Graph Theory: Basic Notions

Vincenzo Bonifaci

March 17, 2016

1 Graphs

Definition 1.1. A *directed graph* $G(V, E)$, or *digraph*, is given by a nonempty set of *nodes* V and a set of *arcs* (or *edges*) $E \subseteq V \times V$.

Notice that this allows *loops* (arcs (i, i) with $i \in V$) and that $(i, j) \neq (j, i)$. The maximum number of arcs of a directed graph is n^2 where $n = |V|$.

Definition 1.2. An (undirected) *graph* $G(V, E)$ is given by a nonempty set of *vertices* (or nodes) V and a set of *edges* $E \subseteq \binom{V}{2}$ where $\binom{V}{2} = \{\{i, j\} : i \in V, j \in V, i \neq j\}$.

Notice that this does not allow *loops* and that $\{i, j\} = \{j, i\}$. The maximum number of edges of a graph is $|\binom{V}{2}| = \binom{n}{2} = n(n-1)/2$ where $n = |V|$.

If $e = \{a, b\} \in E$ then a and b are *adjacent* or *neighbors* in G. The edge e is *incident* to a and b; a and b are the *endpoints* of e. Two different edges e,e' are *incident* if they share an endpoint.

For a directed graph, if $e = (a, b) \in E$ then a is the *tail* of e and b is the *head* of e.

Example 1.3. The *complete graph* (*clique*) K_n on n vertices has $|V| = n$ and $E = \binom{V}{2}$. It has $n(n-1)/2$ edges. The graph K_3 is also called a *triangle*.

Definition 1.4. A graph is *bipartite* if there exist $V_1, V_2 \subseteq V$ such that $V_1 \cup V_2 = V$, $V_1 \cap V_2 = \emptyset$, and no edge in E has both endpoints in V_1 or both endpoints in V_2.

Example 1.5. The *complete bipartite graph* (*bipartite clique*) $K_{p,q}$ has $V = V_1 \cup V_2$, $V_1 \cap V_2 = \emptyset$, $|V_1| = p$, $|V_2| = q$, $E = \{\{i, j\} : i \in V_1, j \in V_2\}$. It has pq edges.

The *degree* of a vertex v in G, denoted $\deg_G(v)$ or simply $\deg(v)$, is the number of edges incident to v. For a directed graph we have two quantities, the *out-degree* $\deg^+(v)$ and *in-degree* $\deg^-(v)$, which count the arcs leaving v and entering v, respectively.

Lemma 1.1 (Handshaking lemma)**.** $\sum_{v \in V} \deg(v) = 2|E|$.

Proof. In the sum, every edge is counted exactly twice. □

Very often, the letters n and m are used to denote the number of nodes and number of arcs of G, respectively. Any reasonable description of the graph should not use more than $O(n+m)$ words of memory (assuming that each node identifier fits in a single word, that is, the number of bits in a word is more than $\log_2 n$).

Many interesting graphs are *sparse*, that is, they satisfy $m \ll n^2$. So it may be possible to represent a sparse graph with much fewer than $O(n^2)$ words.

2 Subgraphs

By "removing" nodes and/or edges from a graph we obtain a *subgraph*. If we only remove nodes (and edges incident to them, but nothing else) we obtain an *induced subgraph*. If we only remove edges, we obtain a *spanning subgraph*. The formal definitions follow.

Definition 2.1. Let $G(V,E)$ be a graph. For $V' \subseteq V$, let $E|V' = \{\{a,b\} \in E : a \in V', b \in V'\}$.

- A graph of the form $G'(V', E|V')$ is an *induced subgraph* of G (it is induced by V').
- A graph of the form $G'(V', E')$ with $V' \subseteq V$, $E' \subseteq E|V'$ is a *subgraph* of G.
- A graph of the form $G'(V, E')$ with $E' \subseteq E$ is a *spanning subgraph* of G.

3 Walks, paths and cycles

A *walk* on graph $G(V,E)$ is a sequence $x_1, e_1, x_2, \ldots, x_{p-1}, e_{p-1}, x_p$ (for some $p \geq 1$) with $x_i \in V$ for all $1 \leq i \leq p$, $e_i = \{x_i, x_{i+1}\} \in E$ for all $1 \leq i \leq p-1$.

- The *endpoints* of the walk are x_1 and x_p.
- The *length* of the walk is $p-1$.
- The walk is *closed* if $x_p = x_1$.

