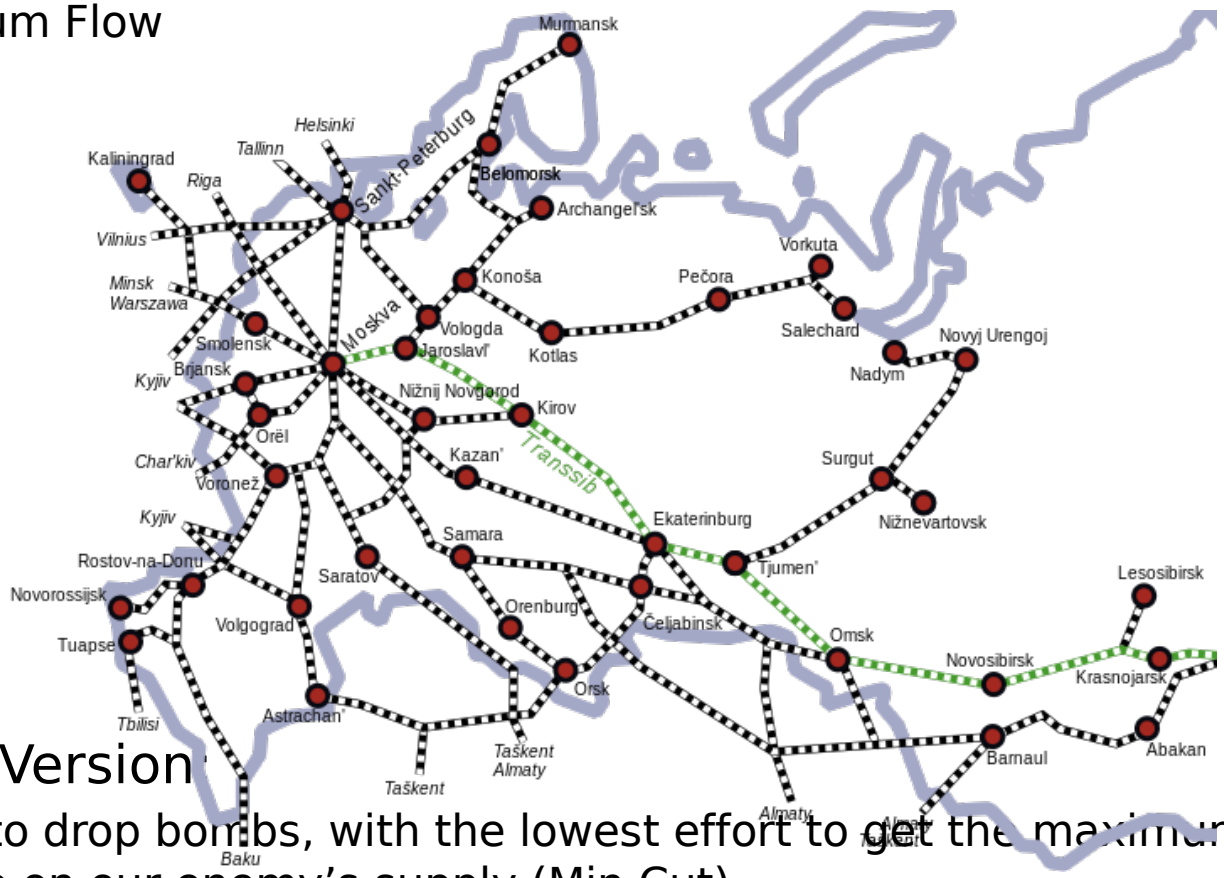
An O(n log n) Algorithm for Maximum st-Flow in a Directed Planar Graph

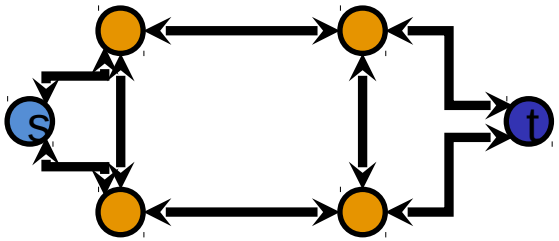
By Oliver Kotulski

- First official Version 1950:
  - Given two 2 cities, intermediate cities, Links with capacity ⇒ Calculate maximum Flow



