various

Lecture Notes of Spring 2013

Algorithms I

University of Mannheim
2013

This script originates from the course "‘Algorithms I"’ at the University of Mannheim as lecture notes.

The accuracy of its content is not guaranteed and the author(s) do not assume responsibilitty for possible damages of any kind. This lecture notes are not an official document released by employees of the University of Mannheim, hence those do not assume responsibility, as well.

Many thanks to Ingo Bürk, who initially published the underlying LaTeX template here.

Contents

1

TODO: Elementary Notions about Graphs

TODO

Definition 1.1
An undirected graph is a pair G=(V, E) where V is a set of nodes and E is a set of edges, together with a function i: E $\rightarrow \gamma(v)$ such that 0 < $|i(e)| \leq$ 2.
If u,v i(e) we call u,v endpoints of e. If V and E are a finite set we call G a finite graph. If i(e_1) = i(e_2) we call e_1, e_2 parallel edges.

If $|i(e)|$= 1 we call e a loop. The degree of a node v is the number of edges for which v is an endpoint where loops a counted twice. Is the degree of v = 0 then we call v isolated.

Example
TODO: Graphic missing!

Lemma 1.1
In a finite graph the number of nodes with odd degree is even.

Proof: $\sum_{i=1}^{n} degree(v_i) = 2 * |E|$
This is because we start with a graph, where each node is isolated. Then we insert one edge after another.
Case 1: i(e) = x then the degree of x is increased by 2
Case 2: i(e) = x, y then the degree of x and y are increased by 1
We asume that $v_1 ... v_i$ have an even degree and $v_i + 1 ... v_n$ have odd degree.

$\sum_{k=1}^{i} degree(v_k) + \sum_{k=i+1}^{n} degree(v_k) = 2|E|$

$\sum_{k=1}^{i} degree(v_k)$ is an even number

$\sum_{k=i+1}^{n} degree(v_k)$ must be an even number and hence the number of nodes with odd degree must be even

$2|E|$ is an even number

Definition 1.2
If G=(V,E) is a graph and $v_1, v_2 \in V$ with $i(e) = \{v1, v2\}$ then we say that v_1, v_2 are neighbours. A path in G is a sequence of edges $e_1, e_2, \ldots$ such that:

i) $e_i, e_i + 1$ share an endpoint

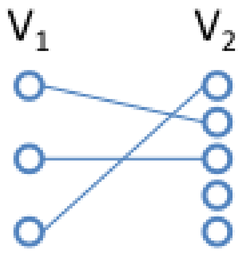
ii) if e_i is not a loop and neither the first nor the last edge. Then e_i shares one endpoint with $e_i - 1$ and the other with $e_i + 1$ [MS: does this make sense? sounds strange!]

Definition 1.3
TODO

Lemma 1.2
TODO

Definition 1.4
Let $G = (V, E)$ be a graph without loops. If there exists $V_1, V_2 \leq V$ and $V_1 \cup V_2 = V$ such that $V_1 \cap V_2 = \emptyset$ and every edge e has one endpoint in V_1 and the other in V_2, then we call G a bipartite.

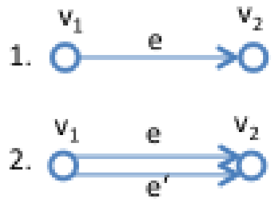
Example



Definition 1.5
A directed graph is a pair $G = (V, E)$ where V is a set of nodes (vertices) and E is a set of edges together with a function $i : E- > V x V$. If $i(e) = (v_1, v_2)$ then v_1 is called start point, v_2 is called end point.

Graphically:

If $i(e) = (v_1, v_2)$ we draw 1.
If $i(e') = (v_1, v_2)$ then this indices a second edge (2.).
If $i(e_1) = i(e_2)$ we call e_1, e_2 parallel.
If $i(e) = (v, v)$ then e is called a directed loop.



$g_{out}(v)$ is the number of edges that have starting point v.
$g_{in}(v)$ is the number of edges with endpoint v.

Lemma 1.3
$\sum\limits_{v \in V} g_{in}(v) = \sum\limits_{v \in V} g_{out}(v)$

Proof: We start with a graph without edges. Then we insert one after the other edges in E. Each edge contributes 1 to both sides of the equation.