A *path* is a walk for which the e_i are distinct and the x_i are distinct.

A *cycle* is a closed walk for which the e_i are distinct and all the x_i are distinct except for $x_p = x_1$.

The definitions can be easily extended to directed graphs by requiring that the direction of the walk is consistent with the direction of the arcs traversed by it.

A graph is *acyclic* if it contains no cycle; otherwise it is *cyclic*.

Whether a directed graph is acyclic or not can be determined in time $O(n+m)$, by running a complete *postorder* depth-first visit of the graph and marking nodes with the "time" they are visited. Say that node u gets the index t_u. If there is an arc (u,v) with $t_u < t_v$, then the graph has a cycle. Otherwise, the timestamps give a *topological order* of the nodes: a mapping $t : V \to \{1, \ldots, n\}$ such that, if $t_u < t_v$, then there is no arc from u to v. Topological orders are certificates of acyclicity and can be used to speed up algorithms for directed acyclic graphs.

Exercise 3.1. Show that if G has a topological order, then it is acyclic.

4 Connectivity and distances

Node $u \in V$ is connected to node $v \in V$ (equivalently, v is *reachable* from u) if there is a path from u to v.

A (directed) graph is *(strongly) connected* if for every $u, v \in V$, u is connected to v.

Note that in any *connected* graph, $m \geq n-1$ (why?) and so, for example, $O(n+m)$ can be shortened to $O(m)$ for connected graphs.

Whether a graph is connected or not can be determined in time $O(n+m)$, by performing a visit of the graph from an arbitrary node: if not all nodes are visited, the graph must be disconnected. For directed graphs, determining strong connectivity also costs $O(n+m)$: from an arbitrary starting node, we perform both a "forward" visit (on the original graph) and a "backward" visit of the "reverse" graph, obtained by replacing each arc (u, v) by (v, u). The original graph is strongly connected if and only if all the nodes are touched by both visits.

A *connected component* of G is an induced subgraph that is connected and *maximal*, that is, not properly contained in any connected induced subgraph of G.

The *out-component* of node u in a digraph is the set of nodes reachable from u (including u). Similarly, the *in-component* of node v is the set of nodes from which v can be reached (including v).

The strongly connected component of node u is the intersection of the out-component of u and the in-component of u (why?).

The connected components of an undirected graph can be determined in linear time by an exhaustive visit of the graph. For directed graphs, there is also a linear time algorithm to determine the strongly connected components, but it is slightly more complicated and requires two distinct visits (Kosaraju's algorithm).

Kosaraju's algorithm for finding strong components in digraphs

1. Given G, construct its reverse graph G^R (each arc (u, v) in G becomes (v, u) in G^R).
2. Construct the node ordering σ given by a *reverse post-order* depth-first search of G^R.
3. Use the ordering σ to perform a complete depth-first search of G. Each set of nodes visited in an outer DFS call is a strongly connected component of G.

Example 4.1. Consider the graph in the right part of the following figure.

The reversed graph is depicted on the left. The post-order sequence of nodes, after being reversed, is 1 0 2 3 4 11 9 12 10 6 7 8 5. The third step of the algorithm yields the strongly connected components: $\{1\}$, $\{0, 5, 4, 3, 2\}$, $\{11, 12, 9, 10\}$, $\{6\}$, $\{7, 8\}$.

A *tree* is an undirected graph that is connected and acyclic. A tree on n nodes has $n-1$ edges.

A *forest* is an undirected graph that is acyclic. Each connected component of a forest is a tree.

If u is connected to v, a *shortest path from u to v* is a path from u to v of minimum length.

The distance $d(u,v)$ from u to v is the length of a shortest path from u to v.

The distance from u to all other nodes of the graph can be computed in linear time with a single breadth-first search from u.

The *diameter* of a graph is

$$D = \max_{u,v \in V:\, u \text{ is connected to } v} d(u,v).$$

Determining exactly the diameter of a graph can be costly. After ensuring that the graph is connected, we can, by using n breadth-first searches (one from every node) determine all distances between pairs of nodes (and therefore, the diameter) in $O(mn)$ time. However, if we only need a rough estimate of the diameter, we can run a breadth-first search visit from an arbitrary node u: if the highest distance from u to any other node is B, then $B \leq D \leq 2B$. Therefore, approximating the diameter within a factor of two costs $O(n+m)$ time.

