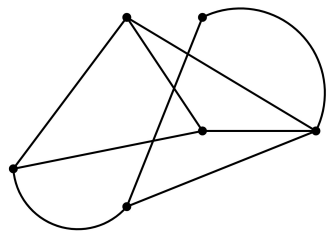
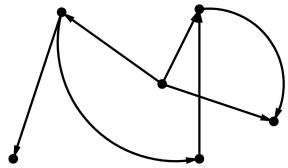
4.3 Graphs

We have successfully answered the problems of consistency, uniqueness and reconstruction of discrete tomography in Section 4.1 in the special case, when the directions of X-rays are parallel to the coordinate axises of a Cartesian coordinate system. Now we would like to introduce a method to solve these problems in the case of arbitrary two lattice directions, which don't need to be parallel to the coordinate directions, and hence don't need to be parallel to the sides of the picture region. The solution method involves the knowledge of maximal flows in networks, which are important concepts in graph theory. Thus we give a brief overview of graph theory in this section and network flows in the next section with all the necessary tools and definitions.

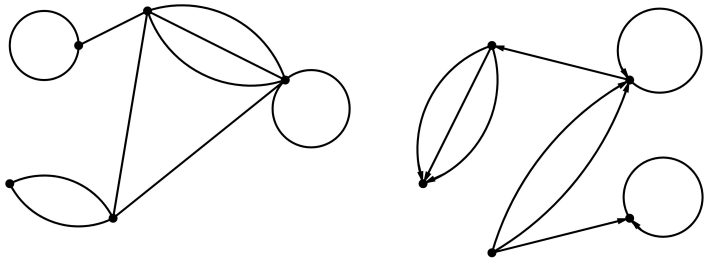
A **simple graph** consists of a (non-empty) finite set V of elements called **vertices** (or **nodes**), a finite set E of elements called **edges**, and a function f which assigns a subset of two elements of V to every element of E, such that if $e_1 \neq e_2$, then $f(e_1) \neq f(e_2)$. If $f(e) = \{u, v\}$ for an $e \in E$ and some $u, v \in V$ then we say the edge **connects** the vertices u and v. The requirement that $f(e_1) \neq f(e_2)$, whenever $e_1 \neq e_2$, means that two distinct edges can't connect the same pair of vertices. When the edge connects the vertices u and v, then u and v are called **adjacent**, and we say u is a **neighbor** of v, and v is a neighbor of u. We may also write $uv \in E$ and say uv is an edge of the graph, if there exists $e \in E$ such that $f(e) = uv$. We note that if uv is an edge, then vu is also an edge and uv is the same edge as vu. A graph can be visualized in the plane by presenting the vertices as points of the plane and presenting the edges as curves or line segments connecting the vertices. It doesn't matter how we arrange the points, or how the shape of the curves presenting the edges looks like, the only important thing is to define which vertices are connected by which edges. The edges may cross each other in points which are not vertices of the graph.



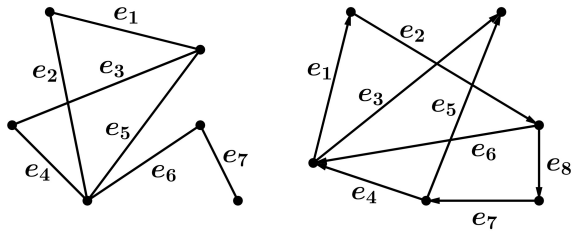
A **simple directed graph** consists of a (non-empty) finite set V of elements called **vertices** (or **nodes**), a finite set E of elements called **edges**, and a function f which assigns an ordered pair of vertices to every element of E, such that if $e_1 \neq e_2$, then $f(e_1) \neq f(e_2)$. We say the vertex u is **connected to** v if there exists $e \in E$, such that $f(e) = (u, v)$. In this case we may write $uv \in E$ and say uv is and edge of the directed graph, and v is called an **out-neighbor** of u, while u is called an **in-neighbor** of v. If $f(e) = (u, v)$, then u is called the **initial vertex** and v is celled the **terminal vertex** of the directed edge e. Note that if uv is an edge of a directed graph it may happen that vu is not an edge. A directed graph can be visualized in the plane by presenting the vertices as points of the plane and presenting the edges as directed curves or directed line segments connecting the vertices.