Definition 1.6
A directed path is a sequence of edges $e_1, e_2 ...$ such that the end point of e_i is the start point of $e_1 + 1, i > 1$([NW] +1 seems strange to me, correct?).

A directed path $e_1 ... e_k$ is called a (directed) cycle, if the start point of e_1 and the end point of e_k coincide.
A simple (directed) path is a path where every node occurs at most once.
A directed cycle is called simple if every node except for the start and end node occurs at most once.

Definition 1.7
A graph directed or undirected is called simple, if it does not contain parallel edges.

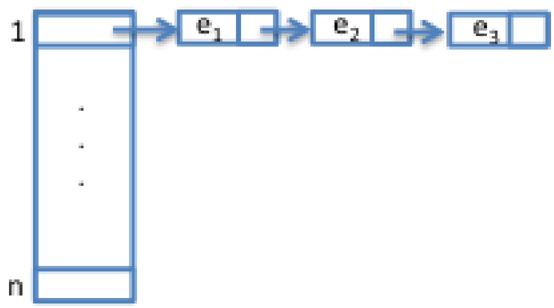
Definition 1.8
A directed graph is called strongly connected if for any pair of nodes (u, v) there is a directd path from u to v.

Let G be a directed graph $G = (V, E)$. $x, y \in V x \sim y$ ([NW] does ~mean are "connected"?) if there is a directed path from x to y and vice versa.

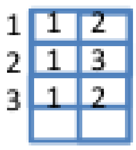
The equivalence classes of this relation $cVxV$ are called strongly connected components. (Analogously: Define connected components for undirected graphs)

We should know how the following terms are defined: reflexivity, symetry, transitivity.

Implementation:
1. Adjacency Lists $V = 1...n, E = e_1 ... e_t$



2. Dynamically changing graphs:
e.g. multi user databses: Nodes ≡ transactions of user; Edges ≡ waiting situations



Graph is used to detect dead locks. Waiting arises when data are locked by a user that modifies these data.
U_1 $write(d)$, $read(d')$
U_2 $read(d)$, $write(d')$

Definition 1.9
An undirected graph is called a tree if it is connected and does not have simple cycles.
Let G be a directed graph, $G = (V, E)$. A node r is called root if every other node can be reached from r via a directed path.
A directed graph is called a tree if it has a root and the underlying undirected graph is a tree.
Let G be a directed graph. A node is called source if $g_{in}(v) = 0$. v is called sink if $g_{out}(v) = 0$

Lemma 1.4
NW: what was lemma 1.3? the next one was 1.4 in my notes

Lemma 1.5
If $G = (V, E)$ is a directed graph without directed cycles, then tere is always a source and sink.

We use this theorem to detect cycles

Proof: Source (sink analogously): Select an arbitrary node v_1. If v_1 is a source we are done. If it is not, then there must be an edge e_1 leading to it $v_2 \xrightarrow{e_1} v_1$.
If v_2 is a source we are done. If not, there must be an edge e_2 leading to it $v_3 \xrightarrow{e_2} v_2 \xrightarrow{e_1} v_1$. We continue this process. It must stop because there are only finitely many nodes and if a node would appear once more on such a path, there would be a directed cycle.

2

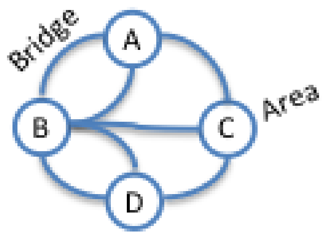
Euler Graphs and Hamilton Graphs

TODO

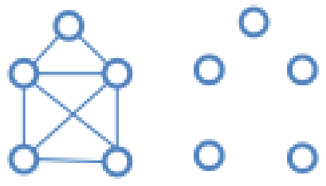
2.1 Euler Graphs

2.1.1 Euler 1736: Königsberger Brückenproblem

Is it possible to do a round walk crossing every bridge exactly once?



Example 2.1



Definition 2.1
Let G be a finite undirected graph. A path $e_1..e_t$ is called a euler path if every edge in E occurs exactly once in the list.
A graph is a euler graph if it has a euler path.