Exercise 4.1. Prove that in the special case when G is a tree, the diameter can be computed in linear time.

Exercise 4.2 (Open research problem!)**.** Find an efficient algorithm that computes a value B such that $B \leq D \leq 1.1B$. The algorithm should be asymptotically faster than $O(mn)$.

5 Graph matrices

The *adjacency matrix* of a directed graph $G(V,E)$ is a matrix $A \in \mathbb{R}^{n \times n}$ ($n = |V|$) defined by

$$A_{ij} = \begin{cases} 1 & \text{if } (i,j) \in E \\ 0 & \text{otherwise.} \end{cases}$$

A similar definition holds for (undirected) graphs; in that case, A is symmetric and has an all-zero diagonal.

The *incidence matrix of an undirected graph* $G(V,E)$ is a matrix $B \in \mathbb{R}^{n \times m}$ ($n = |V|$, $m = |E|$) defined by

$$B_{ve} = \begin{cases} 1 & \text{if } e \text{ is incident to } v \\ 0 & \text{if } e \text{ is not incident to } v. \end{cases}$$

The *incidence matrix of a directed graph* $G(V,E)$ is a matrix $B \in \mathbb{R}^{n \times m}$ ($n = |V|$, $m = |E|$) defined by

$$B_{ve} = \begin{cases} +1 & \text{if } v \text{ is the tail of } e \\ -1 & \text{if } v \text{ is the head of } e \\ 0 & \text{if otherwise.} \end{cases}$$

Recall that the *eigenvalues* of a matrix M are the values $\lambda \in \mathbb{C}$ such that $Mx = \lambda x$ for some vector $x \in \mathbb{C}^n$ (x is called an *eigenvector* associated to λ).

Every real matrix A can be written in the form $A = QTQ^\top$, where Q is an *orthogonal* matrix (which means that both $QQ^\top$ and $Q^\top Q$ equal the identity matrix) and T is an *upper triangular matrix* (which means that all entries below T's main diagonal are zero). This is called the *Schur decomposition* of A.

If A is real and symmetric, then one can write $A = QDQ^\top$ where Q is orthogonal and D is a *diagonal* matrix (which means that all entries outside the main diagonal are zero).

The adjacency matrix A of a directed graph has the following property: the eigenvalues of A are all 0 if and only if A represents an acyclic digraph.

Exercise 5.1. Prove that if A is the adjacency matrix of a directed acyclic graph, then all eigenvalues of A are 0.

What is the number $N_{ij}^{(r)}$ of walks of given length r between two nodes i and j in a given graph or directed graph? If $r = 1$, the answer is simply A_{ij}. If $r = 2$, the answer is

$$N_{ij}^{(2)} = \sum_{k=1}^{n} A_{ik}A_{kj} = [A^2]_{ij}$$

where $[A^2]_{ij}$ denotes the ijth element of the matrix A^2. Similarly, $N_{ij}^{(3)} = [A^3]_{ij}$, and one can show by induction that

$$N_{ij}^{(r)} = [A^r]_{ij}.$$

A special case is when $i = j$, so we are counting the number of closed walks of length r that start and end in i. The total number L_r of closed walks of length r in the network is the sum of this quantity over all possible starting points i:

$$L_r = \sum_{i=1}^{n} [A^r]_{ii} = \operatorname{tr} A^r,$$

where $\operatorname{tr} M$ is the *trace* of M (sum of the elements on M's main diagonal).

By using the Schur decomposition of A, we can show that $L_r = \sum_i \kappa_i^r$, where each κ_i is an eigenvalue of A. Indeed, notice that if x is an eigenvector of A with eigenvalue κ, then $QTQ^\top x = Ax = \kappa x$, and multiplying by $Q^\top$ both sides we get $(Q^\top Q)TQ^\top x = \kappa Q^\top x$, that is, $TQ^\top x = \kappa Q^\top x$, so $Q^\top x$ is an eigenvector of T with the same eigenvalue κ). So (using $\operatorname{tr}(XY) = \operatorname{tr}(YX)$ for square matrices X,Y),

$$L_r = \operatorname{tr} A^r = \operatorname{tr}(QT^rQ^\top) = \operatorname{tr}(Q^\top QT^r) = \operatorname{tr} T^r = \sum_i \kappa_i^r.$$

Exercise 5.2. Prove that $\operatorname{tr}(XY) = \operatorname{tr}(YX)$ for any pair of matrices $X \in \mathbb{R}^{m\times n}$, $Y \in \mathbb{R}^{n\times m}$.