- Inofficial Version
  - Where to drop bombs, with the lowest effort to get the maximum damage on our enemy's supply (Min Cut)

- St – planar graphs

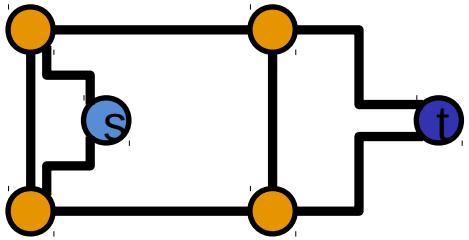


Sink

Source

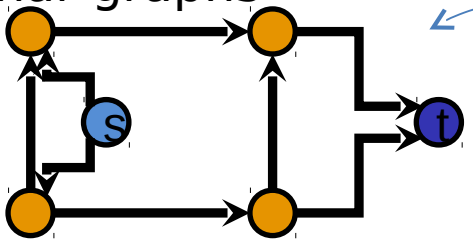
Time: $O(n)$

- Undirected planar graphs



Time: $O(n \log n)$

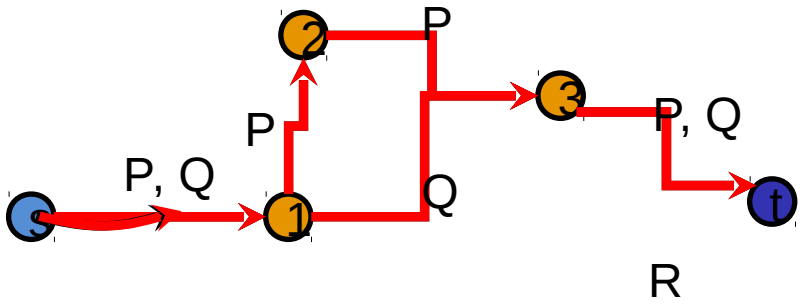
- Directed planar graphs



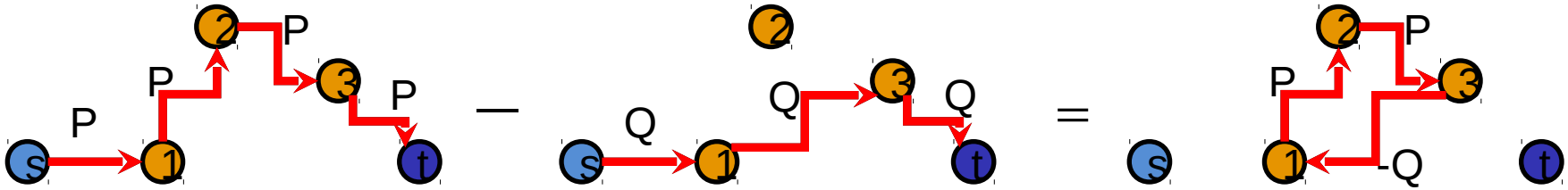
Dart

Time: $O(n \log n)$

- Leftmost Path:
  - Algorithm searches for the leftmost s-t-path



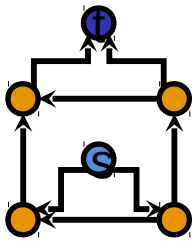
  - Intuitively P is the leftmost path
  - Substract the Paths: $\delta(P) - \delta(Q)$



  - If the resulting graph is clockwise: P is left of Q, else Q is left of P

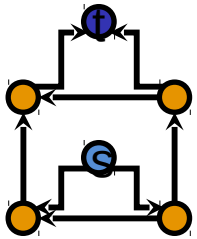
- Developed by Borradile & Klein 2006
- Consists of:

I. Preprocessing I:
   - Choose a face f_{root} incident to sink t



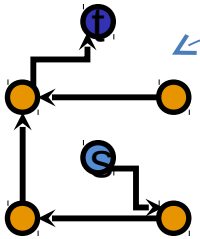
II. Preprocessing II: Leftmost Circulation

f_{root}, G:



f_{root}

G_0:

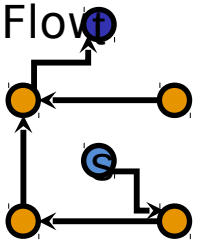


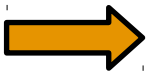
New capacities

, x

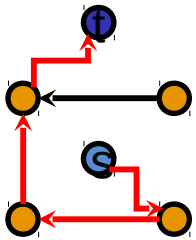
III. Leftmost Flow

G_0:



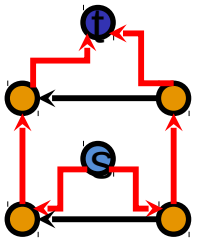


G_0:

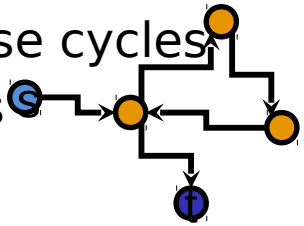


x

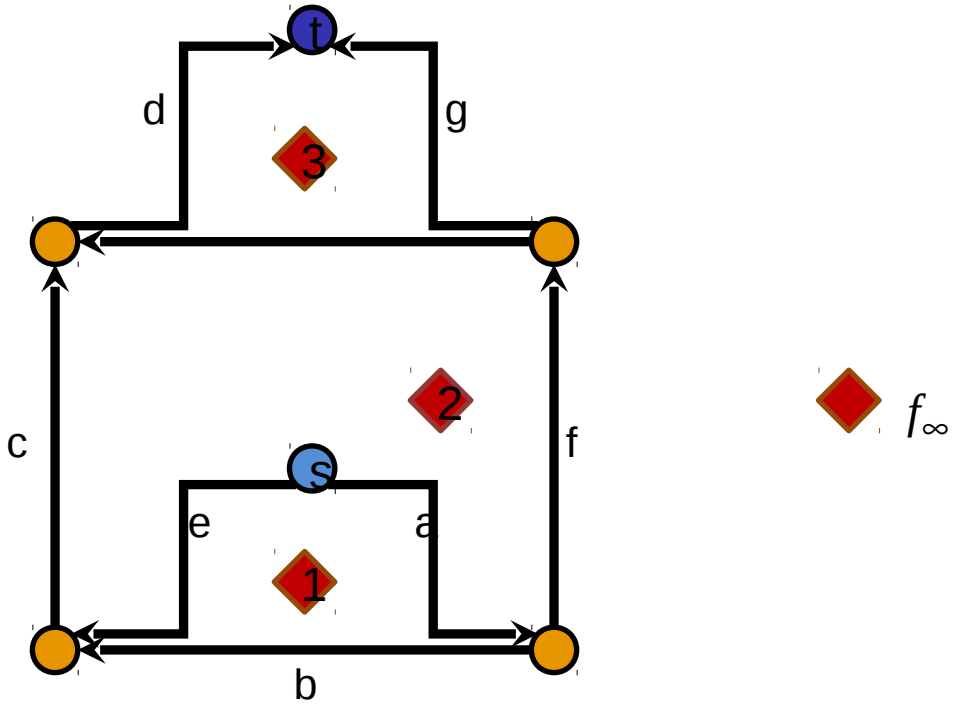
G:



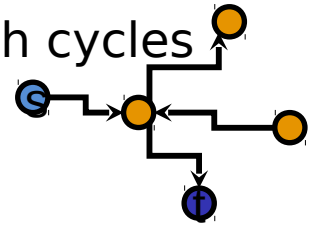
- Why?
  - Leftmost Flow Algorithm: a graph with no clockwise cycles
  - Leftmost Circulation: Saturate all clockwise cycles