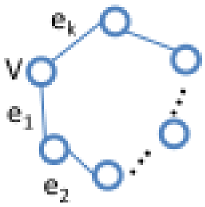
Theorem 2.1
A finite connected graph is a euler graph if and only if:

i) It ha eiter exactly two nodes of odd degree. or

ii) All nodes have even degree.

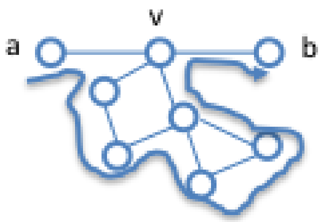
In the last case the path is a cycle. In the first case no euler path is a cycle. Check is possible in linear time.

Proof: " > " Let $G = (V,E)$ be a graph that has a euler path that is not a cycle. Let $|E| = k$ $\circ \xrightarrow{e_1} \circ \xrightarrow{e_2} \ldots \circ \xrightarrow{e_k}$ In this path v_1 and v_{k+1} have od degreee and all other nodes have even degree. Now consider the case teht G has a euler cycle.

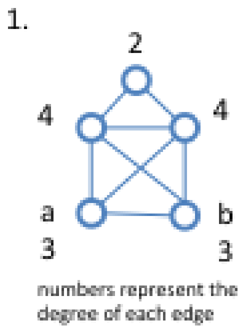


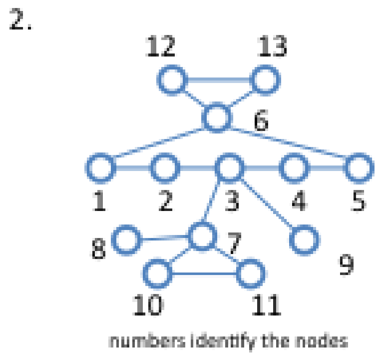
Hence every node has even degree.

" < " Let G be a graph with exactly two nodes with odd degree, let this be a and b. We contradict a euler path as follows:
Start at node a and folow an edge ?inktt? on a. $a \circ \to \circ \to \ldots \circ b$
Case 1: All edges have been used -> done
Case 2: Still edges unused. Then because G is connected there must be some node v on the path from which there is an unused edge. We construct a path starting from v as before. This path must end in v.
⇒ Repeat until there are no more unused edges.
Analogously we proceed when the degree of all nodes is even.



Example





In the directed case a directed Euler path is a directed path on which every edge appears exactly once. Directed Euler cycle analogously.

Theorem 2.2
A finite directed graph is a directed Euler graph if and only if its underlying undirected graph is connected.

i) There is one node a with $g_{out}(a) = g_{in}(a) + 1$ and another node $b g_{out}(b) = g_{in}(b) + 1$ and for all other nodes $v g_{in}(v) = g_{out}(v)$. Or

ii) For all nodes $g_{in}(v) = g_{out}(v)$ (directed Euler cyle)

2.2 Hamiltonian Graphs

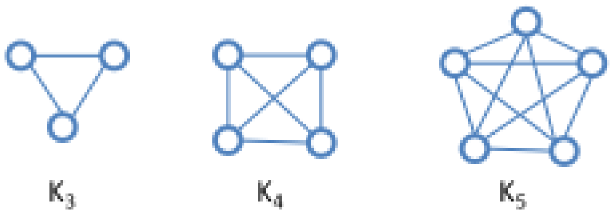
Definition 2.2
Let $G = (V, E)$ be a graph. A Hamiltonian cycle C is a cycle on which every node $\in V$ occurs exactly once. If G has a Hamiltonian cycle it is called Hamiltonian.

Example

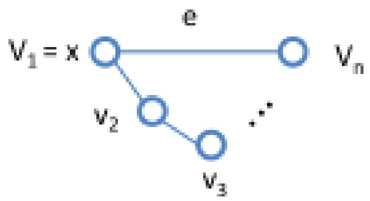
The problem, given an arbitrary undirected graph: Is it Hamiltonian? ⇒ NP complete ⇒ no polinomial time algorithm is known and it is assumed there is no such.
One way out of the complexity issue is to derive conditions that can be tested explicitly and if they are satisfied the desired property is ensured.

Theorem 2.3
Let $G = (V,E)$ be an undirected finite graph without loops and without parallel edges. Let $|V| = n$. If for all $x, y \in V with x \neq y$ and no edge with end points x, y the following holds:
$deg(x) + deg(v) \geq |V| = n$
Then G has a Hamiltonian Cycle.