Simple graphs and simple directed graphs are presented above, which concepts help us to state most graph theoretical theorems in a simple form without the need to care about degenerate situations, while still capable to investigate many interesting problems. However graphs and directed graphs can be defined in a more general context if we allow **parallel edges** and **loops**. Two distinct edges of an undirected graph are called parallel if they connect the same pair of vertices, while two directed edges of a directed graph are called parallel if they share the initial vertex and the terminal vertex. A loop is an edge which connects a vertex with itself, or if initial point and terminal point coincide in the case of a directed graph. Note that if there are parallel edges in a graph, then we can't refer to the edges by a (possibly ordered) pair of vertices, as there can be more than one edges connecting the same vertices. Below there's an example of a graph and a directed graph with parallel edges and loops.

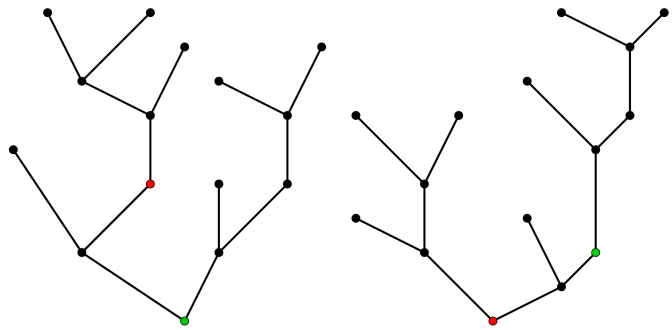


A **walk** in a graph is a finite sequence of edges $e_1, e_2, \ldots, e_m$ such that e_{i-1} and e_i have a common vertex for all $i \in \{2, 3, \ldots, m\}$. A **(directed) walk** in a directed graph is a finite sequence of edges $e_1, e_2, \ldots, e_m$ such that the terminal vertex of e_{i-1} is the initial vertex of e_i for all $i \in \{2, 3, \ldots, m\}$. Any walk determines the sequence of vertices $v_0, v_1, \ldots, v_m$, where e_i connects the vertices v_{i-1} and v_i (or v_{i-1} is connected to v_i by the edge e_i in the case of a directed graph). A walk in which all the edges are distinct is called a **trail**. If, in addition, all the vertices $v_0, v_1, \ldots, v_m$ determined by the path/trail are distinct (except possibly $v_0 = v_m$), then it's called a **path**. A path or trail is **closed** if $v_0 = v_m$, and a closed path is called a **cycle**. Below we present a graph on the left and a directed graph on the right. In the graph on the left, $e_1, e_2, e_4, e_3, e_1, e_2, e_6$ is a walk, but not a trail, $e_1, e_2, e_4, e_3, e_5, e_6$ is a trail, but not a path, and e_1, e_3, e_4, e_6 is a path. In the directed graph on the right, $e_4, e_1, e_2, e_8, e_7, e_4, e_3$ is a directed walk, but not a trail, e_4, e_1, e_2, e_6, e_3 is a directed trail, but not a path, and e_1, e_2, e_8, e_7, e_5 is a directed path.



We say that a graph is **connected** if there exists a path between any pair of vertices. A connected graph which contains no cycle is called a **tree**. Tree are often presented by choosing a vertex, which is called the root, and then

positioning the rest of the vertices above the root. Below you can see two graphs, which present the same tree, but choosing different vertices as roots.