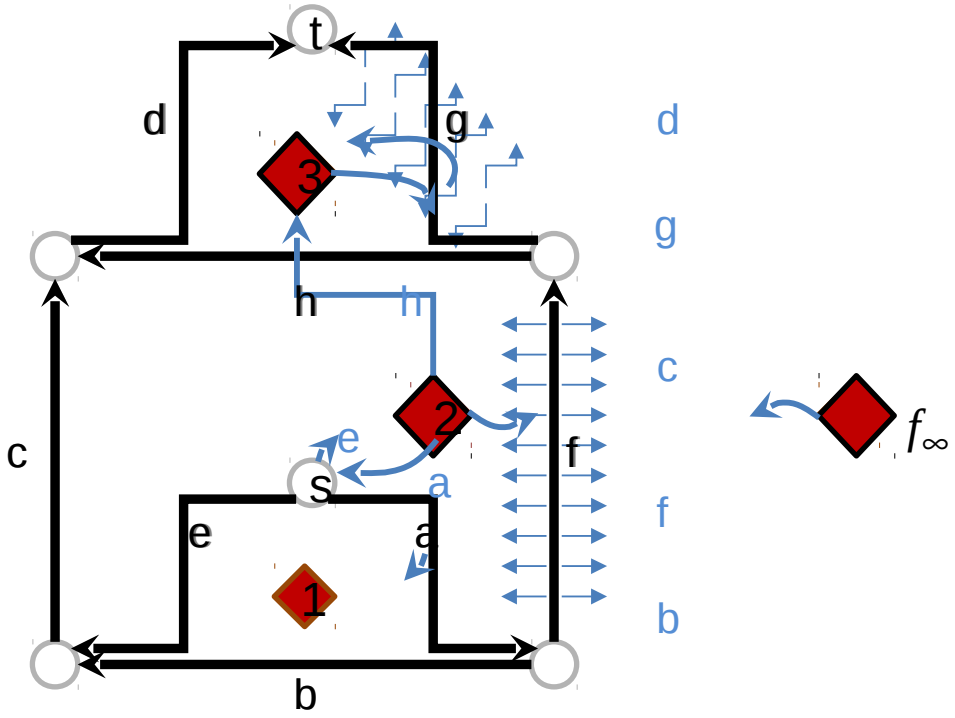


- Input: $G_{f_{root}}$
- Calculate the dual Graph

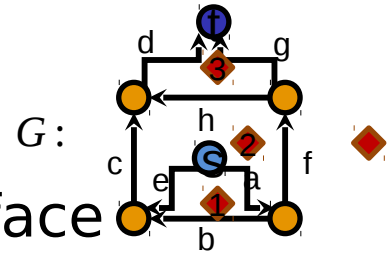


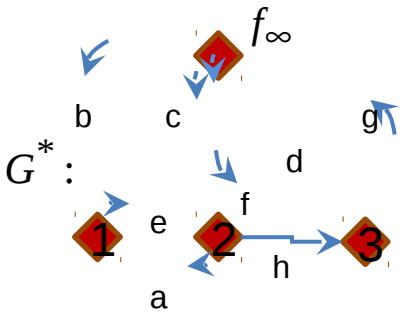
- Why?
  - Leftmost Flow Algorithm can not take a graph with cycles
  - Leftmost Circulation: Saturate all cycles
- Input: $G_{f_{root}}$
- Calculate the dual Graph



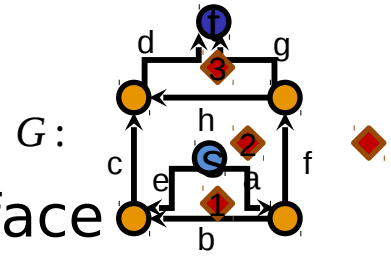


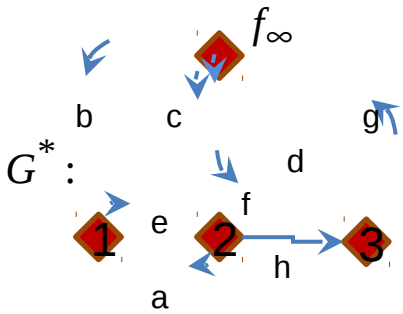
- Input: G, f_{root}
- Calculate the dual Graph G^*
- Assign potential to each face
  - Calculate shortest path (SSSP) in G^*
  - From each $v \in V^*$ to f_{root}
  - Where capacity = distance

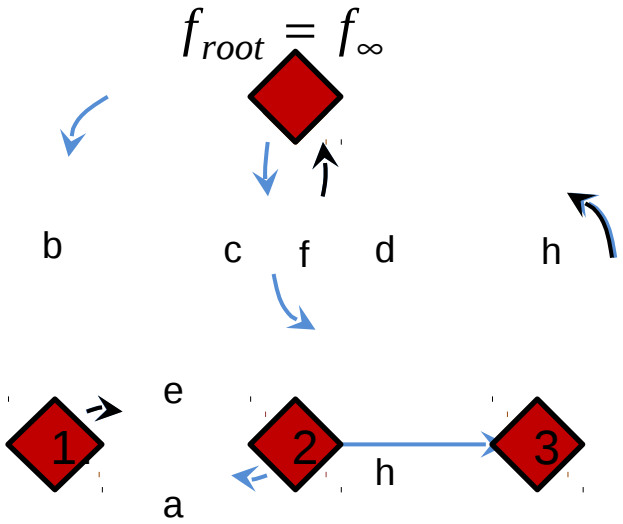




- Input: G, f_{root}
- Calculate the dual Graph G^*
- Assign potential to each face
  - Calculate shortest path (SSSP) in G^*
  - From each $v \in V^*$ to f_{root}
  - Where capacity = distance

