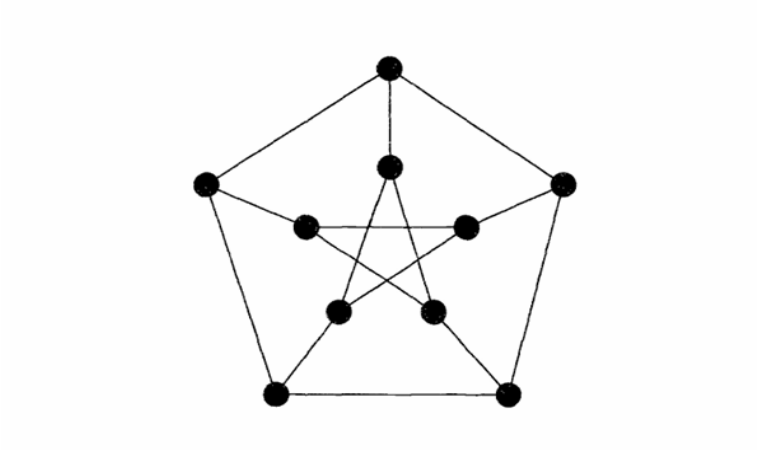
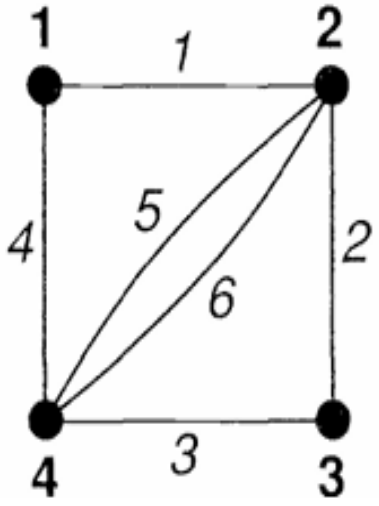


Figure 1.1: Petersen

Matrix representation of graphs

Although it is convenient to represent a graph by a diagram of points joined by lines, such a representation may be unsuitable if we wish to store a large graph in a computer.

Definition 23. *Let $G = (V, E)$ be a graph with $|V| = n$ and $|E| = m$ labelled as $\{1, 2, ..., n\}$ and $\{1, 2, ..., m\}$ respectively. The adjaceny matrix is an $n \times n$ matrix whose entry ij is the number of edges joining vertex i and vertex j.*



Give the adjacency matrix of the graph.

Proposition 1. *The adjacency matrix of a simple graph is symmetric.*

Theorem 4. *Let $G = (V, E)$ be a graph with $V = \{v_1, ..., v_n\}$ and let M be its adjacency matrix. For $k \geq 0$, the (i, j) entry of M^k is the number of walks of length k from v_i to v_j.*

Proof. We will prove the result by induction. For M^1 we have the adjacency matrix and the result holds by definition since a walk of length 1 is simply an edge. Now suppose that the (i, j) entry of M^k is the number of walks of length k from v_i to v_j. We will show that this is true for $k + 1$. We know that

$$M^{k+1} = M^k M$$

so the i, j element of M^{k+1} is

$$(M^{k+1})_{ij} = \sum_{l=1}^{n} (M^k)_{il} M_{lj}.$$

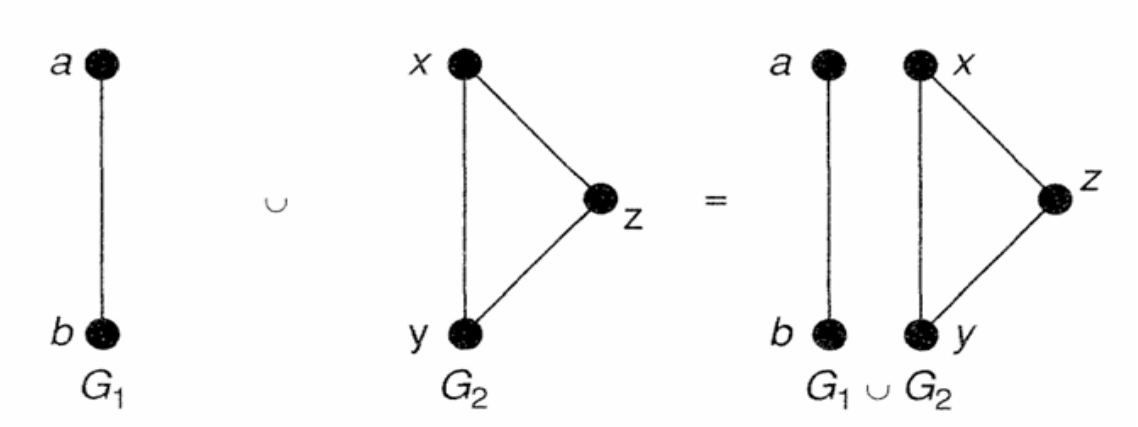
From the induction step, if you fix l, the element $(M^k)_{il}$ is the number of walks of length k from i to l. In that $(M^k)_{il} M_{lj}$ is the number of walks with length $k + 1$ from i to j passing from l. Adding for all possible l gives the result.

□

The definition can be extended to directed graphs. The theorem holds for digraphs as well.

Topology

Definition 24. *(Union of graphs) We can combine two graphs to make a larger graph. If the two graphs are $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, where V_1, V_2 are disjoint, then their union $G_1 \cup G_2$ is the graph with vertex set $V_1 \cup V_2$ and edge family $E_1 \cup E_2$.*



A graph is connected if it cannot be expressed as the union of two graphs, and disconnected otherwise. Clearly any disconnected graph G can be expressed as the union of connected graphs, each of which is a component of G.

Definition 25. *(Reachability)*
Let $G = (V, E)$ be a graph and v_1, v_2 two vertices. We say that v_2 is reachable by v_1 if there is a walk beginning in v_1 and ending in v_2. If the graph is directed, the walk is directed.

Definition 26. *(Connectedness)*
A graph G is connected if for every pair of vertices v_1 and v_2 in V, v_2 is reachable from v_1. If G is a digraph, then G is connected if its underlying graph is connected.

Definition 27. *(Strong connectedness)*
A digraph is strongly connected if for every pair of vertices v_1 and v_2 in V , v_2 is reachable (by a directed walk) from v_1.

