Discrete Mathematical Models

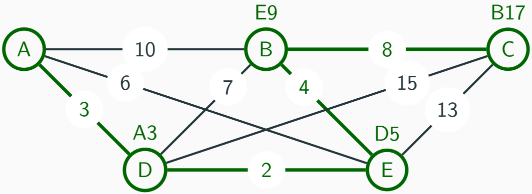
Lecture 26

Kane Townsend

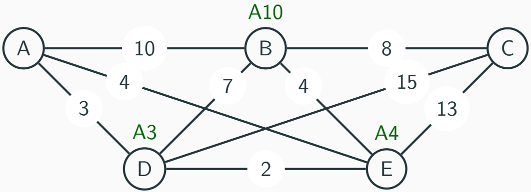
Semester 2, 2024

Dijkstra's Algorithm again

Last lecture we found a minimal path from A to C in the following weighted graph:



We now consider the same problem but with an adjustment to the graph:



Weighted digraphs

Weighted digraphs

In some applications of digraphs there is a non-negative number associated with each edge, known as the **weight** of the edge.

Weighted digraphs

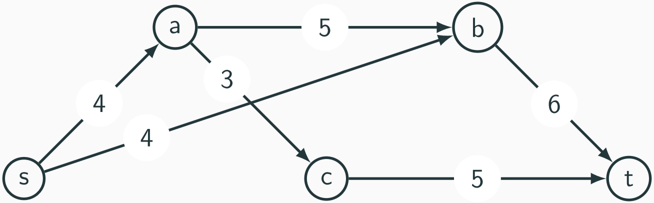
In some applications of digraphs there is a non-negative number associated with each edge, known as the **weight** of the edge.

A **weighted digraph** is a digraph G together with a **weight function** $\text{weight} : E(G) \to \mathbb{Q}_+$.

Transport Networks

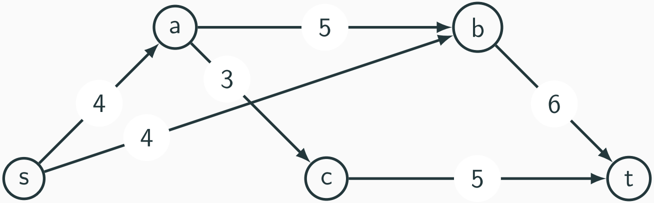
Transport networks

The digraph below is an example of a simple **transport network**:



Transport networks

The digraph below is an example of a simple **transport network**:

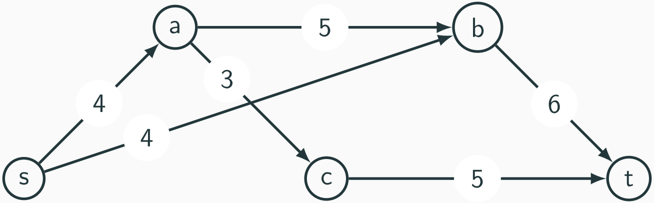


The defining features of a simple transport network are:

- The edges are weighted with positive weights, called **capacities**.

Transport networks

The digraph below is an example of a simple **transport network**:

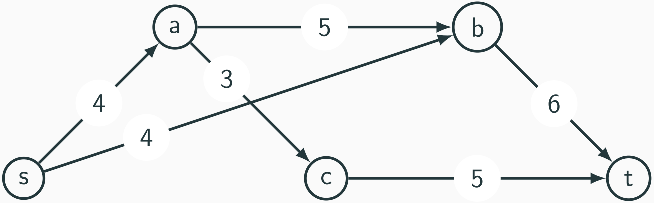


The defining features of a simple transport network are:

- The edges are weighted with positive weights, called **capacities**.
- There is exactly one special vertex called the **source** (labelled ‘s’) which is not the end vertex of any edge.

Transport networks

The digraph below is an example of a simple **transport network**:

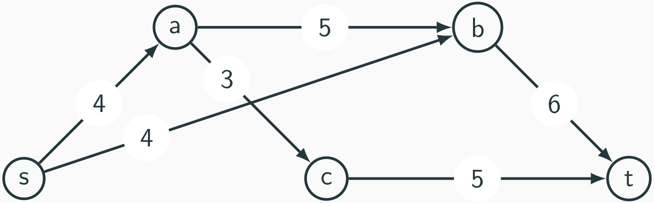


The defining features of a simple transport network are:

- The edges are weighted with positive weights, called **capacities**.
- There is exactly one special vertex called the **source** (labelled ‘s’) which is not the end vertex of any edge.
- There is exactly one special vertex called the **sink** or **target** (labelled ‘t’) which is not the start vertex of any edge.

Transport networks

The digraph below is an example of a simple **transport network**:



The defining features of a simple transport network are:

- The edges are weighted with positive weights, called **capacities**.
- There is exactly one special vertex called the **source** (labelled 's') which is not the end vertex of any edge.
- There is exactly one special vertex called the **sink** or **target** (labelled 't') which is not the start vertex of any edge.
- Every edge lies on some simple (directed) path from s to t.

Flows

The motivation for transport networks is to model the *flow* of something, such as gas or information, through a network of links.

Flows

The motivation for transport networks is to model the *flow* of something, such as gas or information, through a network of links.

A simple example to keep in mind is water flowing from a reservoir (the source) through various pipes (the edges) and exchange nodes (the vertices) to another reservoir (the sink, or target).

Flows

The motivation for transport networks is to model the *flow* of something, such as gas or information, through a network of links.

A simple example to keep in mind is water flowing from a reservoir (the source) through various pipes (the edges) and exchange nodes (the vertices) to another reservoir (the sink, or target).

An 'active' transport network has a flow $F(e) \geq 0$ through each edge e.

Flows

The motivation for transport networks is to model the *flow* of something, such as gas or information, through a network of links.

A simple example to keep in mind is water flowing from a reservoir (the source) through various pipes (the edges) and exchange nodes (the vertices) to another reservoir (the sink, or target).

An 'active' transport network has a flow $F(e) \geq 0$ through each edge e. The **flow** F, like the capacities C, is a function $E(D) \to \mathbb{Q}_+$, where D is the digraph representing the network.

Flows

The motivation for transport networks is to model the *flow* of something, such as gas or information, through a network of links.

A simple example to keep in mind is water flowing from a reservoir (the source) through various pipes (the edges) and exchange nodes (the vertices) to another reservoir (the sink, or target).

An 'active' transport network has a flow $F(e) \geq 0$ through each edge e. The **flow** F, like the capacities C, is a function $E(D) \to \mathbb{Q}_+$, where D is the digraph representing the network. However, each edge e has a *fixed* capacity $C(e)$ whereas its flow $F(e)$ can vary, subject to constraints:

Flows

The motivation for transport networks is to model the *flow* of something, such as gas or information, through a network of links.

A simple example to keep in mind is water flowing from a reservoir (the source) through various pipes (the edges) and exchange nodes (the vertices) to another reservoir (the sink, or target).

An 'active' transport network has a flow $F(e) \geq 0$ through each edge e. The **flow** F, like the capacities C, is a function $E(D) \to \mathbb{Q}_+$, where D is the digraph representing the network. However, each edge e has a *fixed* capacity $C(e)$ whereas its flow $F(e)$ can vary, subject to constraints:

(1) Flow cannot exceed capacity. $[\forall e \in E(D) \;\; F(e) \leq C(e).]$

Flows

The motivation for transport networks is to model the *flow* of something, such as gas or information, through a network of links.

A simple example to keep in mind is water flowing from a reservoir (the source) through various pipes (the edges) and exchange nodes (the vertices) to another reservoir (the sink, or target).