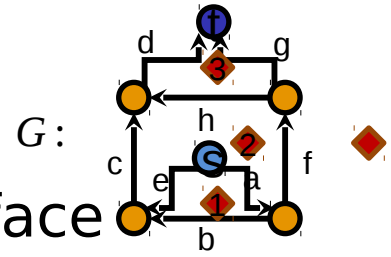


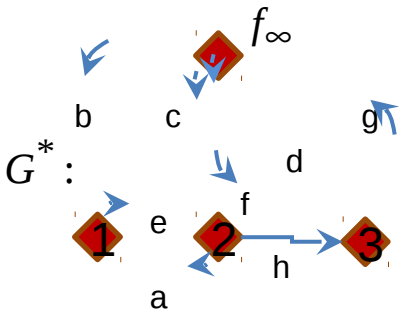




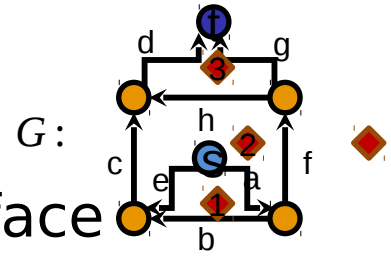
$$\Phi(f) = \begin{pmatrix} f_\infty = \mathbf{0} \\ f_1 = 2 \\ f_2 = 1 \\ f_3 = 1 \end{pmatrix}$$

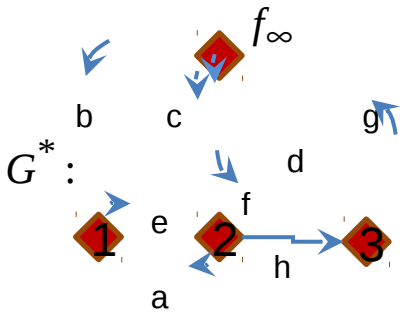
- Input: G, f_{root}
- Calculate the dual Graph G^*
- Assign potential Φ to each face
- Calculate Leftmost Circulation η
  - For each dart $u \in E^*$: $\eta = \Phi(tail_{G^*}(u)) - \Phi(head_{G^*}(u))$





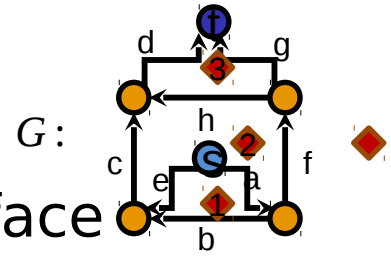
- Input: G, f_{root}
- Calculate the dual Graph G^*
- Assign potential Φ to each face
- Calculate Leftmost Circulation η
  - For each dart $u \in E^*$: $\eta = \Phi(tail_{G^*}(u)) - \Phi(head_{G^*}(u))$