Definition 28. *(Component)*
Let $G = (V, E)$ a graph and H a subgraph of G. H is a component of G if

- *H is conencted and*
- *if K another subgraph of G for which H is a proper subgraph, then K is not connected.*

The number of the components of G is denoted by $c(G)$.

Remark A connected graph has exactly one component.

Theorem 5. *If $G = (V, E)$ is a connected graph and $e \in E$. Then $G' = G - \{e\}$ is connected if and only if e lies on a cycle in G.*

Proof. Recall a graph G is connected if and only if for every pair of vertices v_1 and v_{n+1} there is a walk w from v_1 to v_{n+1} with

$$w = (v_1, e_1, v_2, ..., v_n, e_n, v_{n+1}).$$

Set $G' = G - \{e\}$. Suppose that e lies on a cycle c in G and choose two vertices v_1 and v_{n+1} in G. If e is not on any walk w connecting v_1 to v_{n+1} in G then the removal of e does not affect the reachability of v_1 and v_{n+1} in G'. Therefore assume that e is in the walk w. The fact that e is in a cycle of G implies we have vertices u_1,..u_m and edges f_1,...,f_m such that

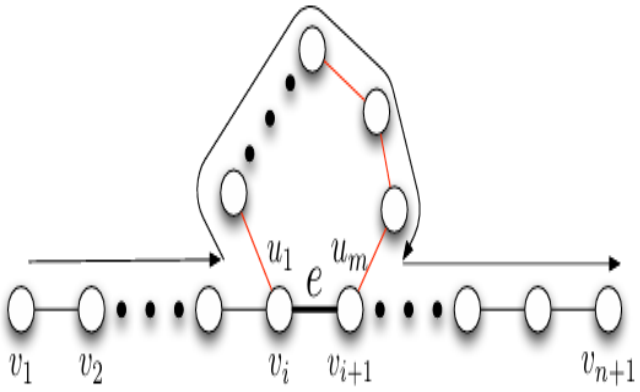
$$c = (u_1, f_1, ..., u_m, f_m, u_1)$$

and e is among the f_i. Without loss of generality, assume that $e = f_m$ and that $e = \{u_m, u_1\}$. (Otherwise, we can re-order the cyle to make this true.) Then in G' we will have the path

$$c' = (u_1, f_1, ..., u_m)$$

The fact that e is in the walk w implies there are vertices v_i and v_{i+1} so that $e = \{v_i, v_{i+1}\}$ (with $v_i = u_1$ and $v_{i+1} = u_m$). In deleting e from G we remove the sub-walk (v_i, e, v_{i+1}) from w but we can recreate a walk with

$$w' = (v_1, e_1, ..., v_i, f_1, u_2, ..., u_{m-1}, f_{m-1}, u_m, ..., e_n, v_{n+1}),$$



Inversely suppose G' is connected. Now let $e = \{v_1, v_{n+1}\}$. Since G' is connected, there is a walk from v_1 to v_{n+1} so there is a path p

$$p = (v_1, e_1, ..., v_n, e_n, v_{n+1}).$$

Since p is a path, there are no repeated vertices in p. We can construct a cycle c containing e in G as

$$p = (v_1, e_1, ..., v_n, e_n, v_{n+1}, e, v_1)$$

since $e = \{v_1, v_{n+1}\} = \{v_{n+1}, v_1\}$. Thus, e lies on a cycle in G. This completes the proof. □

Theorem 6. *Let G be a simple graph with n vertices. If G has k components then the number of edges m of G satisfies*

$$n - k \leq m \leq \frac{(n-k)(n-k+1)}{2}$$

Proof. For the upper bound we proceed as follows. Assume that all components are complete and take two if them C_i, C_j with $n_i \geq n_j > 1$ vertices each. Move one vertex from the one to the other and replace C_i, C_j by complete graphs with $n_i + 1$ and n_{j-1} vertices. The total number of vertices does not change but the number of edges changes by

$$\frac{1}{2}(n_i+1)n_i + \frac{1}{2}(n_j-1)(n_j-2) - (\frac{1}{2}n_i(n_i-1) + \frac{1}{2}(n_j(n_j-1))) = n_i - n_j + 1 > 0.$$

Which means that the number of edges increases when we move vertices from one component to the other, hence to attain the maximum number of edges, G must consist of a complete graph on $n-(k-1)$ vertices and $k-1$ isolated vertices. In that case the number of edges is

$$\frac{(n-k+1)(n-k+1-1)}{2} = \frac{(n-k)(n-k+1)}{2}.$$

For the lower bound proceed by induction on the number of edges. If $m = 0$ this is the null graph and the result is trivial with $k = 0$. Otherwise, assume true for G with m_0 edges, the fewer possible so that it remains connected with this number of edges. Removing one edge increases the number if components by 1. Then, the graph has n vertices, $m_0 - 1$ edges, $k + 1$ components and by the induction step

$$m_0 - 1 \geq n - (k + 1) \implies m_0 \geq n - k.$$

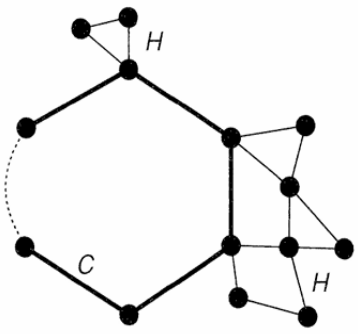
□

Corollary 2. *Any graph with n vertices and more than* $\frac{(n-1)(n-2)}{2}$ *edges is connected.*

Theorem 7. *(Euler)*
A connected graph is Eulerian if and only if the degree of each vertex of G is even.

Proof. Suppose the graph is Eulerian. It contains then a closed walk P where no edge is repeated and every edge is visited exactly once. Every time an edge passes through a vertex it contributes twice to its degree. In a Eulerian graph every edge is visited once so the degree of each vertex is even. Assume now that the degree of all vertices is even. We will construct a Eulerian tour preceeding by induction on the number of edges.