An 'active' transport network has a flow $F(e) \geq 0$ through each edge e. The **flow** F, like the capacities C, is a function $E(D) \to \mathbb{Q}_+$, where D is the digraph representing the network. However, each edge e has a *fixed* capacity $C(e)$ whereas its flow $F(e)$ can vary, subject to constraints:

(1) Flow cannot exceed capacity. $[\forall e \in E(D) \;\; F(e) \leq C(e).]$

(2) In each edge, flow direction = edge direction.

Flows

The motivation for transport networks is to model the *flow* of something, such as gas or information, through a network of links.

A simple example to keep in mind is water flowing from a reservoir (the source) through various pipes (the edges) and exchange nodes (the vertices) to another reservoir (the sink, or target).

An 'active' transport network has a flow $F(e) \geq 0$ through each edge e. The **flow** F, like the capacities C, is a function $E(D) \to \mathbb{Q}_+$, where D is the digraph representing the network. However, each edge e has a *fixed* capacity $C(e)$ whereas its flow $F(e)$ can vary, subject to constraints:

(1) Flow cannot exceed capacity. [$\forall e \in E(D) \;\; F(e) \leq C(e)$.]

(2) In each edge, flow direction = edge direction.

(3) Total flow into a node equals total flow out, except for nodes s, t. [$\forall v \in V(D) \backslash \{s, t\} \;\; \sum_{e \in v_{\mathbf{in}}} F(e) = \sum_{e \in v_{\mathbf{out}}} F(e)$, where $v_{\text{in}}, v_{\text{out}}$ are the sets of edges coming **in** to, and **out** of, v, respectively.]

Finding a maximum flow

For any flow F the constraints on the network imply that the total flow out of the source must equal the total flow into the target:

$$\sum_{e \in s_{\mathbf{out}}} F(e) \;=\; \sum_{e \in t_{\mathbf{in}}} F(e) \;=\; v(F) \text{ say.}$$

Finding a maximum flow

For any flow F the constraints on the network imply that the total flow out of the source must equal the total flow into the target:

$$\sum_{e \in s_{\mathbf{out}}} F(e) \;=\; \sum_{e \in t_{\mathbf{in}}} F(e) \;=\; v(F) \text{ say.}$$

This common total $v(F)$ is the **flow value** and any flow achieving the maximum possible flow value is called a **maximum flow**, F_{max}. (There may be several flows that achieve the maximum.)

Finding a maximum flow

For any flow F the constraints on the network imply that the total flow out of the source must equal the total flow into the target:

$$\sum_{e \in s_{\mathbf{out}}} F(e) \;=\; \sum_{e \in t_{\mathbf{in}}} F(e) \;=\; v(F) \text{ say.}$$

This common total $v(F)$ is the **flow value** and any flow achieving the maximum possible flow value is called a **maximum flow**, F_{max}. (There may be several flows that achieve the maximum.)

I will demonstrate a vertex labelling algorithm for finding an F_{max}.

Finding a maximum flow

For any flow F the constraints on the network imply that the total flow out of the source must equal the total flow into the target:

$$\sum_{e \in s_{\mathbf{out}}} F(e) \;=\; \sum_{e \in t_{\mathbf{in}}} F(e) \;=\; v(F) \text{ say.}$$

This common total $v(F)$ is the **flow value** and any flow achieving the maximum possible flow value is called a **maximum flow**, F_{max}. (There may be several flows that achieve the maximum.)

I will demonstrate a vertex labelling algorithm for finding an F_{max}.
The algorithm starts with the **zero flow** F_0, where $\forall e \in D \;\; F_0(e) = 0$.

Finding a maximum flow

For any flow F the constraints on the network imply that the total flow out of the source must equal the total flow into the target:

$$\sum_{e \in s_{\mathbf{out}}} F(e) \;=\; \sum_{e \in t_{\mathbf{in}}} F(e) \;=\; v(F) \text{ say.}$$

This common total $v(F)$ is the **flow value** and any flow achieving the maximum possible flow value is called a **maximum flow**, F_{max}. (There may be several flows that achieve the maximum.)

I will demonstrate a vertex labelling algorithm for finding an F_{max}. The algorithm starts with the **zero flow** F_0, where $\forall e \in D\ F_0(e) = 0$. Then, over a number of stages, expanded flows $F_1, F_2, \ldots$ are obtained until F_{max} is reached.

Finding a maximum flow

For any flow F the constraints on the network imply that the total flow out of the source must equal the total flow into the target:

$$\sum_{e \in s_{\textbf{out}}} F(e) \;=\; \sum_{e \in t_{\textbf{in}}} F(e) \;=\; v(F) \text{ say.}$$

This common total $v(F)$ is the **flow value** and any flow achieving the maximum possible flow value is called a **maximum flow**, $F_{\max}$. (There may be several flows that achieve the maximum.)

I will demonstrate a vertex labelling algorithm for finding an $F_{\max}$.
The algorithm starts with the **zero flow** F_0, where $\forall e \in D\ F_0(e) = 0$.
Then, over a number of stages, expanded flows $F_1, F_2, \ldots$ are obtained until $F_{\max}$ is reached.

At stage i, flow F_i is constructed as $F_i = F_{i-1} + f_i$, where the incremental flow f_i is based on a constant $k_i \in \mathbb{Q}^+$ and a simple path p_i from s to t:

$$f_i(e) = \begin{cases} k_i & \text{for every edge } e \text{ on the path } p_i \\ 0 & \text{for evey other edge } e. \end{cases}$$

Spare capacity

Giving a formal description of the vertex labelling algorithm at this point may be a little overwhelming, so I will start with an example.

Spare capacity

Giving a formal description of the vertex labelling algorithm at this point may be a little overwhelming, so I will start with an example.

But I need to first explain my terminology and mark-up conventions.

Spare capacity

Giving a formal description of the vertex labelling algorithm at this point may be a little overwhelming, so I will start with an example.
But I need to first explain my terminology and mark-up conventions.

Given a transport network D with capacity and flow functions C, F, we know that $F(e) \leq C(e)$ for every edge $e \in E(D)$. I will call the non-negative difference $S(e) = C(e) - F(e)$ the **spare capacity** of the edge e. (Some authors use the term “excess capacity”.)

Spare capacity

Giving a formal description of the vertex labelling algorithm at this point may be a little overwhelming, so I will start with an example.
But I need to first explain my terminology and mark-up conventions.

Given a transport network D with capacity and flow functions C, F, we know that $F(e) \leq C(e)$ for every edge $e \in E(D)$. I will call the non-negative difference $S(e) = C(e) - F(e)$ the **spare capacity** of the edge e. (Some authors use the term "excess capacity".)

To depict a flow I will follow the capacity value $C(e)$ on each (directed) edge e with the flow value $F(e)$ for that edge. For example

represents a flow of 3 in the edge from x to y, with spare capacity 2.

Levels and labels

At each stage of the vertex labelling algorithm, levels and labels are associated afresh with the vertices of the network.

Levels and labels

At each stage of the vertex labelling algorithm, levels and labels are associated afresh with the vertices of the network.

The **level** of a vertex is determined iteratively as follows:

Levels and labels

At each stage of the vertex labelling algorithm, levels and labels are associated afresh with the vertices of the network.

The **level** of a vertex is determined iteratively as follows:

- The source vertex s always has level 0.

Levels and labels

At each stage of the vertex labelling algorithm, levels and labels are associated afresh with the vertices of the network.

The **level** of a vertex is determined iteratively as follows:

- The source vertex s always has level 0.
- If $e = (s, x)$ and $S(e) > 0$ then x has level 1.

Levels and labels

At each stage of the vertex labelling algorithm, levels and labels are associated afresh with the vertices of the network.

The **level** of a vertex is determined iteratively as follows:

- The source vertex s always has level 0.
- If $e = (s, x)$ and $S(e) > 0$ then x has level 1.
- If x has level n and $S((x, y)) > 0$ then y has level $n + 1$ provided it has not already been assigned a lower level.