Exercise 5.3. Using the above formula for L_r, show that if A is the adjacency matrix of a directed graph and all of the eigenvalues of A are 0, then the digraph is acyclic.

6 Planar graphs

Informally speaking, a graph is *planar* if it can be drawn on the plane without having any edges cross.

An example of non-planar graph is K_5, the clique on 5 vertices.

Kuratowski's theorem characterizes what graphs are planar. An *expansion* of a graph is a graph obtained by by adding extra vertices in the middle of edges (for example, replacing an edge $\{u, v\}$ by the 3 edges $\{u, x\}$, $\{x, y\}$, $\{y, v\}$ where x and y are new nodes).

Theorem 6.1 (Kuratowski)**.** *Every non-planar graph contains at least one subgraph that is an expansion of K_5 or an expansion of $K_{3,3}$.*

A coloring of a graph is an assignment of "colors" (mathematically speaking: integers) to nodes so that pairs of adjacent nodes receive distinct colors. The following is one of the most famous theorems in discrete mathematics.

Theorem 6.2 (Four-Color Theorem)**.** *Every planar graph admits a coloring using at most 4 colors.*

A *face* of a planar graph is one of the connected regions of the plane determined by a planar drawing of the graph. The external region is also considered a face (called the *outer* face). Let n, m, f be the number of vertices, edges and faces of a graph, respectively. The following relation can be shown by induction.

Theorem 6.3 (Euler's Formula)**.** *For any connected planar graph, $n - m + f = 2$.*

Whether a graph is planar can be decided in linear time, but the algorithm is beyond our scope.

Exercise 6.1. Using Euler's Formula, show that the average degree of a connected planar graph is strictly less than 6 (and therefore, any planar graph has a vertex with degree at most 5).

7 Eulerian and Hamiltonian paths and cycles

A *Eulerian path* is a walk that traverses each edge in a network exactly once. A *Eulerian cycle* is a closed walk that traverses each edge in a network exactly once.

A *Hamiltonian path* is a walk that traverses each node in a network exactly once. A *Hamiltonian cycle* is a closed walk that traverses each node in a network once.

A connected undirected graph has a Eulerian cycle if and only if every node has even degree. A strongly connected directed graph has a Eulerian cycle if and only if every node has in-degree equal to its out-degree. Therefore, in both cases detecting the existence of a Eulerian cycle is very simple.

Detecting the existence of a Hamiltonian cycle in a graph (or digraph) is much more difficult – it is, in fact, an NP-complete problem.

You can try out the Android game *uConnect Free*, which requires the player to find Eulerian paths of graphs.

8 Independent paths, cut sets

Two paths between u and v are node-independent (or *node-disjoint*) if they do not share any node except for u and v.

Two paths between u and v are edge-independent (or *edge-disjoint*) if they do not share any edge. If two paths between u and v are node-independent, they are also edge-independent, but the reverse is not true.

The number of independent paths between two nodes is called the *connectivity* of the two vertices. It can be thought as a measure of how strongly connected the two nodes are.

A *node cut set* for u and v is a set of nodes whose removal disconnects u from v. Similarly, an *edge cut set* for u and v is a set of edges whose removal disconnects u from v. A *minimum cut set* for u and v is the smallest cut set that disconnects u and v. A minimum cut set need not be unique.

Theorem 8.1 (Menger's Theorem)**.** *If there is no cut set of size less than k between a given pair of nodes, then there are at least k independent paths between the same nodes.*

Note that the theorem is true for both situations: 1) comparing edge cut sets to edge indepedent paths, and 2) comparing node cut sets to node independent paths.

Therefore, the size of the minimum node (respectively, edge) cut set that disconnects a given pair of nodes in a network is equal to the node (respectively, edge) connectivity of the same nodes.

Menger's Theorem is closely related to the maximum flow / minimum cut theorem, which applies to networks where the edges have weights (often also called *capacities*). In this case, a minimum edge cut set is an edge cut set with the smallest sum of weights that separates a given pair of nodes.