Since G is connected, each vertex has degree at least 2 and so it contains a cycle C. If C contains every edge of G, the proof is complete. If not, we remove from G the edges of C to form a new, possibly disconnected, graph H with fewer edges than G and in which each vertex still has even degree. By the induction hypothesis, each component of H has an Eulerian tour. Since each component of H has at least one vertex in common with C, by connectedness, we obtain the required Eulerian tour of G by following the edges of C until a non-isolated vertex of H is reached, tracing the Eulerian tour of the component of H that contains that vertex, and then continuing along the edges of C until we reach a vertex belonging to another component of H, and so on. The whole process terminates when we return to the initial vertex.



□

Definition 29. *(Distance)*
Let $G = (V, E)$ be a graph. The distance between v_1 and v_2 in V is the length of the shortest walk beginning at v_1 and ending at v_2 if such a walk exists. Otherwise, it is $+\infty$. We will write $d_G(v_1, v_2)$ for the distance from v_1 to v_2 in G.

In the case of a digraph we talk about directed distance induced by the shortest directed walk.

Definition 30. *(Diameter)*
The diameter of G $diam(G)$ is the length of the largest distance in G. That is

$$diam(G) = max_{v_1,v_2 \in V} d_G(v_1, v_2).$$

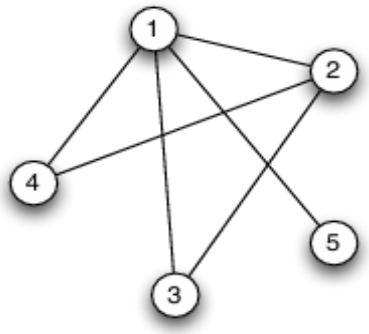
Definition 31. *(Eccentricity)*
Let G be a graph and v_1 a vertex of G. The eccentricity of v_1 is the largest distance of v_1 to any other vertex v in V,

$$ecc(v_1) = max_{v \in V} d_G(v_1, v).$$

Definition 32. *(Radius)*
The radius of a graph is the smallest of all the eccentricities of all the vertices,

$$rad(G) = min_{v \in V} ecc(v).$$

Definition 33. *(Girth/ Circumference)*
Let G be a graph. If there is a cycle in G (that is there is a cycle graph as subgraph) then the girth (circumference respectively) is the length of the shorthest (longest respectively) cycle. When G contains no cycle this is defined as 0 or infinity respectively.



In this graph we have $ecc(v_1) = 1, ecc(v_2) = ecc(v_3) = ecc(v_4) = ecc(v_5) = 2$ so the radius which is the min of these values is 1. The maximum distance (distance=shortest path) is 2 so the diameter is 2. The circumferene is 4 since the largest cycle that we can find goes from 1 to 4, to 2 to 3 and then back to vertex 1. There are many 3 cycles so the girth is 3.

Centrality, acyclic graphs, trees

Centrality is a measure of importance of a vertex. There are two types.

Definition 34. *(Degree centrality)*
If $G = (V, E)$ is a graph, the degree centrality os vertex $v \in V$ is the normalized degree of v,

$$\frac{deg(v)}{2|E|} \in [0, 1].$$

Note that the sum of all centralities is 1.

Another measure is the geodesic centrality which will be defined step by step as follows

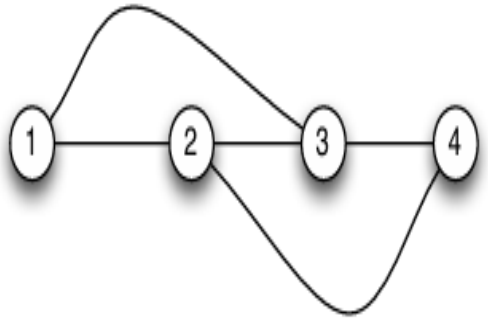
- Denote σ_{st} the total number of shortest paths connecting vertex s to vertex t.
- Denote $\sigma_{st}(v)$ the total number of these shortest paths passing from v.
- The quotient $\frac{\sigma_{st}(v)}{\sigma_{st}}$ represents the percentage of the shortest paths from s to t containing v.

Definition 35. *(Geodesic centrality)*
Let $G = (V, E)$ be a graph. The geodesic centrality (sometimes called the betweeness) of a vertex $v \in V$ is the fraction of times v occurs on any shortest path connecting any other pair of vertices $s, t \in V$. Formally

$$C_B(v) = \sum_{s \neq t \neq v} \frac{\sigma_{st}(v)}{\sigma_{st}},$$

where the subscript B stands for betweeness. To get a normalized centrality in $[0, 1]$, this number is divided by $C_B(ALL) = \sum_v C_B(v)$.

Example Consider the graph below and calculate the centrality and geodesic centrality of each vertex.



We have that $deg(1) = 2 = deg(4), deg(2) = 3 = deg(3)$ for a total of 10 with corresponding centralities $\frac{2}{10}$ and $\frac{3}{10}$. For the geodesic centralit we will compute the number of shortest paths for each paire of vertices. We have

- From 1 to 2 there is one shortest path of length 1, $1 \to 2$.
- From 1 to 3 there is one shortest path of length 1, $1 \to 3$.
- From 1 to 4 there are two shortest paths of length 2, $1 \to 2 \to 4$ and $1 \to 3 \to 4$.
- From 2 to 3, one shortest path of length 1, $2 \to 3$
- From 2 to 4 one shortest path of length 1, $2 \to 4$.
- From 3 to 4 one shortest path of length 1, $3 \to 4$. It is easy to see that

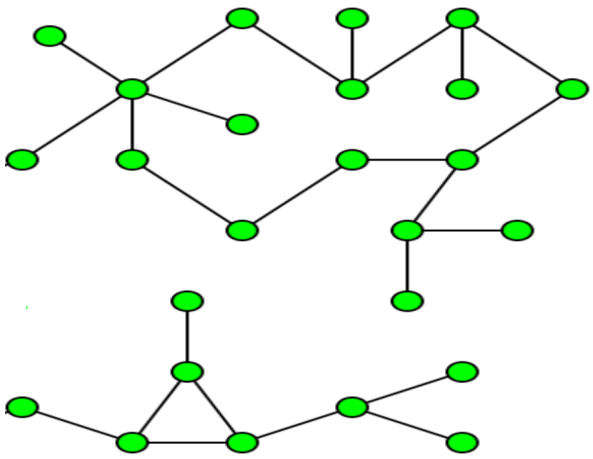
  $$\sigma_B(1) = \sigma_B(4) = 0$$

  since there is no path having 1 or 4 in between. Then
  - $\sigma_B(2) = \sum_{s \neq t \neq 2} \frac{\sigma_{s,t}(2)}{\sigma_{s,t}} = \frac{\sigma_{1,4}(2)}{\sigma_{1,4}} = \frac{1}{2}$.
  - $\sigma_B(3) = \sum_{s \neq t \neq 3} \frac{\sigma_{s,t}(3)}{\sigma_{s,t}} = \frac{\sigma_{1,4}(3)}{\sigma_{1,4}} = \frac{1}{2}$.