Levels and labels

At each stage of the vertex labelling algorithm, levels and labels are associated afresh with the vertices of the network.

The **level** of a vertex is determined iteratively as follows:

- The source vertex s always has level 0.
- If $e = (s, x)$ and $S(e) > 0$ then x has level 1.
- If x has level n and $S((x, y)) > 0$ then y has level $n + 1$ provided it has not already been assigned a lower level.

The **label** on a vertex v of level $n \geq 1$ has the form uk, where u is a vertex of level $n-1$ and $(u,v) \in E(D)$ is an edge on the path for a potential incremental flow through v with flow value k.

Levels and labels

At each stage of the vertex labelling algorithm, levels and labels are associated afresh with the vertices of the network.

The **level** of a vertex is determined iteratively as follows:

- The source vertex s always has level 0.
- If $e = (s, x)$ and $S(e) > 0$ then x has level 1.
- If x has level n and $S((x, y)) > 0$ then y has level $n + 1$ provided it has not already been assigned a lower level.

The **label** on a vertex v of level $n \geq 1$ has the form uk, where u is a vertex of level $n-1$ and $(u,v) \in E(D)$ is an edge on the path for a potential incremental flow through v with flow value k.

The algorithm assigns labels in ascending order of levels, and in alphabetical order within levels.

Levels and labels

At each stage of the vertex labelling algorithm, levels and labels are associated afresh with the vertices of the network.

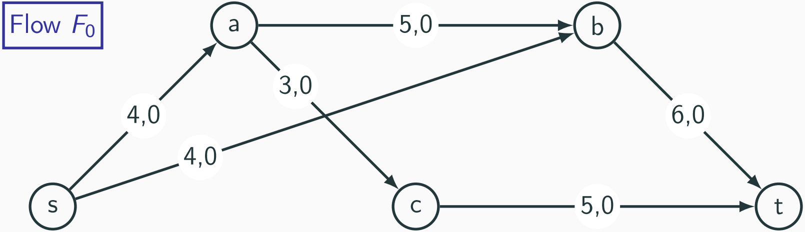
The **level** of a vertex is determined iteratively as follows:

- The source vertex s always has level 0.
- If $e = (s, x)$ and $S(e) > 0$ then x has level 1.
- If x has level n and $S((x, y)) > 0$ then y has level $n + 1$ provided it has not already been assigned a lower level.

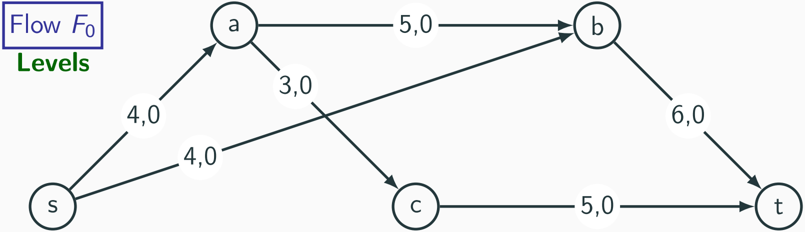
The **label** on a vertex v of level $n \geq 1$ has the form uk, where u is a vertex of level $n-1$ and $(u,v) \in E(D)$ is an edge on the path for a potential incremental flow through v with flow value k.

The algorithm assigns labels in ascending order of levels, and in alphabetical order within levels. Exactly how u and k are determined is explained in the formal description of the algorithm, but you may be able to infer the rule from the next example.

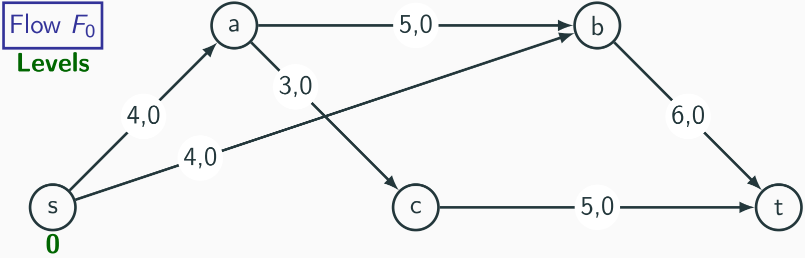
Vertex labelling algorithm, Example 1: Stage 1: F_0 to F_1



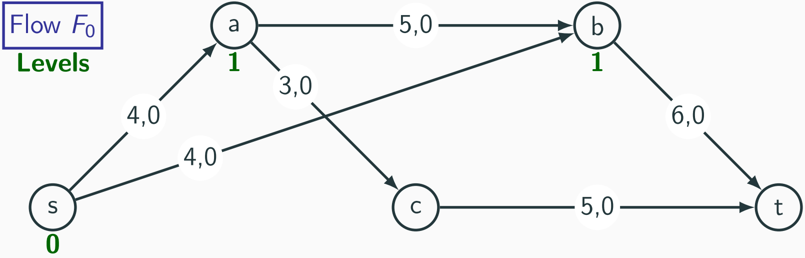
Vertex labelling algorithm, Example 1: Stage 1: F_0 to F_1



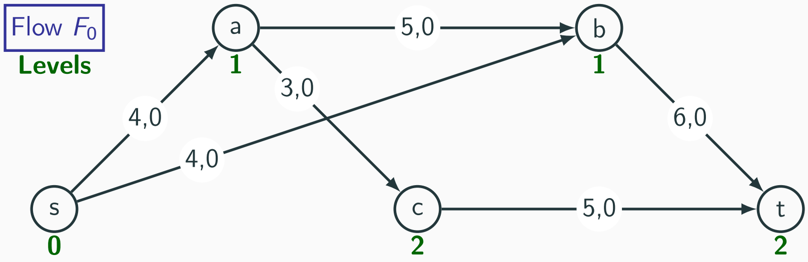
Vertex labelling algorithm, Example 1: Stage 1: F_0 to F_1



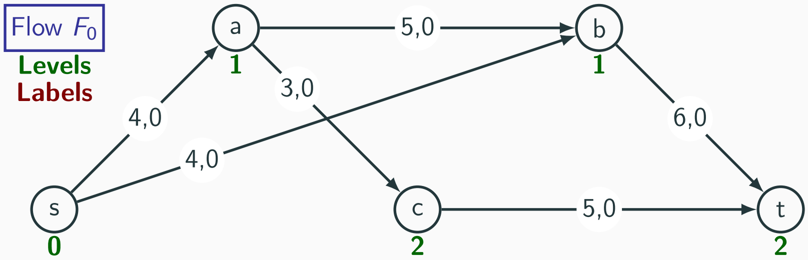
Vertex labelling algorithm, Example 1: Stage 1: F_0 to F_1



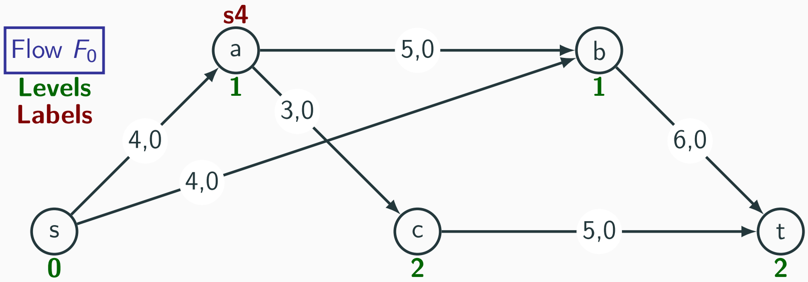
Vertex labelling algorithm, Example 1: Stage 1: F_0 to F_1