Theorem 8.2 (Max-flow/min-cut Theorem)**.** *The maximum flow between a given pair of nodes in a network is equal to the sum of the weights on the edges of the minimum edge cut set that separates the same two nodes.*

A maximum flow in a weighted graph (with rational weights) can be computed relatively efficiently. For example, in time $O(m^2n)$ with the Edmonds-Karp algorithm, or in time $O(mF)$, where F is the maximum flow value, with the Ford-Fulkerson algorithm.

The maximum flow algorithms can, in particular, be used to determine the edge connectivity between any two vertices s and t: apply the algorithm to a network where the capacity of every edge is 1; the value of the maximum flow between s and t will be the value of the edge connectivity between s and t.

To determine vertex connectivity, we can use the following reduction to the edge connectivity problem. Assume that the graph is directed (if it is not, first replace each edge by two opposite directed edges). Then replace every node with two nodes separated by a directed edge; all original incoming edges connect to the first of these two, and all outgoing edges to the second. The value of the edge connectivity between s and t in the new digraph is equal to the value of the node connectivity in the original graph.

Exercise 8.1. Prove that the edge and vertex connectivity between two nodes of a simple connected graph (that is, without repeated edges) can both be computed in time $O(mn)$. Hint: use the Ford-Fulkerson algorithm.

9 The graph Laplacian

Given an undirected graph G, consider the following matrix:

$$D = \begin{pmatrix} \deg(1) & 0 & 0 & \cdots \\ 0 & \deg(2) & 0 & \cdots \\ 0 & 0 & \deg(3) & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

Then if A is the adjacency matrix of G, the matrix $L = D - A$ is called the Laplacian of G and plays an important role. Note that we have

$$L_{ij} = \begin{cases} \deg(i) & \text{if } i = j \\ -1 & \text{if } i \neq j \text{ and } \{i, j\} \in E \\ 0 & \text{otherwise.} \end{cases}$$

If we orient arbitrarily the edges of G, and let B be the signed incidence matrix of the resulting digraph, we also have

$$L = BB^\top.$$

This implies that L can only have real non-negative eigenvalues: if v is any eigenvector of L, say with eigenvalue λ, and v is normalized so that $v^\top v = 1$, then

$$\lambda = \lambda v^\top v = v^\top L v = (v^\top B)(B^\top v) = (B^\top v)^\top (B^\top v) \geq 0$$

and so we have shown that all eigenvalues of L are nonnegative.

The eigenvalues of the Laplacian of a graph are conventionally indexed from the smallest to the largest: $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$. The eigenvalues contain useful information about the graph. For example, if a graph has exactly c connected components, then $\lambda_1 = \lambda_2 = \ldots = \lambda_c = 0$, while $\lambda_{c+1} > 0$. Note that since a graph has always at least one connected component, λ_1 is always 0. In fact, the vector $(1, 1, \ldots, 1)$ is always an eigenvector of L with eigenvalue 0 (why?).

When the graph is connected, $\lambda_2 > 0$. In fact, λ_2 has special importance and is called the *algebraic connectivity* of the graph.

We discuss two settings where the Laplacian is useful: random walks and resistor networks.

9.1 Random walks

A *random walk* is a discrete process on a network in which, starting at some specified initial node, at each step we choose uniformly at random an edge to traverse among those attached to the current node, move along the edge, and repeat. Let $p_i(t)$ be the probability that the walk is at node i at time t, then we have

$$p_j(t) = \sum_{i=1}^{n} \frac{A_{ij}}{\deg(i)} p_i(t-1),$$

where A is the adjacency matrix of the graph and $\deg(j)$ is the degree of node j. In matrix notation, $p(t)^\top = p(t-1)^\top D^{-1} A$, where p is the probability distribution vector of the random walk, and D is the diagonal matrix with the degrees of the nodes on its diagonal.

Under suitable hypothesis, the random walk process converges. Assume that the random walk process converges, that is, the limit of $p_i(t)$ as $t \to \infty$ exists for all i. Then the limit probability distribution (called the *stationary distribution*) has to satisfy:

$$p^\top = p^\top D^{-1} A.$$

Rearranging, this is equivalent to

$$p^\top (I - D^{-1}A) = p^\top D^{-1}(D - A) = p^\top D^{-1} L = 0.$$

On a connected *undirected* network, we know that L has only a single eigenvalue with value 0 and we have argued before that the constant vector $\mathbf{1}$ is an eigenvector associated to eigenvalue 0. But notice that $p^\top D^{-1} L = L D^{-1} p = 0$ implies that p is also an eigenvector of L with eigenvalue 0. Therefore $D^{-1}p$ and $\mathbf{1}$ have to be parallel vectors (because $\lambda_2 > 0$, and so the space of eigenvectors associated to eigenvalue 0 has dimension 1), meaning that there is a constant c such that $p_i = c \cdot \deg(i)$ for all i. In other words, in the limit, the probability distribution p is proportional to the degree of each node.