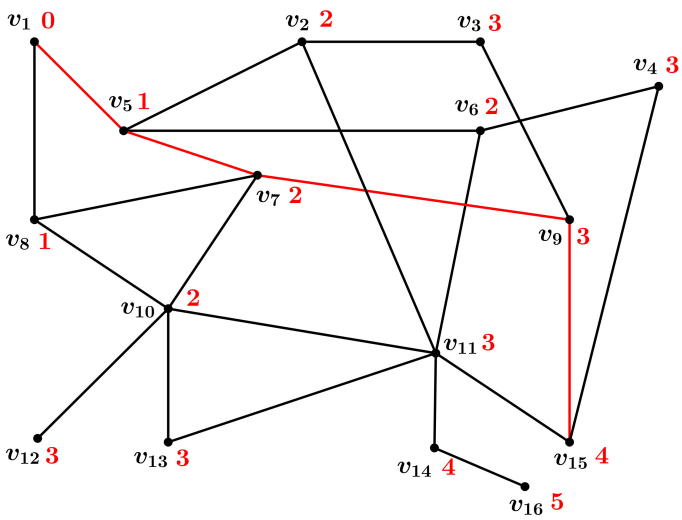


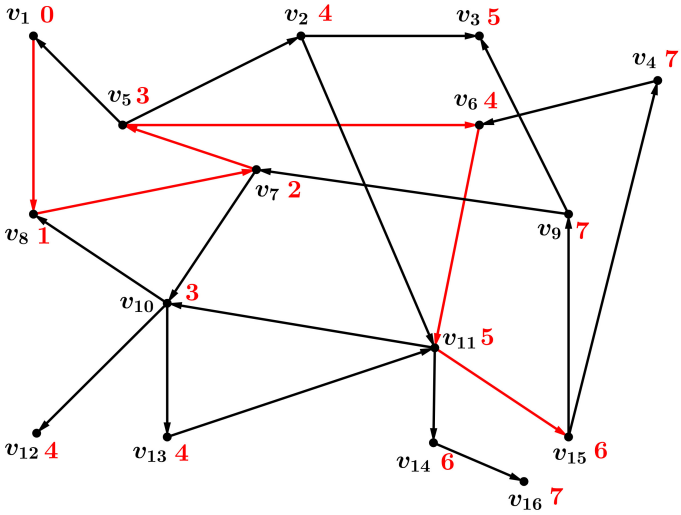
The **length** of a path or a cycle is the number of edges in it. The **distance of the vertices** s and t in a connected graph is the length the shortest path connecting s and t. Given a vertex of $s \in V$ of any (directed) graph there's an easy method to find the distance of any further vertex form s, which can be connected to s with a path. It's based on assigning labels to the veritces that give the distances from s.

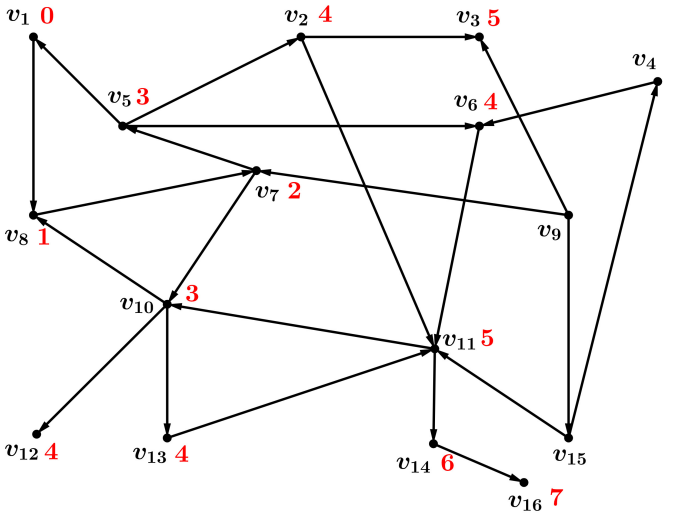
1. Assign the label zero to the vertex s.
2. Assuming that the highest label of the vertices of the graph is $k \in \mathbb{Z}$, look for the unlabeled neighbors (or out-neighbors) of the vertices with label k. If there's at least one such neighbor (or out-neighbor) then assign the label $k + 1$ to them.
3. Repeat step 2 as long as possible.
4. If the vertices with the highest label have no unlabeled neighbors (or out-neighbors) then we found all the distances of the vertices of the graph which can be connected to s with a path. If there are still unlabeled vetrices, then those cannot be accessed from s via any path.

The labels assigned by the above procedure give the distances form s. It's also easy to find a shortest path connecting s to a vetrex t with the help of these distances. If t has no label, then it cannot be accessed from s. Otherwise let the label of t be $k \in \mathbb{Z}$. Then t must have a neighbor (or in-neighbor) denoted by v_{k-1}, which has label $k-1$. The vertex v_{k-1} must have a neighbor (or in-neighbor) denoted by v_{k-2}, which has label $k-2$. This can be continued until a vertex v_1 with label 1 is found. Then s must be a neighbor (or in-neighbor) of v_1, and hence the vertices $s, v_1, v_2, \ldots, v_{k-1}, t$ determine a shortest path form s to t.