Example



Proof: Assume there is a graph $G = (V,E)$ with $deg(x) + deg(y) \geq |V|$ for all x and y with $x \neq y$ and no edges between them, but is not Hamiltonian. Among all graphs with nodes in V, we choose one that has this property and has the maximal number of edges, we call graph $G_0 = (V,E_0)$. As the complete graph (every node is connected with every other node) is Hamiltonian, we know there must be an edge e connecting some x and y and $e \in E_0$.
We add edge e to the graph and obtain a new graph $G_1 = (V,E_1)$ that still satisfies the degree conditions and must be Hamiltonian because G_0 was the one with the largest number of edges. We know that the Hamiltonian cycle must contain the edge e.



$v_i \neq v_j fori \neq j$

$S = \{v_i : 1 \leq i \leq n$ x,y are connected with an edge in $E_0\}$
$T = \{v_i : 1 \leq i \leq n$ there is an edge between y and v_i in $E_0\}$

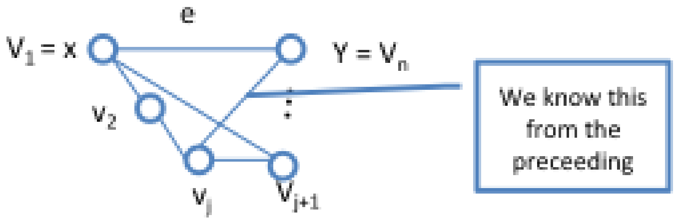
Observation:

i) $y = v_n \in S \cup T$

ii) $|S \cup T| < |V| = n$

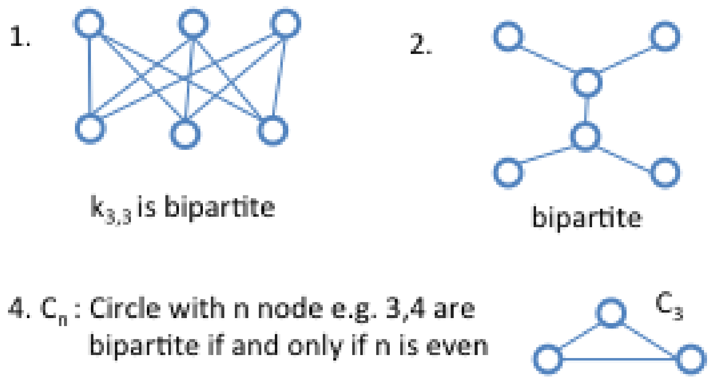
iii) $deg(x) = |S|$
$deg(y) = |T|$

Hence $S \cap T \neq \emptyset, letv_j \in S \cap T$ hence there is an edge between x, v_{j+1} and an edge between y, v_j.
Now remove edge e and there is a Hamiltonian left.
Cost of checking the condition $O(M^2) |E| \leq |V|^2$



2.3 Bipartite Graphs

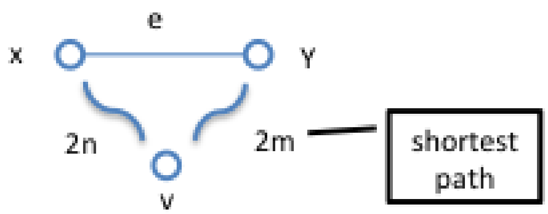
Example



Theorem 2.4
Let $G = (V, E)$ be a connected undirected graph without loops and parallel edges. G is bipartite if and only if it does not contain any circle of odd length.

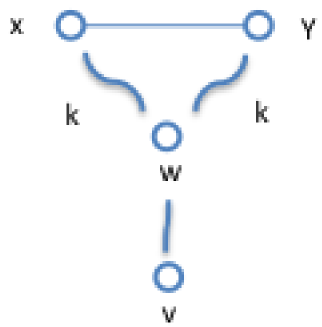
Corollary: All trees are bipartite

Proof: $\Rightarrow$ IF G contains a cicle of odd length then it is not bipartite. $\Leftarrow$ Let G not have any circle of odd length we choose node v.
$V_1 = \{u \in V$ a shortest path between u and v is of odd length$\}$
$V_2 = \{u \in V$ a shortest path between u and v is of even length$\}$
$V \in V_2, V = V_1 \dot\cup V_2$ (disjoint union)
Claim: Ther is no edge e with both endpoints in V_1 respectively V_2. Assume there is an edge e with both endpoints in V_1. Let the end points be x, y



$2m \leq 2n + 1 \text{and} 2n \leq 2m + 1 \Rightarrow m = n$

Let $P(x)$ a shortest path from v to x, analogously $P(y)$ let w be the last node on the paths starting at v that lies on both paths.