Definition 36. *(Acyclic)*
A graph is called acyclic if it has no cycles.

Definition 37. *(Tree, forest)*
Let $G = (V, E)$ be an acyclic graph. If G has one component is called a tree, otherwise it is a forest.

Definition 38. *(Spanning tree, spanning forest)*
Let $G = (V, E)$ be a graph. If $F = (V', E')$ is an acyclic subgraph of G such that $V = V'$ then F is called a spanning forest of G. If F has exactly one component, then F is called a spanning tree.



The graph with two components is a spanning forest and each component is spanning tree.

Theorem 8. *Every connected graph $G = (V, E)$ has a spanning tree $T = (V, E')$.*

Corollary 3. *Every graph has a spanning forest.*

Definition 39. *(Leaf)*
Let $T = (V, E)$ be a tree. If $v \in V$ and $deg(v) = 1$, then v is called a leaf of T.

Lemma 1. *Every tree with one edge has at least two leaves.*

Proposition 2. *Let $T = (V, E)$ be a tree. If $|V| = n$ then $|E| = n - 1$.*

Corollary 4. *Let $G = (V, E)$ be a forest. If $|V| = n$ and $c(G)$ the number of connected components, then $|E| = n - c(G)$.*

This corolorary tells us in particular the number of components in a graph, since given G we are given $|V|$ and $|E|$.

Theorem 9. *A graph is connected if and only if it has a spanning tree.*

Proof. Any connected graph has a spanning tree by the theorem. Inversely assume a graph $G = (V, E)$ has a spanning tree T. Then T has $|V|$ vertices and T is connected so there is a walk between any two vertices, so a walk between any two vertices of G. This completes the proof. □

Graph Laplacian

Definition 40. *(Degree matrix)*
Let $G = (V, E)$ be a simple graph. The degree matrix is the diagonal matrix D that has on its diagonal the degree of the vertices. D_{ii} is the degree of vertex v_i.

Definition 41. *(Graph Laplacian)*
Let $G = (V, E)$ be a simple graph with adjacency matrix A and degree matrix D. The laplacian matrix is defined as

$$L = D - A.$$

Example You are given a graph with the following adjacency matrix

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

find the laplacian matrix.

Proof. Adding the rows of the adjacency matrix of a simple graph gives the degree of the vertices. Hence

$$D = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

and the laplacian is

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

□

Notice that the row sum of the laplacian is 0 for each row and that L is symmetric. This is a general fact.

Proposition 3. *The laplacian matrix of a simple graph is symmetric.*

Proof. This is evident since D and A are both symmetric.

$$L^T = (D - A)^T = D^T - A^T = D - A = L.$$

□

Proposition 4. *The row-sum for each row of the Laplacian matrix of a simple graph is zero.*

Proposition 5. *Let L be the Laplacian matrix of a simple graph G with n vertices. 0 is always an eigenvalue of L with associated eigenvector $v_0 = (1, .., 1) \in \mathbb{R}^n$. The algebraic multiplicity of 0 is the number of components of G.*

Proof. The product Lv_0 gives the zero matrix since $(Lv_0)_i = 0$ for each line i by definition of the laplacian. We prove now that the algebraic multiplicity of 0 is equal to the number of connected components of G. For that, assume that G has k components C_i, each of these has n_i vertices and L_i is the corresponding laplacian matrix, $i = 1, 2, .., k$. This gives L in blocks

$$L = \begin{bmatrix} L_1 & 0 & 0 & ... \\ 0 & L_2 & 0 & ... \\ .... & & & \\ 0 & 0 & ... & L_k \end{bmatrix}$$

The fact that 1_i (a vector of $1's$ with dimension appropriate to L_i) is an eigenvector for L_i with eigenvalue 0 implies that $v_i = (0, 0, ...1_i, ...0)$ is an eigenvector for L with eigenvalue 0. Thus, L has eigenvalue 0 with multiplicity at least k. In that, if v is an eigenvector associated to $\lambda = 0$, then $v \in Ker(L)$. The kernel has then at least k vectors

$$dim(kerL) \geq k.$$

On the other hand, since all L_i are symmetric, their rank is $n_i - 1$ (a symmetric matrix is written like PDP^{-1} with P invertible and $rankD$= the number of non zero eigenvalues of D). This means that the rank of L is

$$n_1 - 1 + ... + n_k - 1 = n - k$$

so

$$dim(Col(L)) = n - k$$

giving from the dimension theorem that

$$dim(ker)) = n - (n - k) = k$$

and the multiplicity of 0 is exactly k. □

Proposition 6. *Let G be a graph with Laplacian matrix L. The eigenvalues of L are all $\lambda \geq 0$.*

Definition 42. *Let G be a simple graph with with n vertices and L its laplacian matrix whose eigenvalues are $\lambda_1 \leq \lambda_2 \leq ....\lambda_n$. The second smallest eigenvalue λ_2 is called the Fiedler value and its corresponding eigevector the Fiedler vector.*

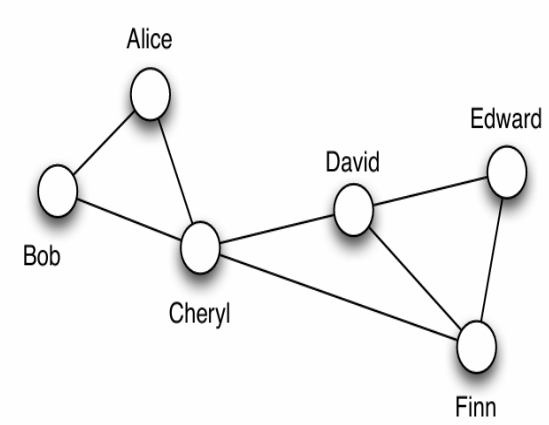
Proposition 7. *Let G be a graph with Laplacian matrix L. The Fiedler value $\lambda_2 > 0$ if and only if G is connected.*