We show an example of a graph with labels that present the distances from the vertex v_1 (red numbers), and a shortest path connecting v_1 and v_{15}. Two further examples show the distances in two directed graphs from the vertex v_1 (red numbers). In the first example we also present a shortest directed path connecting v_1 and v_{15}, while there's no path from v_1 to v_{15} in the second example.




v_1 0
v_2 4
v_3 5
v_4 7
v_5 3
v_6 4
v_7 2
v_8 1
v_9 7
v_{10} 3
v_{11} 5
v_{12} 4
v_{13} 4
v_{14} 6
v_{15} 6
v_{16} 7


v_1 0
v_2 4
v_3 5
v_4
v_5 3
v_6 4
v_7 2
v_8 1
v_9
v_{10} 3
v_{11} 5
v_{12} 4
v_{13} 4
v_{14} 6
v_{15}
v_{16} 7

4.4 Networks and flows

A **network** is a directed graph, where non-negative real numbers are assigned to the directed edges, which are called **capacities**. The capacity of the edge e is denoted by $U(e)$, where $U\colon E \to \mathbb{R}$ is the **capacity function**. A **pseudo-flow** is a function $Y\colon E \to \mathbb{R}$ with non-negative values, such that $Y(e) \leq U(e)$ for all $e \in E$. The real number $Y(e)$ is called the value of the flow on the edge e, and the only condition what a pseudo-flow must satisfy is that the value of the flow on any edge e can't be larger than the capacity of e. The sum of the flow values on all edges uv, where u is an in-neighbor of v, is called the **inflow** at the vertex v, while the sum of the flow values on all edges vu, where u is an out-neighbor of v, is called the **outflow** at the vertex v. Given two specified vetrices s and t, which are called **source** and **sink** respectively, a pseudo-flow is called **flow** is it satisfies the following conditions:

1. the inflow of the source s is zero,
2. the outflow of the sink t is zero,
3. the inflow and outflow equal to each other for all vertices, except (possibly) for s and t.

The third condition is called the flow conservation property. The **size of a flow** is the outflow of the source s, which equals to the inflow of the sink t due to the flow conservation property at rest of the vertices. Given a network and flow, an edge e is called **saturated** if $Y(e) = U(e)$, otherwise it's called **unsaturated**. A flow is called **maximal** if there's no other flow on the same network, whose size is larger.

Example

Let G be the simple directed graph with vertex set $V = \{v_1, v_2, v_3, v_4, s, t\}$, where all the directed edges are v_1v_2, v_1v_3, v_4v_2, v_4v_3, sv_1, sv_4, v_2t, v_3t, see Figure 4.2. Note that we refer to the directed edges with ordered pairs of vertices, because G is simple. Let the capacities be

$$\begin{array}{rrrr} U(v_1v_2) = 3 & U(v_1v_3) = 2 & U(v_4v_2) = 2 & U(v_4v_3) = 4 \\ U(sv_1) = 4 & U(sv_4) = 5 & U(v_2t) = 3 & U(v_3t) = 6 \end{array}$$

The directed graph G together with the capacity function U provides a network. The capacities are denoted by black numbers in brackets next to the

corresponding edges in Figure 4.2. We specify two flows Y and $\overline{Y}$ on this network as

$$\begin{array}{rrrr} Y(v_1v_2)=3 & Y(v_1v_3)=2 & Y(v_4v_2)=2 & Y(v_4v_3)=4 \\ Y(sv_1)=4 & Y(sv_4)=5 & Y(v_2t)=3 & Y(v_3t)=6 \end{array}$$

and

$$\begin{array}{rrrr} \overline{Y}(v_1v_2)=3 & \overline{Y}(v_1v_3)=2 & \overline{Y}(v_4v_2)=2 & \overline{Y}(v_4v_3)=4 \\ \overline{Y}(sv_1)=4 & \overline{Y}(sv_4)=5 & \overline{Y}(v_2t)=3 & \overline{Y}(v_3t)=6 \end{array}$$