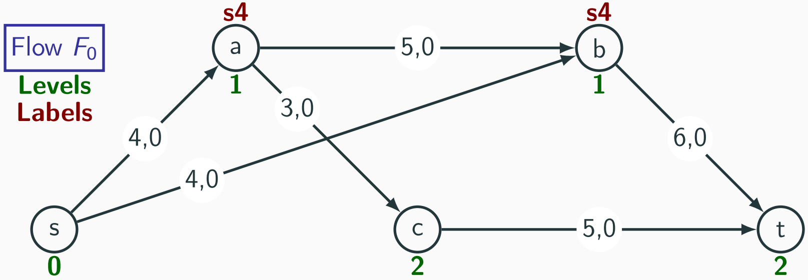
Vertex labelling algorithm, Example 1: Stage 1: F_0 to F_1



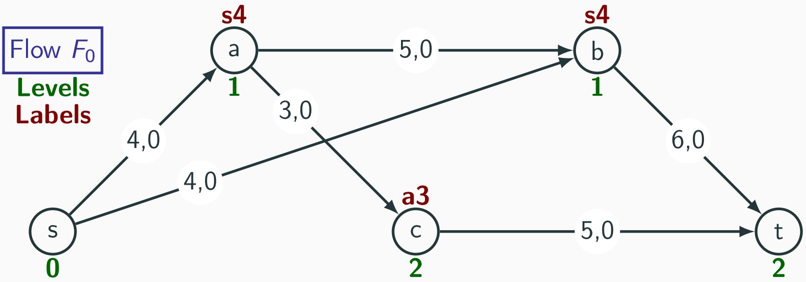
Vertex labelling algorithm, Example 1: Stage 1: F_0 to F_1



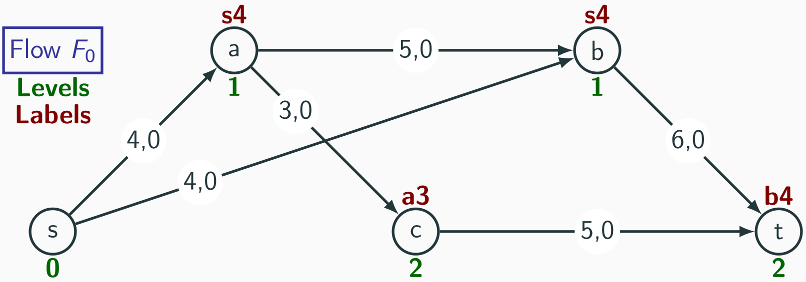
Vertex labelling algorithm, Example 1: Stage 1: F_0 to F_1



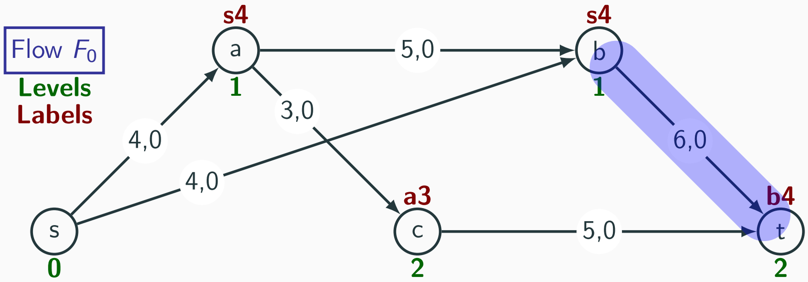
Vertex labelling algorithm, Example 1: Stage 1: F_0 to F_1



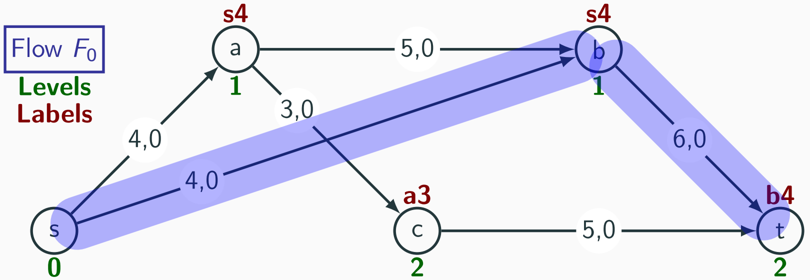
Vertex labelling algorithm, Example 1: Stage 1: F_0 to F_1



Vertex labelling algorithm, Example 1: Stage 1: F_0 to F_1

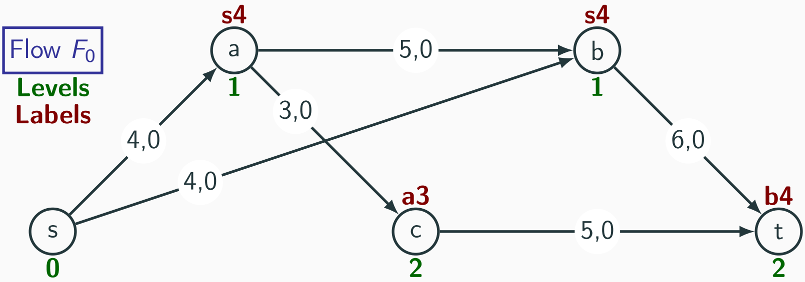


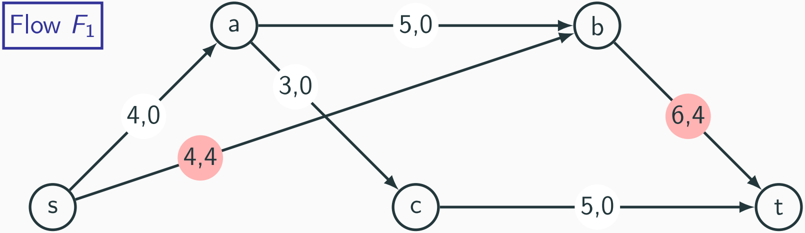
Vertex labelling algorithm, Example 1: Stage 1: F_0 to F_1



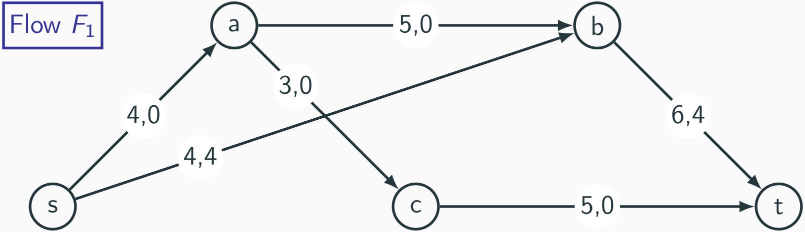
The incremental flow f_1

Vertex labelling algorithm, Example 1: Stage 1: F_0 to F_1

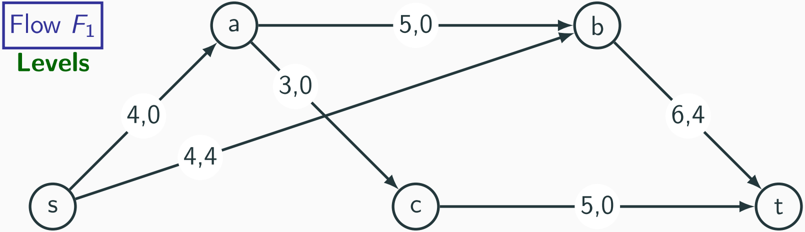




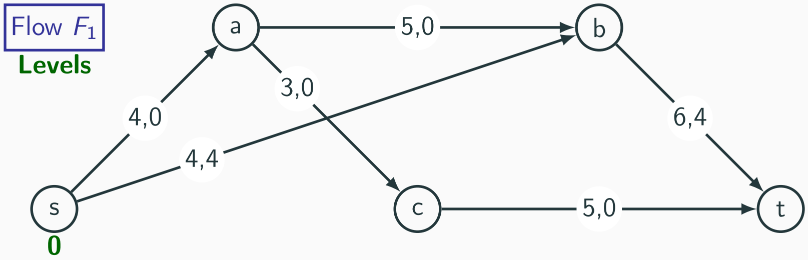
Vertex labelling algorithm, Example 1: Stage 2: F_1 to F_2