9.2 Resistor networks

There are connections between random walks on networks and the calculation of current flows in networks of resistances. Consider a graph in which edges are identical resistors of resistance R and the vertices are junctions between the resistors, and suppose we apply a voltage between two vertices s and t such that a current I_0 flows from s to t in the network. What is the current flow through any given resistor in the network?

Let V_i be the voltage at vertex i, measured relative to any reference potential. *Kirchhoff's current law* says that electricity is conserved:

$$\sum_{j=1}^{n} A_{ij} \frac{V_j - V_i}{R} + I_i = 0,$$

where I_i represent the external current (if any) injected into vertex i, in particular:

$$I_i = \begin{cases} +I_0 & \text{for } i = s, \\ -I_0 & \text{for } i = t, \\ 0 & \text{otherwise.} \end{cases}$$

We now observe that $\sum_j A_{ij} = \deg(i)$, so Kirchhoff's law is equivalent to $\deg(i)V_i - \sum_j A_{ij}V_j = RI_i$, or

$$\sum_j (\delta_{ij} \deg(i) - A_{ij})V_j = RI_i,$$

which in matrix form is (recall that $L = D - A$ is the graph Laplacian)

$$LV = RI.$$

Since $\mathbf{1}$ is an eigenvector of L, with eigenvalue 0, we indeed verify that adding the same value c to all the voltages yields a new valid set of voltages:

$$L(V + c\mathbf{1}) = LV + L\mathbf{1} = LV = RI.$$

The Laplacian matrix L is not invertible (since 0 is always an eigenvalue of L), but there is a unique matrix L^+ such that $LL^+ = L^+L = I$ and such a matrix is called a pseudoinverse of L. If the pseudoinverse L^+ has been computed, we can obtain the voltages at every vertex by the formula

$$V = L^+RI.$$

In practice, we can assume that the voltage V_t of the sink vertex equals 0 and remove, from the system $LV = RI$, the row and column of L corresponding to t and the row of V corresponding to t, without affecting the results. We get a new system $L'V' = RI'$, where

- L' is the Laplacian without row and column t;
- V' is the voltage vector without row t;
- I' is the external current vector without row t.

The reduced matrix L' has full rank and therefore it can be inverted to obtain $V' = R(L')^{-1}I'$. Once we have the voltages, we can compute easily other quantities of interest, such as the current flow along any given edge.

Solving an $n \times n$ linear system costs $O(n^3)$ arithmetic operations if we use Gaussian elimination. However, to solve approximately (within high accuracy) a Laplacian linear system of the form $Lx = b$, it is not necessary to use a Gaussian elimination routine. There are faster algorithms that can find x in time $O(m(\log n)^3 \log(1/\epsilon))$, where m and n are the number of edges and nodes of the graph and $\epsilon > 0$ is the relative error. The algorithm guarantees that $\|x - x^*\|_L \leq \epsilon \|x^*\|_L$, where x is the solution found by the algorithm, x^* is the exact solution of $Lx = b$, and $\|z\|_L = z^\top Lz$.

Example 9.1. Consider a triangle graph (K_3) where every edge is a resistor of value $R = 1$. We inject a unit current in node 1, and extract a unit of current from node 2. Let's determine the voltages of nodes 1, 2 and 3.

The Laplacian of K_3 is

$$L = \begin{pmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{pmatrix}.$$

The external current vector I is

$$I = \begin{pmatrix} +1 \\ -1 \\ 0 \end{pmatrix}.$$

The sink is $t = 2$. Therefore we get the equations $V_2 = 0$ and

$$\begin{aligned} 2V_1 - V_3 &= 1 \\ -V_1 + 2V_3 &= 0. \end{aligned}$$

The solution is $V_1 = 2/3$, $V_2 = 0$, $V_3 = 1/3$. The current on edge $(1, 2)$ is $2/3$, the current on edges $(1, 3)$ and $(3, 2)$ is $1/3$.