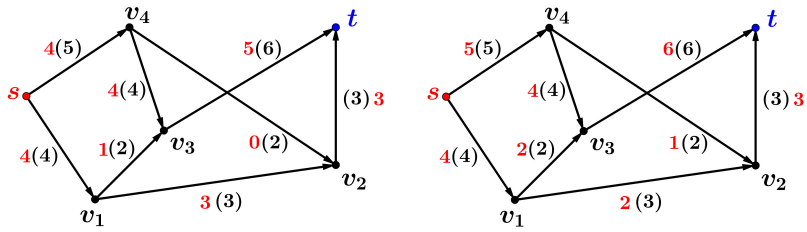


Figure 4.2

The flow values of Y are presented by red numbers in Figure 4.2 on the left, and the flow values of $\overline{Y}$ are presented by red numbers in the same figure on the right. The edges v_1v_2, v_4v_3, sv_1, v_2t are saturated in the flow Y, while the rest of the edges are unsaturated. The edges v_1v_3, v_4v_3, sv_1, sv_4, v_2t, v_3t are saturated in the flow $\overline{Y}$, while the rest of the edges are unsaturated. The size of the flow Y is 8, and the size of the flow $\overline{Y}$ is 9. This immediately shows that the flow Y is not maximal, and it's easy to see that $\overline{Y}$ is a maximal flow as the sum of the capacities of the edges with initial vertex s equals to 9, which can't be exceeded by the size of any flow.

An interesting question is how can we decide whether a given flow is maximal or not, and how can we find a maximal flow. This problem seem to be quite challenging for large networks at the first sight, however an easy method exists to solve it with the help of flow-augmenting paths. Given a directed graph G, an **undirected walk** is a finite sequence of edges $e_1, e_2, \ldots, e_m$ such that e_{i-1} and e_i have a common vertex for all $i \in \{2, 3, \ldots, m\}$. Note that the difference between an undirected walk and a directed walk is that a directed walk must also satisfy that the terminal vertex of e_{i-1} is the initial vertex of e_i for all $i \in \{2, 3, \ldots, m\}$. An undirected trail and undirected path can

be defined similarly. Any undirected path determines a sequence of vertices $v_0, v_1, \ldots, v_m$, where e_i connects the vertices v_{i-1} and v_i. Then we say that the path connects the vertex v_0 to the vertex v_m. We can assign directions to the edges e_i of the undirected path by defining v_{i-1} as the initial point and v_i as the terminal point of the edge e_i for all $i \in \{1, 2, \ldots, m\}$. This (possibly new) direction of the edge e_i is called the **associated direction** determined by the path $e_1, e_2, \ldots, e_m$. An edge e_i of an undirected path in a directed graph is called **forward edge** if the direction of e_i in the directed graph is the same as the associated direction determined by the path. If e_i is not a forward edge of an undirected path in a directed graph, then it's called a **backward edge**.

Given a network with capacity function U, and given a flow Y, a **flow-augmenting path** is an undirected path connecting the source s to the sink t, which satisfies that $Y(e) < U(e)$ for all forward edges of the path, and $0 < Y(e)$ for all backward edges of the path. Note that if there's a flow-augmenting path for a flow on a network, then we can increase the value of the flow on forward edges and decrease the value of the flow on the backward edges by the same positive number without violating the flow preserving property. If we also don't want to violate non-negativity and the pseudo-flow property (which requires $Y(e) \leq U(e)$ for all edges), then we can decrease the values of the flow on backward edges only by the minimum of the values $Y(e)$ for all the backward edges of the flow-augmenting path, and we can increase the values of the flow on forward edges only by the minimum of the differences $U(e) - Y(e)$ for all the forward edges of the flow-augmenting path. Thus let the minimum of these two be called the **value of the flow-augmenting path**. Note also that the very first edge and the very last edge of any flow-augmenting path must be a forward edge, since the inflow of the source is zero, just as the outflow of the sink is zero. Thus if there exists a flow-augmenting path for a flow on a network, then increasing the values of the flow on the forward edges and decreasing the values of the flow on the backward edges by the value of the flow-augmenting path will result a valid flow with larger size on the same network. This shows that if there's a flow-augmenting path, then the given flow is not maximal. A little more interesting fact is that the only reason of not being able to find a flow-augmenting path is that the flow is maximal. This is summarized in the following theorem.

Theorem 13 *A flow on a network is maximal if and only if there exists no*

flow-augmenting path for the flow.

In the example above, presented by Figure 4.3, $sv_4, v_4v_2, v_2v_1, v_1v_3, v_3t$ is a flow-augmenting path for the the flow Y. This is highlighted by red in Figure 4.3 on the left. Note that all these edges are forward edges except v_2v_1. The value of the flow on this backward edge is 3. The differences of the capacities and flow values on the forward edges $sv_4, v_4v_2, v_1v_3, v_3t$ are $1, 2, 1, 1$ respectively. The minimum of these values is 1, and then the minimum of 3 and 1 equals to 1. Thus the value of the flow-augmenting path is 1. If we increase the values of the flow by 1 on the forward edges $sv_4, v_4v_2, v_1v_3, v_3t$, and decrease the value of the flow by 1 on the backward edge v_2v_1, then we get the valid flow $\overline{Y}$ (see Figure 4.2 on the right). There's no flow-augmenting path for the flow $\overline{Y}$, since all edges with initial point s are saturated, and hence we can't increase the value of the flow on any of them.