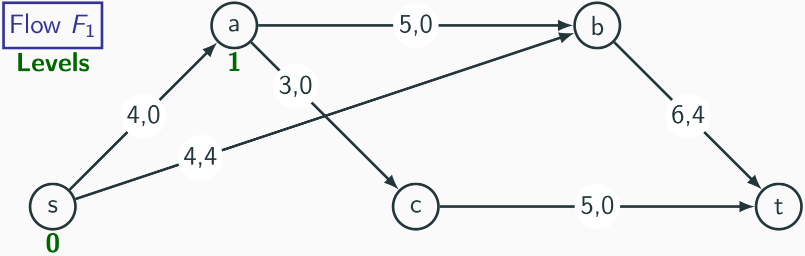
Vertex labelling algorithm, Example 1: Stage 2: F_1 to F_2



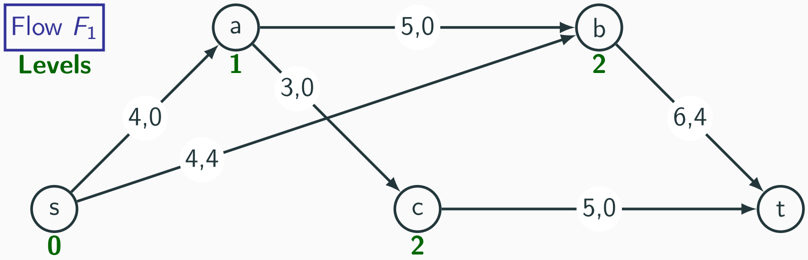
Vertex labelling algorithm, Example 1: Stage 2: F_1 to F_2



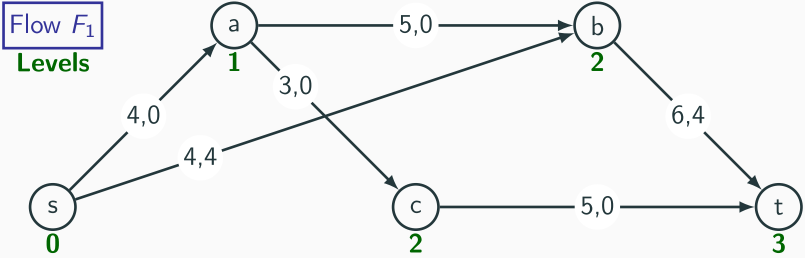
Vertex labelling algorithm, Example 1: Stage 2: F_1 to F_2



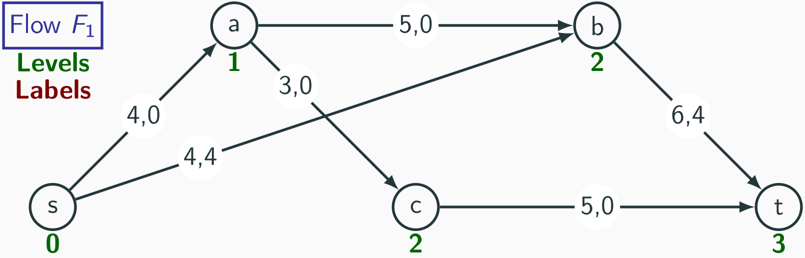
Vertex labelling algorithm, Example 1: Stage 2: F_1 to F_2



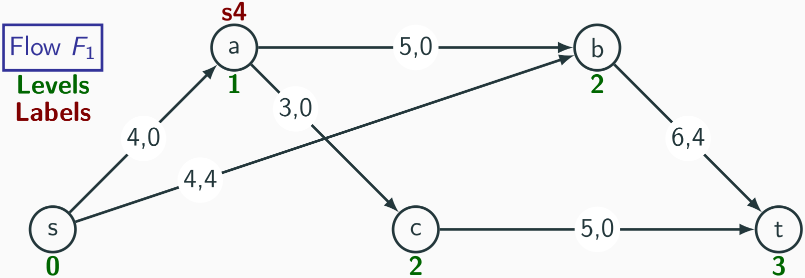
Vertex labelling algorithm, Example 1: Stage 2: F_1 to F_2



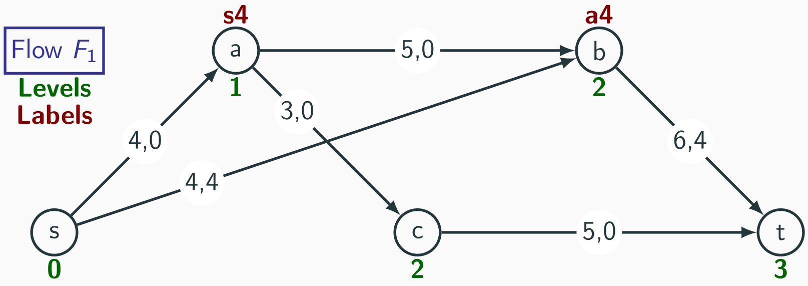
Vertex labelling algorithm, Example 1: Stage 2: F_1 to F_2



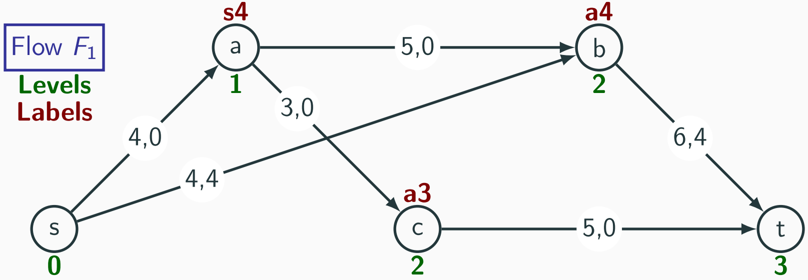
Vertex labelling algorithm, Example 1: Stage 2: F_1 to F_2



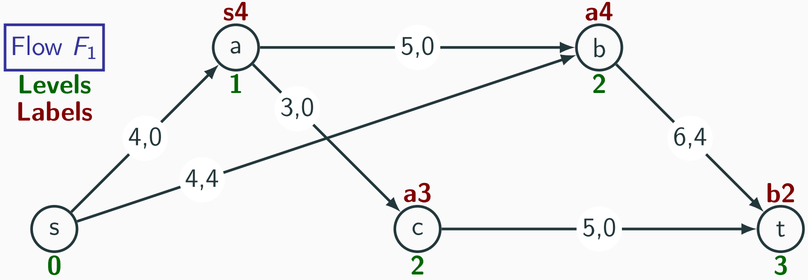
Vertex labelling algorithm, Example 1: Stage 2: F_1 to F_2



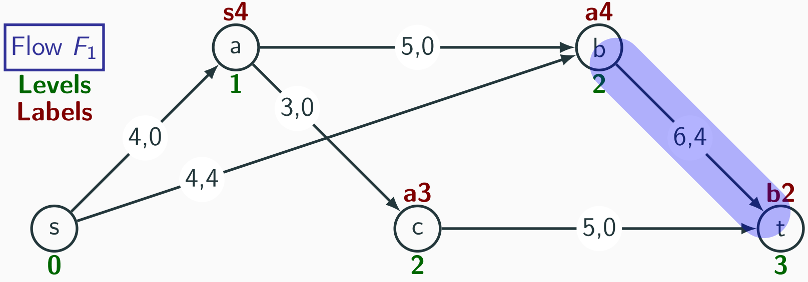
Vertex labelling algorithm, Example 1: Stage 2: F_1 to F_2