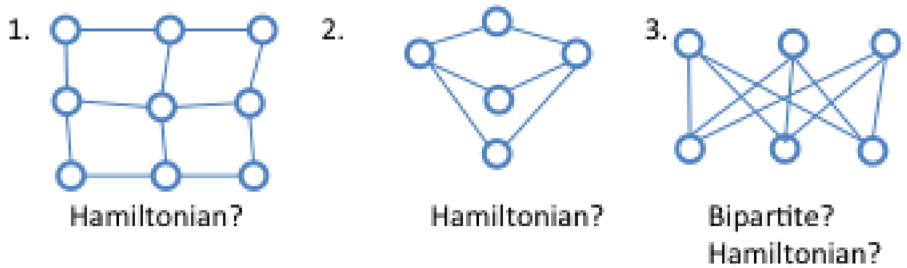


The length of the path from w to x coincides with the length of path from w to y.
The circle w - x - y - w is of odd length i.e. $2k+1 \Rightarrow$ contradiction!

Corollary 2.5: A bipartie graph with an odd number of nodes cannot be Hamiltionian

Proof: Assume if were Hamiltionian then there is a cycle where node appears exactly once. This cycle is of odd length $\Rightarrow$ contradicts Theorem 2.4

Example 2.2



$\Rightarrow$ We have two theorems to check:

i) Count degrees

ii) Corollary 2.5

3
(Network) Flow Problems

TODO

3.1 Network Flow Problems

Example 3.1
Example: Oil field + transportation

Definition 3.1
A network N consists of

i) A finite directed graph $G = (V, E)$ without loops and parallel edges

ii) a function $c : E -> \mathbb{R}^+$, which assigns a capacity to each edge

iii) two designated nodes s and t, called source and sink

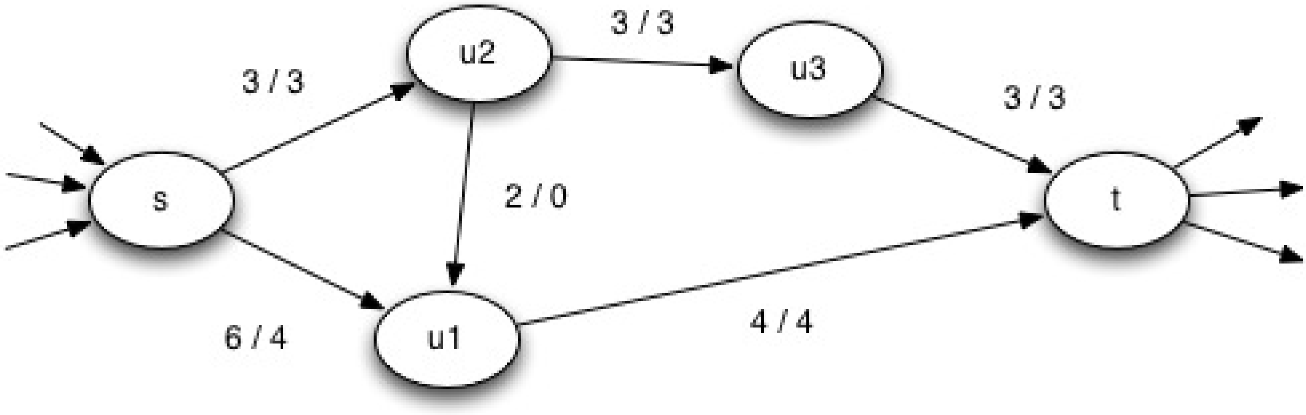
Short: $N = (G, c, \{s, t\})$

Definition 3.2
Let $N = (G, c, \{s, t\})$ be a network. A flow function on N is a function $f : E -> \mathbb{R}$ such that