Proof. If G is connected it has one component and the multiplicity of 0 is 1 making the second smallest eigenvalue strictly positive. Inversely if $\lambda_2 > 0$ then 0 being always an eigenvalue, only $\lambda_1 = 0$ so the multiplicity is 1. □

Theorem 10. *(Spectral clustering) Let G be a graph with laplacian L and set of vertices $V = \{v_1, ..., v_n\}$. If v_F is the Fiedler vector and c a constant then the set of vertices $V_c = \{v_i \in V : v_{F_i} \geq c\}$ and the edges between these vertices form a connected subgraph.*

For example for $c = 0$ we can separate the components in positive and negative and the corresponding vertices give two clusters. The algorithm can be generalized for k clusters using the smallest k positive eigenvalues but it is mostly efficient for 2 clusters. For k large it fails. We will be using the fiedler vector for spectral clustering with 2 clusters.

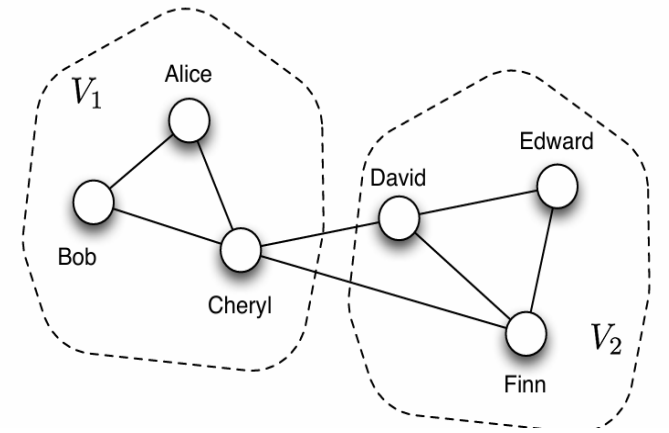
Example The following example illustrates how spectral clustering works.



Consider v_i with alphabetic order. If we compute the Fiedler value for this graph we see it is $\lambda_2 = 3 - \sqrt{5} > 0$ with corresponding Fiedler vector

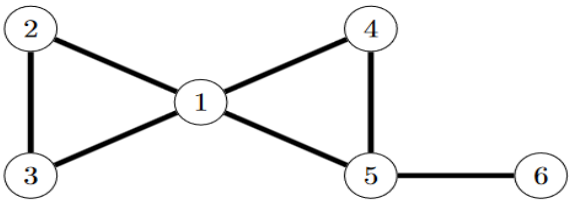
$$v_F = (\frac{-1-\sqrt{5}}{2}, \frac{-1-\sqrt{5}}{2}, \frac{\sqrt{5}-3}{2}, 1, \frac{1+\sqrt{5}}{2}, 1)$$

Positive components are the last 3 and group David, Edward and Finn together while positive components (the first 3) group together Aice, Bob and Cheryl. That is we have set $c = 0$ and applied the Fiedler decomposition.



It is worth noting that if an entry is 0, the corresponding vertex can be placed in either partition or in a partition of its own. When we partition a graph, a certain number of edges must be removed.

Look at this example



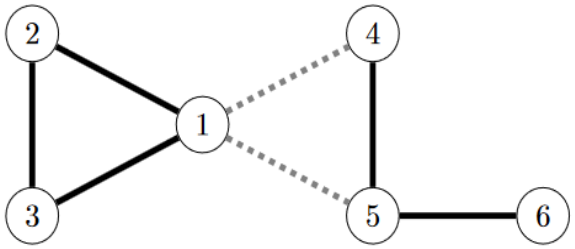
Laplacian

$$L = \begin{bmatrix} 4 & -1 & -1 & -1 & -1 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 \\ -1 & -1 & 2 & 0 & 0 & 0 \\ -1 & 0 & 0 & 2 & -1 & 0 \\ -1 & 0 & -1 & -1 & 3 & -1 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

eigenvalues

$$0, 0.6314, 1.4738, 3, 3.7877, 5.1071$$

The Fiedler vector is $v_F = \{-0.16, -0.44, -0.44, 0.07, 0.26, 0.71\}$ which gives



This is not the only way to cluster. We could have separated the vertices by keeping vertex 6 alone but Fiedler gives a more balanced clustering. In practice we look for good sized clusters, where the clusters have almost the same number of elements. In a clustering of vertices V we separate V in two groups $G_1 = \{a_1, ..., a_n\}$ and $G_2 = \{b_1, ..., b_m\}$ with n close to m $G_1 \cup G_2 = V$ and $G_1 \cap G_2 = \emptyset$. This turns into a minimization problem, where we look for locations x, y such that the sum of the distances of each a_i to x and each b_i to y is minimal,

$$min \sum_{i=1}^{n} ||a_i - x||^2 + \sum_{i=1}^{m} ||b_i - y||^2.$$

Remark A real symmetric matrix has a basis of real orthogonal eigenvector.

Shortest paths

Definition 43. *(Weighted graph)*
A weighted graph is $G = (V, E, w)$ where w is a weight function

$$w : E \to \mathbb{R}.$$

Remark A Markov chain is an example of a weighted graph where the probabilities play the role of the edge weights.

Remark Any graph can be thought of as a weighted graph in which we assign the weight of 1 to each edge. In that case the distance from a vertex v_1 to a vertex v_2 is the weight of the path linking the two vertices. We can generalize to weighted graphs. If

$$p = (v_1, e_1, v_2, ..., e_n, v_n)$$

is a path, the weight of the path is

$$w_p = \sum_{i=1}^{n} w(e_i).$$

The shortest path problem in a weighted graph is the problem of finding the least weight path p linking a given vertex u to another vertex v.