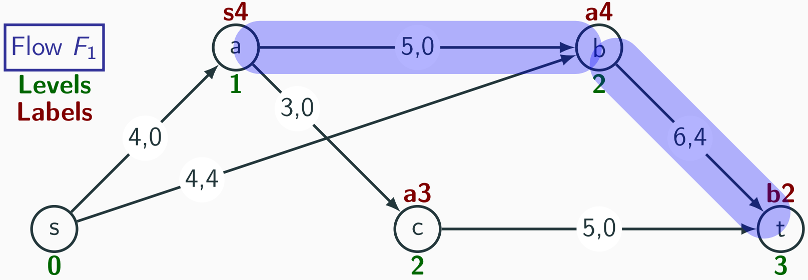
Vertex labelling algorithm, Example 1: Stage 2: F_1 to F_2



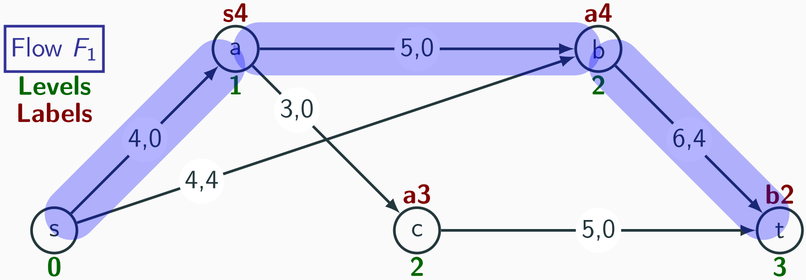
Vertex labelling algorithm, Example 1: Stage 2: F_1 to F_2



Vertex labelling algorithm, Example 1: Stage 2: F_1 to F_2

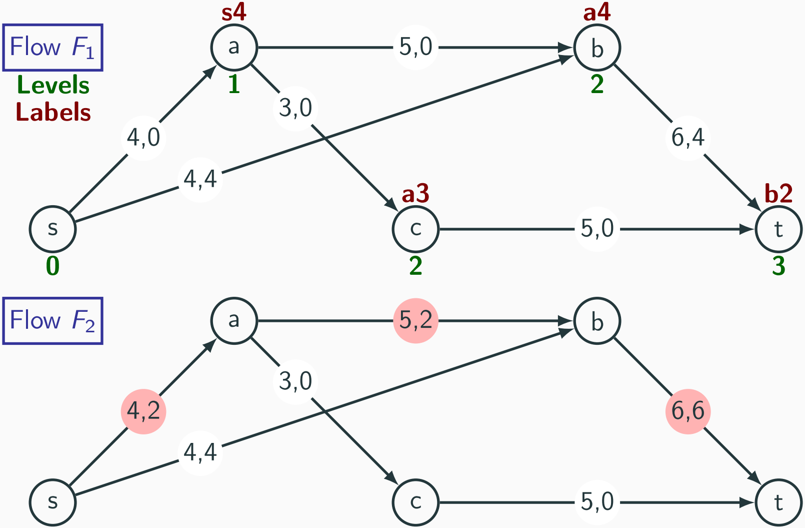


Vertex labelling algorithm, Example 1: Stage 2: F_1 to F_2

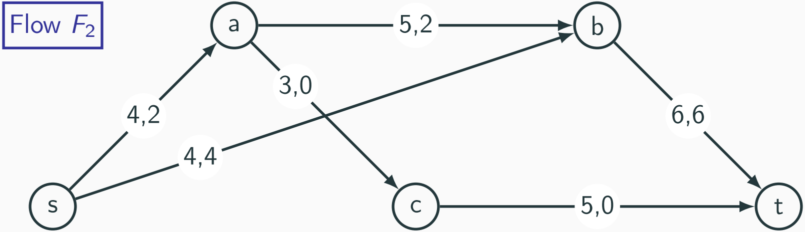


The incremental flow f_2

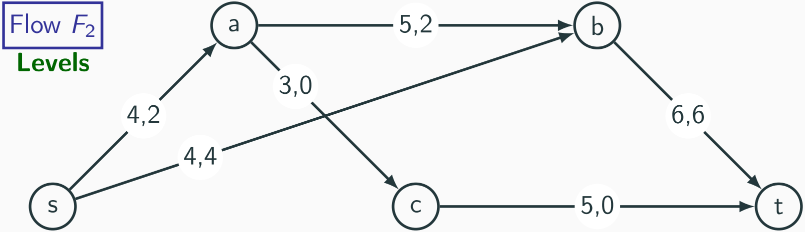
Vertex labelling algorithm, Example 1: Stage 2: F_1 to F_2



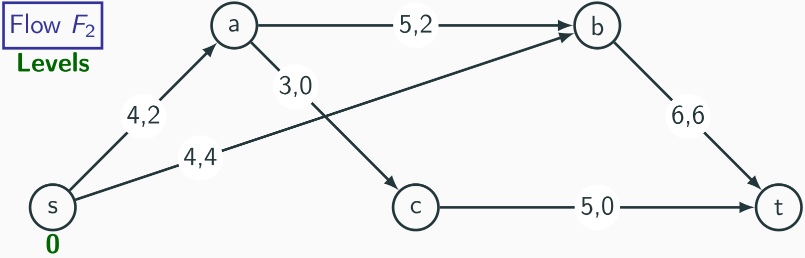
Vertex labelling algorithm, Example 1: Stage 3: F_2 to F_3



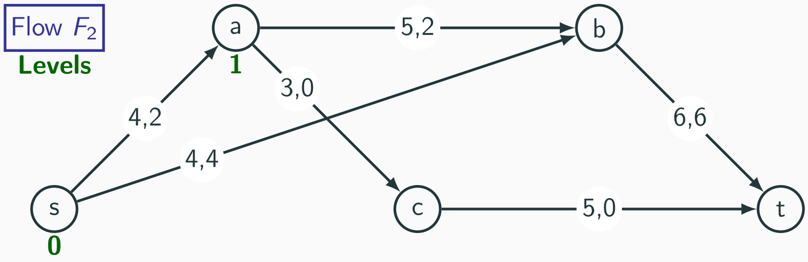
Vertex labelling algorithm, Example 1: Stage 3: F_2 to F_3



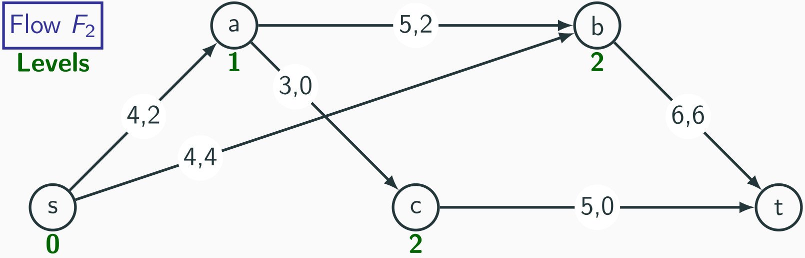
Vertex labelling algorithm, Example 1: Stage 3: F_2 to F_3



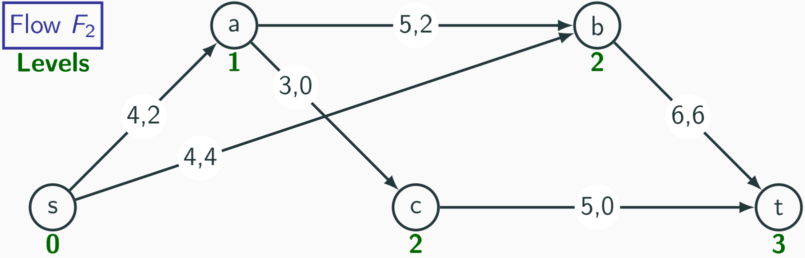
Vertex labelling algorithm, Example 1: Stage 3: F_2 to F_3