- $0 \leq f(e) \leq c(e), \forall e \in E$
- $\alpha(v) := \{e :$ endpoint of e is v $\}, v \in V$
  $\beta(v) := \{e :$ startpoint of e is v $\}, v \in V$

For every $v \in V \setminus \{s, t\}$
$\sum_{e \in \alpha(v)} f(e) = \sum_{e \in \beta(v)} f(e)$
This is called "conservation function".
The total flow of the flow function is given by $F = \sum_{e \in \alpha(t)} f(e) - \sum_{e \in \beta(t)} f(e)$

Example 3.2



Notion: 6/4 describes the capacity and the flow of an edge. In this example the capacity of the edge is 6 and the flow is 4.

Problem:
Given an arbitrary network N, find a flow function f, where the total flow F is maximal.

Definition 3.3
Let N = $(G, c, \{s, t\})$ be a network. Let $S \subseteq V$ with s $\in S, t \notin S$
$\bar{S} = V \setminus S$ (i.e. $t \in \bar{S}$)
$E_{S\bar{S}} = \{e$: all edges with starting point in S and endpoint in $\bar{S}\}$
$E_{\bar{S}S} = \{e$: all edges with start point in $\bar{S}$ and end point in S $\}$
$E_{S\bar{S}} \cup E_{\bar{S}S}$ is the cut defined by S.
The capacity of a cut defined by S: $c(S) = \sum_{e \in E_{\bar{S}S}} c(e)$

Lemma 3.1
Let $N = (G, c, \{s, t\})$ be a network, $f : E \to \mathbb{R}$ be a flow function then for any $S \subseteq V$ with $s \in S, t \notin S$:

$$F = \sum_{e \in E_{S\bar{S}}} f(e) - \sum_{e \in E_{\bar{S}S}} f(e)$$

Proof.

$$F = \sum_{e \in \alpha(t)} f(e) - \sum_{e \in \beta(t)} f(e)$$

$$0 = \sum_{e \in \alpha(v)} f(e) - \sum_{e \in \beta(v)} f(e); \forall v \in \bar{S} \setminus \{t\}$$

We add all these equations up. Left hand side: F remains. Right hand side: Let $x \xrightarrow{e} y$ be an edge. We need to consider 4 cases:

i) $x, y \in S$, then the value $f(e)$ does not occur in the summation

ii) $x, y \in \bar{S}$, then $f(e)$ occurs one time positive in the summation, namely for y
f(e) occurs one time negative in the summation, namely for x

iii) $x \in S; y \in \bar{S}$, f(e) occurs positive for y and nowhere else and $e \in E_{S\bar{S}}$

iv) $x \in \bar{S}; y \in S$, then $f(e)$ occurs negative for x and nowhere else and $e \in E_{\bar{S}S}$

This leads to the following equation:

$$F = \sum_{e \in E_{S\bar{S}}} f(e) - \sum_{e \in E_{\bar{S}S}} f(e)$$

Only case 3 are 4 contribute. □

Lemma 3.2
For every flow function f with total flow F and any ser $S \subseteq V$, $s \in S$, $t \notin S$

$$F \leq c(S)$$

Proof. From lemma 3.1 we know

$$F = \sum_{e \in E_{S\bar{S}}} f(e) - \sum_{e \in E_{\bar{S}S}} f(e) \leq \sum_{e \in E_{S\bar{S}}} f(e) \leq \sum_{e \in E_{S\bar{S}}} c(e) = c(S)$$

□

Korollar 3.1 **Max Flow - Min Cut Statement**
If $F = c(S)$ then the total flow F is maximal and the capacity of the cut defined by S is minimal.

Proof. Let $F = c(S)$, consider another flow function f' with total flow F'.

i) $F' \leq c(S)$ (Lemma 3.2) // $F' \leq c(S) = F$
Hence, f is a flow function with maximal total flow.

ii) Let S' with $s \in S'$, $t \notin S'$ be given. $c(S) = F \leq c(S')$. Hence the capacity $c(S)$ is minimal among all other capacities.