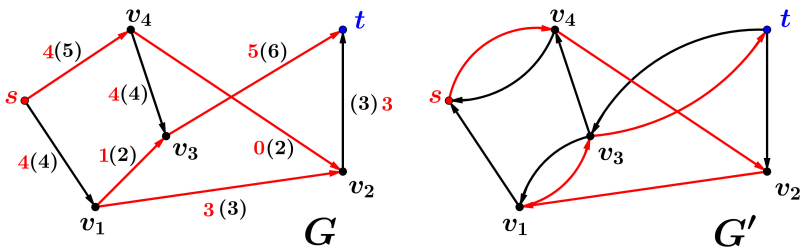


Figure 4.3

Now the only questions we need to answer is how to find a flow-augmenting path and how to know if there's no such path. This can be answered easily with the help of the **associated graph**. Let G be a network with capacity function U and let Y be a flow on this network. The associated graph is the directed graph G' with the same vertices as G and directed edges are chosen in the following way:

1. If e is a directed edge of G with zero capacity then it's deleted from G.
2. If e is a directed edge of G with nonzero capacity and zero flow value, then e is preserved as a directed edge of G'.
3. If e is an unsaturated directed edge of G with a positive flow value,

connecting vertex v_i to the vertex v_j, then preserving e as a directed edge of G', we add a new directed edge e', which connects v_j to v_i.

4. If e is a saturated directed edge of G with a positive flow value, connecting vertex v_i to the vertex v_j, then e is deleted form G and replaced by an directed edge e', which connects v_j to v_i.

The associated graph G', corresponding to the network G and flow Y presented on the left of Figure 4.3, is shown in the same figure on the right. There's no edge in G with zero capacity. The directed edge v_4v_2 has a nonzero capacity, but the value of the flow is zero on it, thus it remains a directed edge in G'. The directed edges v_1v_3, sv_4 and v_3t in G are unsaturated with positive flow values, thus these remain edges of G', but edges of the opposite directions are also added, which are v_3v_1, v_4s, tv_3. The directed edges v_1v_2, v_4v_3, sv_1, v_2t in G are saturated with positive flow values, thus these edges are deleted and replaced by edges of the opposite directions, which are v_2v_1, v_3v_4, v_1s, tv_2. We discussed in the previous section how to find the shortest directed path connecting the source s to the sink t in the associated graph G'. The shortest directed path $sv_4, v_4v_2, v_2v_1, v_1v_3, v_3t$ is highlighted by red in Figure 4.3 on the right. Note that we can tell exactly that which edge of G corresponds to each edge of the directed path in G'. Hence we have a corresponding undirected path in G. If the path in G' contains an edge which is not an edge of G, then it defines a backward edge in the corresponding undirected path, while the rest of the edges define forward edges of the corresponding undirected path. The way we define the associated graph ensures that undirected path in G, corresponding to any directed path in G' connecting s to t, is a flow-augmenting path. The flow-augmenting path corresponding to the directed path in Figure 4.3 on the right is presented in the same figure on the left.

Now we can collect all the steps required to find a maximal flow on any network. Let G be a network with capacity function U.

1. First let Y be the zero flow on the network G, which assigns the value zero to every directed edge of G. Note that this is a valid flow for any network G.

2. Create the associated graph G' for the network G and flow Y.

3. If there's no directed path in G' connecting the source s to the sink t, then Y is maximal and the procedure terminates. Otherwise choose a shortest directed path in G' connecting the source s to the sink t.

4. Consider the undirected path in G corresponding to the directed path we've chosen in the previous step. This is a flow-augmenting path for the flow Y.

5. Compute the value of the flow-augmenting path found in the previous step.

6. Decrease the values of Y on the backward edges, and increase the values of Y on the forward edges of the flow-augmenting path by the value of the flow-augmenting path.

7. Repeat steps (2)-(6) as long as possible.