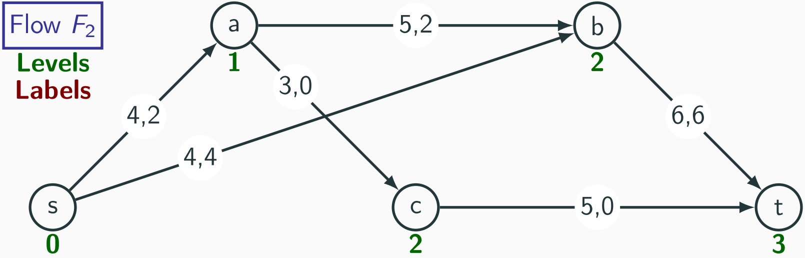
Vertex labelling algorithm, Example 1: Stage 3: F_2 to F_3



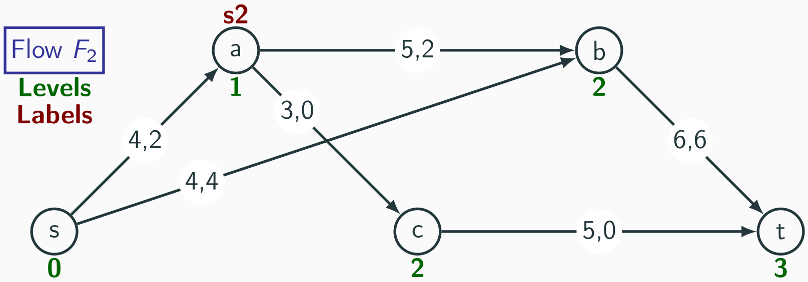
Vertex labelling algorithm, Example 1: Stage 3: F_2 to F_3



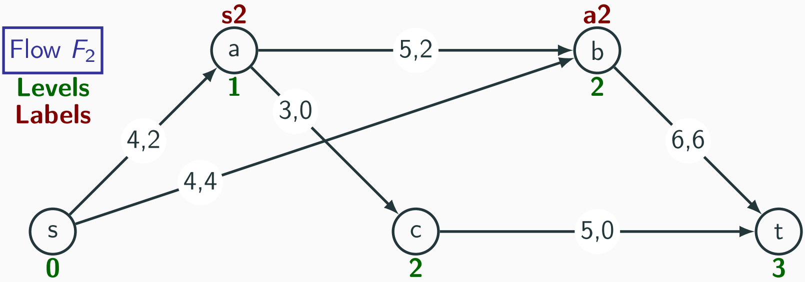
Vertex labelling algorithm, Example 1: Stage 3: F_2 to F_3



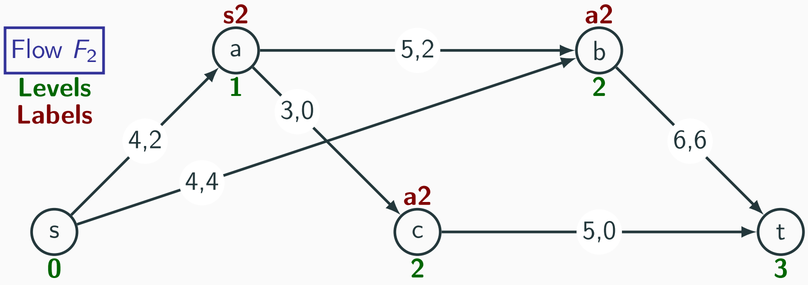
Vertex labelling algorithm, Example 1: Stage 3: F_2 to F_3



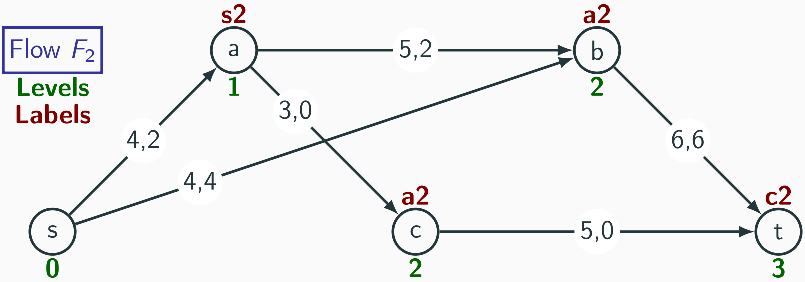
Vertex labelling algorithm, Example 1: Stage 3: F_2 to F_3



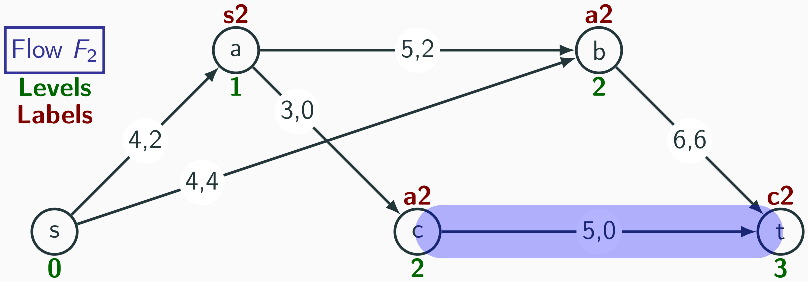
Vertex labelling algorithm, Example 1: Stage 3: F_2 to F_3



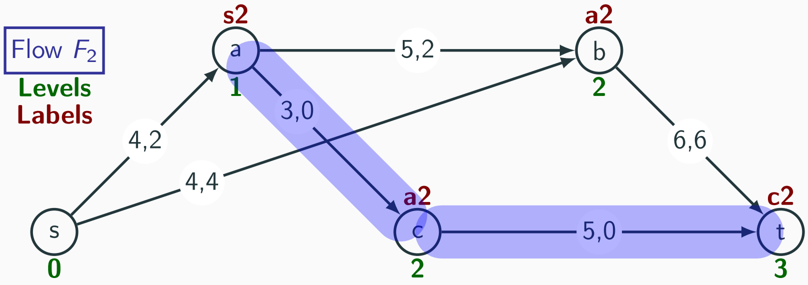
Vertex labelling algorithm, Example 1: Stage 3: F_2 to F_3



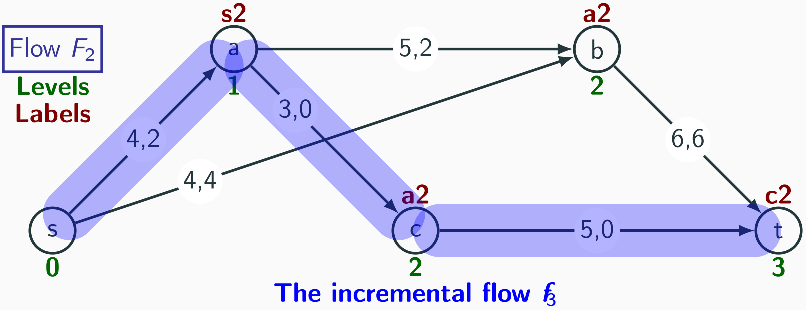
Vertex labelling algorithm, Example 1: Stage 3: F_2 to F_3



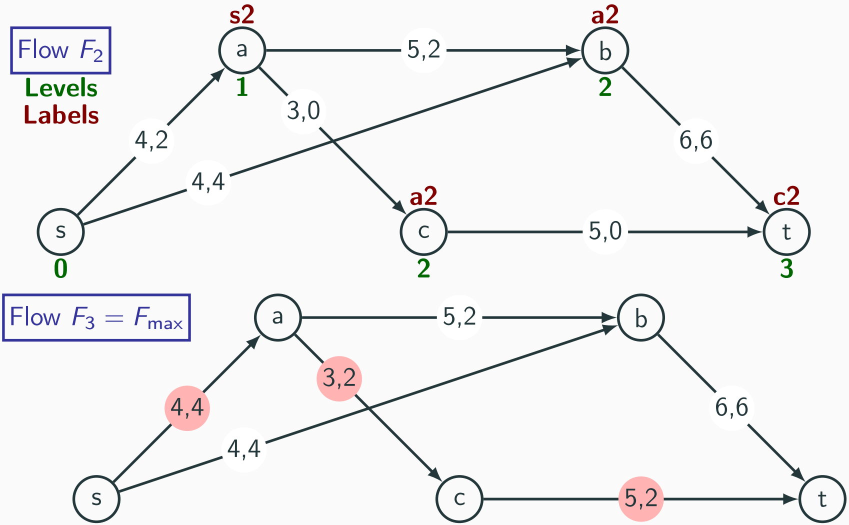
Vertex labelling algorithm, Example 1: Stage 3: F_2 to F_3



Vertex labelling algorithm, Example 1: Stage 3: F_2 to F_3



Vertex labelling algorithm, Example 1: Stage 3: F_2 to F_3



Vertex labelling algorithm, Formal description

The algorithm is unavoidably complicated to describe fully.
Moreover a further complication (virtual capacity) still needs adding.