Dijkstra

Dijkstra's Algorithm finds the shortest path by growing a spanning tree in a graph with positive weights. The algorithm works by starting at u and choosing the shortest path to any vertex. Define

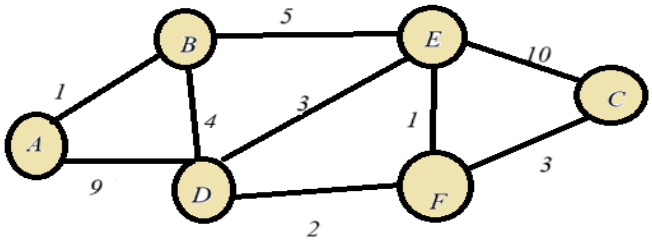
$$\delta(u, v) = \begin{cases} min\{w(p), u \rightarrow^p v\}, & \textit{if any such path } p \textit{ exists} \\ +\infty \quad \textit{otherwise.} \end{cases}$$

The goal is to find δ. With Dijstra we find both δ and the path. Initialize δ for all vertices u to $+\infty$ since we found no path yet from the source u to other vertices. We want to reduce this inifnity for paths that are reachable from the source.

While growing the tree we keep track of some information concerning the path

- The nodes that have not been visited yet.
- The nodes that have been visited.

Example Calculate the shortest distance from A to any other node of the following graph.



Node	Current distance with source A
A	0
B	$+\infty$
C	$+\infty$
D	$+\infty$
E	$+\infty$
F	$+\infty$

- Step 1. Visited= $\emptyset$ and NonVisited=$\{A, B, C, D, E, F\}$. Current node=A. Its neighbors are B and D with respective distances $1 < +\infty$ and $9 < +\infty$ smallest than the current shortest distances so replace them in the table.

Node	Current distance with source A	Previous node
A	0	
B	1	A
C	$+\infty$	
D	9	A
E	$+\infty$	
F	$+\infty$	

Then Visited= A and NonVisited=$\{B, C, D, E, F\}$.

- Step 2. As next node always choose the one with minimal distance to the visited node. Hence current node=B whose neighbors are E and D with corresponding distances $1 + 5$ and $1 + 4$ smallest than their current shortest distances that are infinity and 9 respectively.

Node	Current distance with source A	Previous node
A	0	
B	1	A
C	$+\infty$	
D	5	B
E	6	B
F	$+\infty$	

Now the visited nodes are Visited=$\{A, B\}$ and NonVisited=$\{C, D, E, F\}$

- Step 3. Current node=D (with weight 5) since it has the minimal distance to B from the unvisited nodes. Neighbors E and F with corresponding distances $5 + 3$ and $5 + 2$. 8 is largest than the current shortest distance of E, which is 6, and 7 is smallest than $+\infty$ so update only F.

Node	Current distance with source A	Previous node
A	0	
B	1	A
C	$+\infty$	
D	5	B
E	6	B
F	7	D

Visited=$\{A, B, D\}$, nonVisited=$\{C, E, F\}$.

- Current node=E since its current distance from the source is smaller than F's distance. Neighbors F, C with distances $6 + 1 = 7$ exactly the same as previously (we don't update anything) and $6 + 10 < \infty$ and update.

Node	Current distance with source A	Previous node
A	0	
B	1	A
C	16	E
D	5	B
E	6	B
F	7	D

Visited=$\{A, B, D, E\}$, nonVisited=$\{C, F\}$.

- Minimal distance to the source has F so start with current node=F. Distance to C is now $10 < 16$, update C.

Node	Shortest distance with source A	Previous node
A	0	
B	1	A
C	10	F
D	5	B
E	6	B
F	7	D

Visited=$\{A, B, D, E, F\}$, nonVisited=$\{C\}$.

- The last node is C which has no unvisited neighbors. We are done.

This table has the shortest distances from A to any other node. If one wants to determine a path from A to another node, say C, start at C and by backwards substitution find the node that led to C, that is F, the node that led to F is D, before D it was B and before B it was A. The path is

$$A \to B \to D \to F \to C.$$

Remark If at a current node the distance turns out to be the same as from the previous node we don't update anything.

Theorem 11. *Let (G, V, E, w) be a weighted graph with vertex v_0. Then Dijkstra's algorithm returns a spanning tree T so that the distance from v_0 to any vertex v in T is the minimum distance from v_0 to v in (G, V, E, w).*

We will now construct a pseudo-algorithm for Dijkstra'a method. Choose a source vertex S and call $d[v]$ the length of the current shortest path from S to v. The length

to the source is 0 so $d[S] = 0$ and to all the other vertices is initialized to infinity. So $d[v] = +\infty$ initially.

Then we wish to reduce these d values to what we call δ values that is

$$\delta(S, v) = \textit{the lenght of a shortest path from } S \textit{ to } v.$$

When all vertices have values of d convergent to δ the algorithm is done. To recontruct the path we need the predecessors which will be stored in

$$\Pi(v) = \textit{the predecessor of } v \textit{ in the shortest path from } S \textit{ to } v.$$

Starting from a vertex and updating the d value of its adjacent vertices is called relaxation. This process works as follows.

$$\begin{aligned} \textit{Relax}(u, v, w): \quad \textit{if} \quad d[v] > d[u] + w(u, v) \quad \textit{then} \\ d[v] = d[u] + w(u, v) \\ \Pi(v) = u \end{aligned}$$

which tells us that we found a better way of reaching v and that is through u. Note that we want to be able to converge from the top, that means that we want to make sure that relaxation never creates a d that is smaller than the δ.

Lemma 2. *(Relaxation is safe)*
The relaxation algorithm maintains the invariant that

$$d[v] \geq \delta(S, v)$$

for all $v \in V$.

Proof. We will do the proof by induction on the number of steps. At the first step we are in S and it is obvious. By the induction step $d[u] \geq \delta(S, u)$. From the triangle inequality

$$\delta(S, v) \leq \delta(S, u) + \delta(u, v)$$

since $\delta(S, v)$ is the length of the shortest path from S to v whereas on the right hand side we have another way to get to v passing from u which is certainly longest than the shortest way. By assumption $\delta(S, u) \leq d[u]$ and $\delta(u, v) \leq w(v, u)$ again since δ is the shortest path. Finally $d[u] + w(v, u) = d[v]$ and we are done.