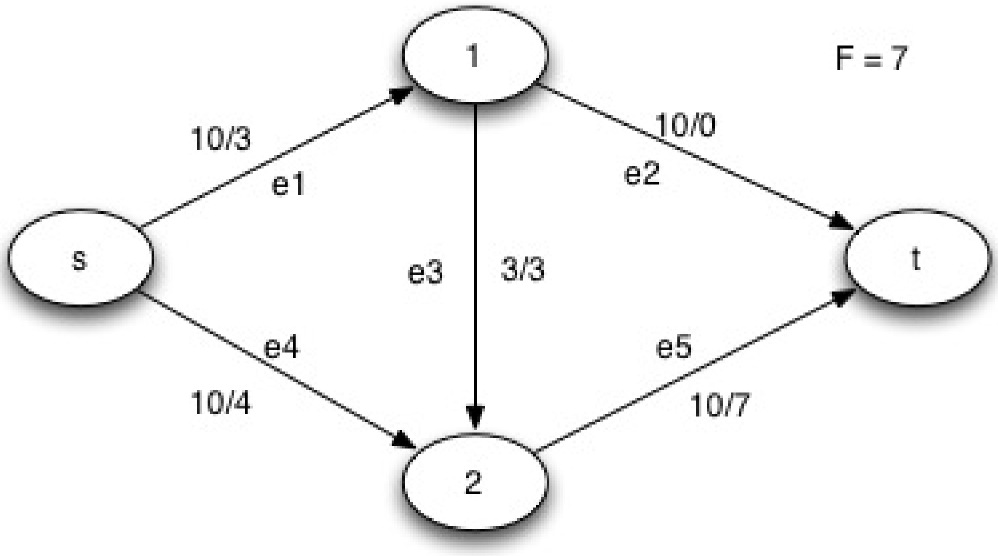
□

An augmenting path is a simple path from s to t, that is not necessarily directed. And for which the following two cases hold: Let e be an edge on this path:

i) $s \to \circ \to \circ \to \ldots \to \underset{s_i}{\circ} \xrightarrow{e} \underset{s_{i+1}}{\circ} \ldots \underset{t}{\circ}$ then we request that $f(e) < c(e)$

ii) $s \to \circ \ldots \underset{s_i}{\circ} \xleftarrow{e} \underset{s_{i+1}}{\circ} \ldots \underset{t}{\circ}$ then we request that $f(e) > 0$

Example 3.3



Which of the following is an augmenting path?

- e_1e_2
- $e_1e_3e_5$
- $e_4e_3e_2$
- e_4e_5

Solution:
The first, third and fourth example are augmenting paths. The second path violates case 1.

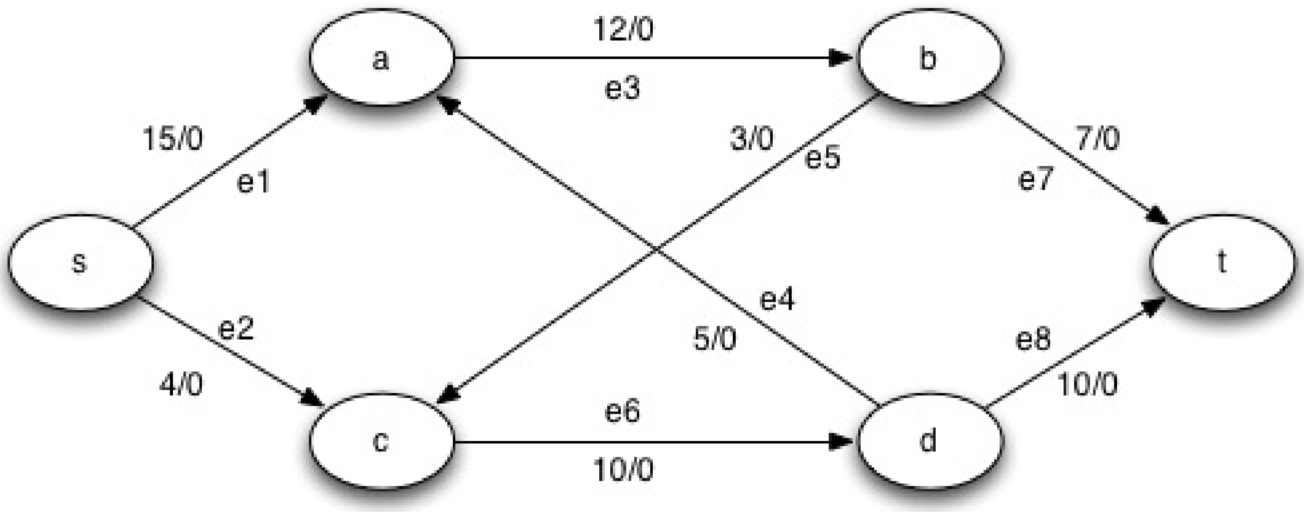
We use $e_4e_3e_2$ to improve the flow function as follows:

- For forward egdes $e : c(e) - f(e)$:
  - $e_4 : 6$
  - $e_2 : 10$
- For backward edges $e : f(e)$
  - $e_3 : 3$

We chose the minimum from the values and add the value to the flow of forward egdes and substract it from backward egdes. The flows of the edges change as follows:

- $e_4 = {}^{10}/_7$
- $e_2 = {}^{10}/_3$
- $e_3 = {}^{3}/_0$

Example 3.4



Augmenting path:
$s* \xrightarrow{e_2} c* \xrightarrow{e_6} d* \xrightarrow{e_4} a* \xrightarrow{e_3} b* \xrightarrow{e_7} t*$
Compute deltas:

- $\Delta_{(e_2)} = 4$
- $\Delta_{(e_6)} = 10$
- $\Delta_{(e_4)} = 5$
- $\Delta_{(e_3)} = 12$
- $\Delta_{(e_7)} = 7$

The minimum Δ is 4, so the flow of the edges will be increased by 4.

- $e_2 = {}^{4}/_4$
- $e_6 = {}^{10}/_4$
- $e_4 = {}^{5}/_4$
- $e_3 = {}^{12}/_4$
- $e_7 = {}^{7}/_4$

The next steps or paths would be:

- $s \to a \to b \to c \to d \to t$
- $s \to a \to b \to t$
- $s \to a \to d \to t$

The application of this paths leads to a new flow: $F = 14$.

Lemma 3.3
When executing a step in the algorithm, the actual function f is a flow function.

Proof. The assumption is obviously true for step 1 because $f \equiv 0$ is a flow function. It is obviously true for steps 2, 3 and 5, too, because f is not modified.
Step 4:
Let f be a flow function when we enter step 4. We have to show that after performing step 4, the newly calculated function f is still a flow function.
Let f_{old} be the function with which we enter step 4 and f_{new} the newly calculated one. f_{old} is a flow function. Hence,

$$\sum_{e \in \alpha(v)} f_{old}(e) = \sum_{e \in \beta(v)} f_{old}(e); \forall v : v \neq s, v \neq t$$

Let $s \to v_0 \to v_1 ... v_{f_{e-1}} \to v_{f_e} \to t$ be an augmenting path used in step 4. By definition of $\Delta f_{new}(e) < c(e)$ and $f_{new}(e) > 0$.
For step 4: Let $s = v_0 \to ... \to v_2 = t$ be the path along which we achieved the marking. Only the flow value of the edges along this path is modified, so we have to check only the edges respectively nodes along this path. We have to check:

i) $0 \leq f_{new}(e) \leq c(e) \forall e$: e edge on the path

ii) $\sum_{e \in \alpha(v)} f_{new}(e) = \sum_{e \in \beta(v)} f_{new}(e), \forall v$, v on the path, $v \neq s, v \neg t$

The check:

i) 1 holds by the definition of Δ

ii) Let $v_i, v_i \neq s, v_i \neq t$ be a node on the path:

   a) $\xrightarrow{e_i} v_i \xrightarrow{e_{i+1}}, e_i \in \alpha(v_i), e_{i+1} \in \beta(v_i)$ for both edges f_{new} is obtained from f_{old} by adding Δ, so 2 holds in this case

   b) $\xrightarrow{e_i} v_i \xleftarrow{e_{i+1}}, e_i, e_{i+1} \in \alpha(v_i)$. One contributes Δ, the other contributes $-\Delta$, so 2 holds for v_i

   c) $\leftarrow v_i \leftarrow$ analogously

   d) $\leftarrow v_i \to$ analogously, too

□