Vertex labelling algorithm, Formal description

The algorithm is unavoidably complicated to describe fully.
Moreover a further complication (virtual capacity) still needs adding.

Input: Transport network D with capacity function C.

Output: A maximum flow function F_{max} for the network.

Vertex labelling algorithm, Formal description

The algorithm is unavoidably complicated to describe fully.
Moreover a further complication (virtual capacity) still needs adding.

Input: Transport network D with capacity function C.

Output: A maximum flow function $F_{\max}$ for the network.

Method: Initialise F to the zero flow F_0. Initialize i to 1.
For $i = 1, 2, \ldots$ carry out stage i below to attempt to build an incremental flow f_i.

Vertex labelling algorithm, Formal description

The algorithm is unavoidably complicated to describe fully.
Moreover a further complication (virtual capacity) still needs adding.

Input: Transport network D with capacity function C.

Output: A maximum flow function $F_{\max}$ for the network.

Method: Initialise F to the zero flow F_0. Initialize i to 1.
For $i = 1, 2, \ldots$ carry out stage i below to attempt to build an incremental flow f_i.
If stage i succeeds, define $F_i = F_{i-1} + f_i$ and proceed to stage $i+1$.
If stage i fails, define $F_{\max} = F_{i-1}$ and stop.

Stage i:

1. If $i > 1$, mark up the amended edge flows for F_{i-1}.

Vertex labelling algorithm, Formal description

The algorithm is unavoidably complicated to describe fully.
Moreover a further complication (virtual capacity) still needs adding.

Input: Transport network D with capacity function C.

Output: A maximum flow function $F_{\max}$ for the network.

Method: Initialise F to the zero flow F_0. Initialize i to 1.
For $i = 1, 2, \ldots$ carry out stage i below to attempt to build an incremental flow f_i.
If stage i succeeds, define $F_i = F_{i-1} + f_i$ and proceed to stage $i+1$.
If stage i fails, define $F_{\max} = F_{i-1}$ and stop.

Stage i:

1. If $i > 1$, mark up the amended edge flows for F_{i-1}.
2. Mark up the levels for F_{i-1}, as explained earlier.

Vertex labelling algorithm, Formal description

The algorithm is unavoidably complicated to describe fully.
Moreover a further complication (virtual capacity) still needs adding.

Input: Transport network D with capacity function C.

Output: A maximum flow function F_{max} for the network.

Method: Initialise F to the zero flow F_0. Initialize i to 1.
For $i = 1, 2, \ldots$ carry out stage i below to attempt to build an incremental flow f_i.
If stage i succeeds, define $F_i = F_{i-1} + f_i$ and proceed to stage $i+1$.
If stage i fails, define $F_{\text{max}} = F_{i-1}$ and stop.

Stage i:

1. If $i > 1$, mark up the amended edge flows for F_{i-1}.
2. Mark up the levels for F_{i-1}, as explained earlier.
3. Next slide....

3. If t is assigned a level, stage i will succeed, so continue. If not, then stage i fails, so return above to define F_{max} and terminate.

3. If t is assigned a level, stage i will succeed, so continue. If not, then stage i fails, so return above to define $F_{\max}$ and terminate.
4. Mark up labels for F_{i-1} as follows until t is labelled:

3. If t is assigned a level, stage i will succeed, so continue. If not, then stage i fails, so return above to define $F_{\max}$ and terminate.
4. Mark up labels for F_{i-1} as follows until t is labelled:
    (a) Label each level 1 vertex v with sk_{v}, where $k_{\mathrm{v}} = S((s,v))$.

3. If t is assigned a level, stage i will succeed, so continue. If not, then stage i fails, so return above to define $F_{\max}$ and terminate.
4. Mark up labels for F_{i-1} as follows until t is labelled:
   (a) Label each level 1 vertex v with sk_{v}, where $k_{\text{v}} = S((s,v))$.
   (b) If t has level 2 or more now work through the level 2 vertices in alphabetical order, labelling each vertex v with uk_u, where

3. If t is assigned a level, stage i will succeed, so continue.
   If not, then stage i fails, so return above to define $F_{\max}$ and terminate.
4. Mark up labels for F_{i-1} as follows until t is labelled:
   (a) Label each level 1 vertex v with sk_{v}, where $k_{\mathrm{v}} = S((s,v))$.
   (b) If t has level 2 or more now work through the level 2 vertices in alphabetical order, labelling each vertex v with uk_u, where
      - u is the alphabetically earliest level 1 vertex with $(u,v) \in E(D)$ and $S((u,v)) > 0$,

3. If t is assigned a level, stage i will succeed, so continue.
   If not, then stage i fails, so return above to define F_{max} and terminate.
4. Mark up labels for F_{i-1} as follows until t is labelled:
   (a) Label each level 1 vertex v with sk_{v}, where $k_{\text{v}} = S((s,v))$.
   (b) If t has level 2 or more now work through the level 2 vertices in alphabetical order, labelling each vertex v with uk_u, where
      - u is the alphabetically earliest level 1 vertex with $(u,v) \in E(D)$ and $S((u,v)) > 0$,
      - k_{u} is the minimum of $S((u,v))$ and the value part of u's label.

3. If t is assigned a level, stage i will succeed, so continue. If not, then stage i fails, so return above to define $F_{\max}$ and terminate.
4. Mark up labels for F_{i-1} as follows until t is labelled:
   (a) Label each level 1 vertex v with sk_{v}, where $k_{\mathrm{v}} = S((s,v))$.
   (b) If t has level 2 or more now work through the level 2 vertices in alphabetical order, labelling each vertex v with uk_u, where
      - u is the alphabetically earliest level 1 vertex with $(u,v) \in E(D)$ and $S((u,v)) > 0$,
      - $k_{\mathbf{u}}$ is the minimum of $S((u,v))$ and the value part of u's label.

   (c) If t has level 3 or more now work through the level 3 vertices in a similar manner and so on.

3. If t is assigned a level, stage i will succeed, so continue.
   If not, then stage i fails, so return above to define $F_{\max}$ and terminate.
4. Mark up labels for F_{i-1} as follows until t is labelled:
   (a) Label each level 1 vertex v with sk_v, where $k_\mathrm{v} = S((s,v))$.
   (b) If t has level 2 or more now work through the level 2 vertices in alphabetical order, labelling each vertex v with uk_u, where
      - u is the alphabetically earliest level 1 vertex with $(u,v) \in E(D)$ and $S((u,v)) > 0$,
      - k_u is the minimum of $S((u,v))$ and the value part of u's label.

   (c) If t has level 3 or more now work through the level 3 vertices in a similar manner and so on.
5. Let p_i be the path $\mathrm{u}_0\mathrm{u}_1 \ldots \mathrm{u}_n$ where $\mathrm{u}_n = \mathrm{t}$ and for $0 < j \leq n$ u_j has label $\mathrm{u}_{j-1}k_j$.
   Define f_i to be the incremental flow on p_i with flow value k_n.

End of Method