□

Now we can define the Dijkstra's algorithm. $G = G(V, E)$ the graph, w the weights and S the starting vertex.

- Dijkstra(G, w, S):

$$\begin{aligned} d[S] = 0, d[v] = +\infty, v \neq S \\ Visisted \leftarrow \emptyset, nonVisited \leftarrow V \end{aligned}$$

- While $nonVisited \neq \emptyset$

$$u \leftarrow extract.min(nonVisited)$$
$$Visited.append(u)$$
$$for\ each\ vertex\ v\ \in adj(u):$$
$$Relax(u, v, w)$$

- end

The command extract.min(nonVisited) takes out from the nonVisited list a priority vertex whose distance to the source is minimal.

PageRank

Definition 44. *(Eigenvector centrality)*
Let $G = (V, E)$ be a graph with adjacency matrix $A = (a_{i,j})_{ij}$. The eigenvector centrality of a vertex v is the number

$$x_v = \frac{1}{\lambda} \sum_{u \in N(v)} x_u = \frac{1}{\lambda} \sum_{t \in V} a_{v,t} x_t,$$

where $N(v)$ is the set of neighbors of v and λ a constant to be precised.

Cetrality is a measure of importance of the vertex and in that context a vertex is considered important if it is adjacent to other important vertices. Observe that if we repeat this for all the components of the vector $x = (x_1, ..., x_{|V|})$ what we get is

$$x = \frac{1}{\lambda} Ax$$

which implies that x is the eigenvector of A associated to the eigenvalue λ. The eigenvalue of A that should be chosen to appropriately estimate the importance of each vertex is the one whose associated eigenvector has positive entries. The following theorem guarantees the existence and the uniqueness of such an eigenvalue and eigenvector.

Definition 45. *(Irreducible matrix)*
A square matrix A is irreducibe if for all i, j there exists some k such that $A^k_{i,j} > 0$.

Remark The adjacency matrix is irreducible.

Theorem 12. *(Perron-Frobenius theorem) Let M be an irreducible matrix. Then M has an eigenvalue λ_0 with the following properties.*

- *$\lambda_0 > 0$ and dominates all other eigenvalues of M, in the sense that if λ is an eigenvalue of M then $\lambda_0 \geq |\lambda|$.*

- *The eigenvector v_0 associated to λ_0 has only positive entries when properly scaled.*
- *λ_0 is a simple eigenvalue hence the vector v_0 is unique up to scalar.*
- *v_0 is the only vector up to scaling that has these properties.*

Theorem 13. *(Power method-vector form)*
Let $G = (V, E)$ be a graph with adjacency matrix A. Take λ_0 and v_0 the Perron-Frobenius eigenvalue and its corresponding vector. Further assume $|\lambda_0| > |\lambda|$ for all the other eigenvalues of A. If $x \in \mathbb{R}^n$ is a column vector such that $< x, v_0 > \neq 0$ then

$$lim_n \frac{A^n x}{\lambda_0^n} = a_0 v_0,$$

where a_0 is the coefficient of $x = a_0 v_0 + ... + a_n v_n$ in the decomposition of x in the basis of eigenvectors.

Remark A basis of eigenvectors exists since the adjacency matrix is symmetric. The power method, which approaches the largest eigenvalue of a matrix, also gives the corresponding eigenvector as a limit vector. If every iterate is normalized then the limit vector is unit.

Definition 46. *(Discrete Markov chain)*
A discrete time Markov chain is a directed graph $G = (V, E, p)$ where p are the weights of G, with $p \in [0, 1]$. p is interpreted as the probabilty to travel from the one edge to the other. The vertices are called states and the edges transitions. For all neighbors of a vertex v we have

$$\sum_{u \in N(v)} p(v, u) = 1.$$

$N(v)$ is the neighborhood reachable by out-edge from v. If there is no edge (v, u) the corresponding edge weight is 0.

Definition 47. *(Stochastic matrix)*
Let $G = (V, E, p)$ be a Markov chain. The stochastic matrix of G, or probability transition matrix, is the matrix M

$$M_{i,j} = p(u_i, u_j).$$

By definition, the row sum of the stochastic matrix is 1 (for all vertices apart from those without arrows leaving from them). M is an analogue of the adjacency matrix, where instead of 0 and 1 to indicate adjacency, we have the transition probability from u_i to u_j. They have zeros at the same position.

For a particle moving on the graph we can associate a probability of being at a given vertex at a given time.

Definition 48. *(State probability vector)*
Let G be a Markov chain and x a vector $x = (x_1, x_2, ..., x_n)$. x is a state vector if x_i is the probability of being at the state i (vertex v_i) and $x_1 + ... + x_n = 1$.

Definition 49. *Let $G = (V, E, p)$ be a Markov chain with stochastic matrix M and x^0 an initial probability state. Assume that we take a walk of length k then the final probability state is $x^k = (M^T)^k x^0$.*

Definition 50. *(Stationary vector)*
Let G be a vector chain with stochastic matrix M. A stationary vector x^* is a vector that satisfies

$$x^* = M^T x^*.$$

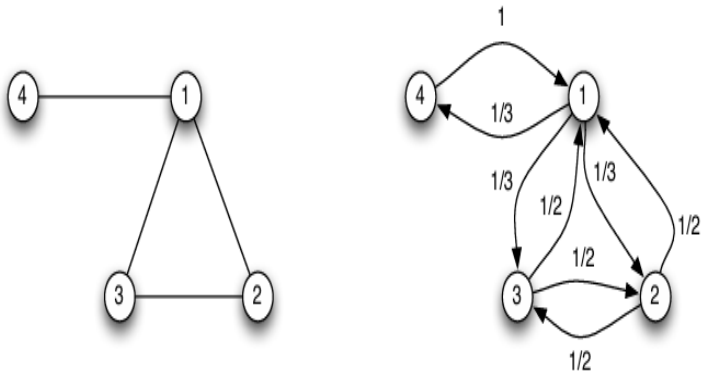
Note that if M^T is irreducible, then the Perron-Frobenius theorem applies. Moreover if the dominant eigenvalue is 1 then the stationary vector is the eigenvector associated to $\lambda = 1$.

Theorem 14. *Let $G = (V, E, p)$ be a Markov chain and M its stochastic matrix. If G is strongly connected then M^T and M are irreducible.*

Theorem 15. *Let $G = (V, E, p)$ be a Markov chain and M its stochastic matrix being irreducible. Then M has $\lambda = 1$ as dominant eigenvalue and a unique stationary probability distribution x^0 which is the eigenvector associated to $\lambda = 1$.*

Definition 51. *(Induced Markov chain)*
Let $G = (V, E)$ be a graph. The Markov chain induced by G is the digraph obtained from G when every edge $e = \{u, v\}$ is replaced by two directional edges (u, v) and (v, u) and the probability function p is defined by

$$p(u, v) = \frac{1}{deg_{out}(u)}.$$



For the above non directed graph we have an adjacency matrix

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}.$$

For the digraph the adjacency matrix is the same and the stochastic matrix is

$$M = \begin{bmatrix} 0 & 1/3 & 1/3 & 1/3 \\ 1/2 & 0 & 1/2 & 0 \\ 1/2 & 1/2 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}.$$

The stationary vector of M^T is $x^* = (3/8, 2/8, 2/8, 1/8)$ which is the eigenvector corresponding to the eigenvalue $\lambda = 1$ of M^T. From the power method

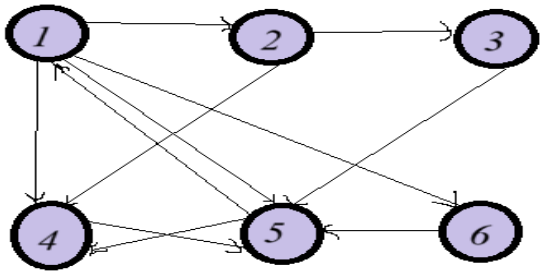
$$x^* = lim_n \frac{(M^T)^k x}{1^k} = a_0 x_0$$

and $x^* = \frac{a_0 x_0}{||a_0 x_0||}$. We can use this vector to clssify the vertices according to their importance. For instance with x^* the most important vertex is 1, 2 and 3 are equally important and the less important is vertex 4. We would get the same ordering using the eigenvector centrality with the eigenvector of A associated to its largest eigenvalue $\lambda = 2.17$ being $v_0 = (0.6116, 0.5227, 0.5227, 0.2818)$.

PageRank Consider a web with n websites and on each site their is/are a link/s leading to some other/s websites. Google ranks the importance of the websites using an algorithm called Page Rank which uses the idea of eigenvector centrality. We can see the net as a big network represented by a digraph, where an arrow from v_i to v_j means that there is a link in page i leading to page j. A page on web is considered as imporant based on how often it is visited and this frequency is based on the importance of the websites leading to it. For instance, if page i is exclusively accessible by page j but page j leads to 3 more pages, then the importance of i is shared by 4. Equivalently to the eiganvector centrality (where the importance of a vertex is defined in a self-referential way using the centrality of its neighbors), page rank represents the net as a Markov chain with probability function the one that uniformly distributes the weight of an edge according to the number of its neighbors.

Imagine a surfer who will click among these webpages following links until a dead-end is reached (a page with no out bound links). In this case, the websurfer will type a new URL chosen from the pages available) and the process will continue. Using uniform probability is not exclusive, but unless otherwise stated we take equal scores.

Example
Consider a network with 6 pages



To associate the probabilities we will use uniform distribution. For instance 2 is exclusively accesible by 1 and 1 gives access to 3 others so $p(1,2) = 1/4$. Similarly 2 gives access to 3 and 4 so $p(2,3) = 1/2$ and $p(2,5) = 1/2$. We continue like this and this gives $p(1,6) = p(1,4) = p(1,5) = 1/4$, $p(4,5) = 1$ and $p(5,4) = p(5,1) = 1/2$. Once the weights determined we can find the importance of each vertex (taking into account the vertices that lead to this vertex). Note $R = (R_1, R_2, R_3, R_4, R_5, R_6)$ the rank vector. Hence, $R_1 = (1/2)R_5, R_2 = (1/4)R_1, R_3 = (1/2)R_2, R_4 = (1/4)R_1 + (1/2)R_2 + (1/2)R_5, R_5 = (1/4)R_1 + R_3 + R_4 + R_6, R_6 = (1/4)R1$.

$$R = \begin{bmatrix} 0 & 0 & 0 & 0 & 1/2 & 0 \\ 1/4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 0 & 0 & 0 \\ 1/4 & 1/2 & 0 & 0 & 1/2 & 0 \\ 1/4 & 0 & 1 & 1 & 0 & 1 \\ 1/4 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} * R$$

The matrix that appears is the M^T (note that the column sum is 1) whose dominant eigenvalue is $\lambda = 1$. Either solve for the corresponding eigenvector R^* which will give the ranking of the vertices or approximate this eigenvector with the power method. For the power method an initial guess for R_0 must be done, typically we take the vector of equiprobabilities so that no bias is introduced from the first step (but whatever choice is acceptable as long as R_0 is not chosen in the

space spanned uniquely by the other eigenvectors apart from R^*). Numerically we can solve for R^* by taking $R_0 = (1/6, 1/6, 1/6, 1/6, 1/6, 1/6)$ and searching for

$$lim_n \frac{(M^T)^n R_0}{1^n}$$

the result is $R^* = (0.2, 0.05, 0.025, 0.275, 0.4, 0.05)$ which gives a ranking

$$R_5, R_4, R_1, R_2, R_6, R_3.$$

We can assume that at some point the surfer will be bored and abandon the process, so at every step there is a probability d that the process continues by following the flow and a probability $1 - d$ that the surfer jumps to another page of the net. In that case the ranking vector becomes

$$R_i = (1 - d) + d \sum_{j=1}^{n} (M^T)_{i,j} R_j.$$

If we want the rank values to add to 1 and not to $n = |V|$ we must divide $1 - d$ by n which is typically what Google uses for its algorithm

$$R_i = \frac{(1 - d)}{n} + d \sum_{j=1}^{n} (M^T)_{i,j} R_j.$$

d is called the dumping factor and its value is estimated (from the frequency that an average surfer uses her browser's bookmark feature) to be around $d = 0.85$.