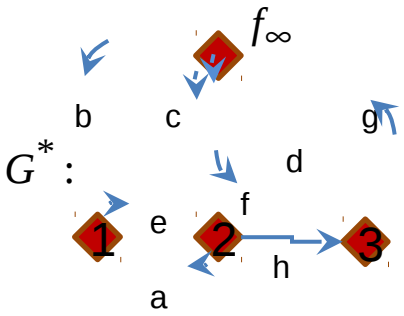




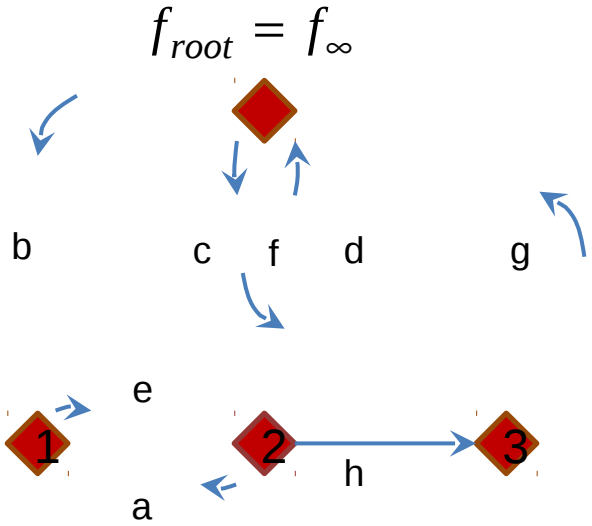
$$\Phi(f) = \begin{pmatrix} f_\infty = 0 \\ f_1 = 2 \\ f_2 = 1 \\ f_3 = 1 \end{pmatrix}$$

- Input: G, f_{root}
- Calculate the dual Graph G^*
- Assign potential Φ to each face
- Calculate Leftmost Circulation η
  - For each dart $u \in E^* : \eta = \Phi(tail_{G^*}(u)) - \Phi(head_{G^*}(u))$

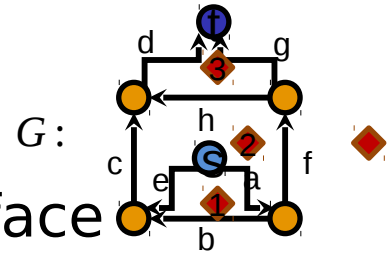


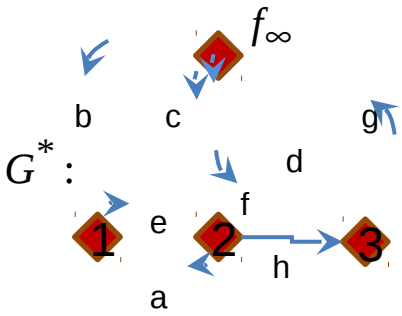


$$\Phi(f) = \begin{pmatrix} f_\infty = 0 \\ f_1 = 2 \\ f_2 = 1 \\ f_3 = 1 \end{pmatrix}$$

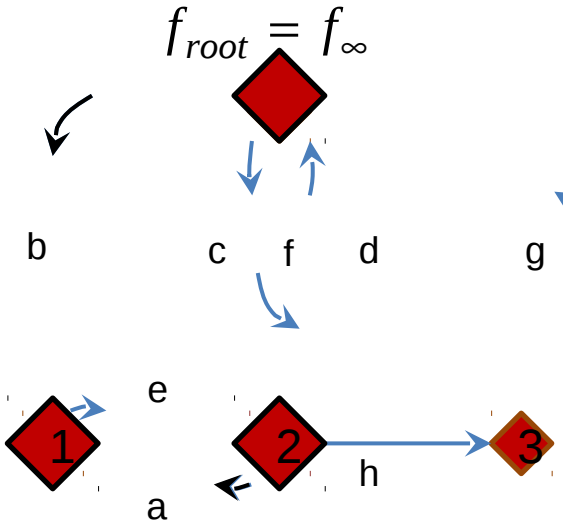


- Input: G, f_{root}
- Calculate the dual Graph G^*
- Assign potential Φ to each face
- Calculate Leftmost Circulation η
  - For each dart $u \in E^* : \eta = \Phi(tail_{G^*}(u)) - \Phi(head_{G^*}(u))$





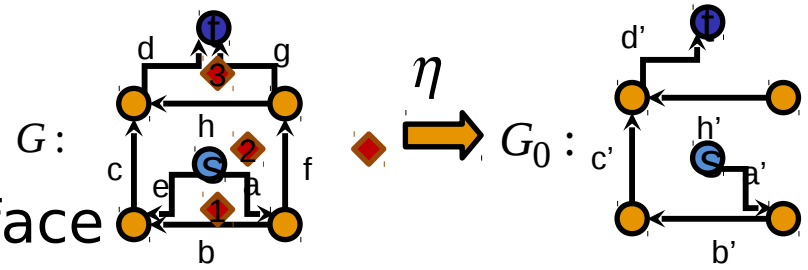
$$\Phi(f) = \begin{pmatrix} f_\infty = 0 \\ f_1 = 2 \\ f_2 = 1 \\ f_3 = 1 \end{pmatrix}$$



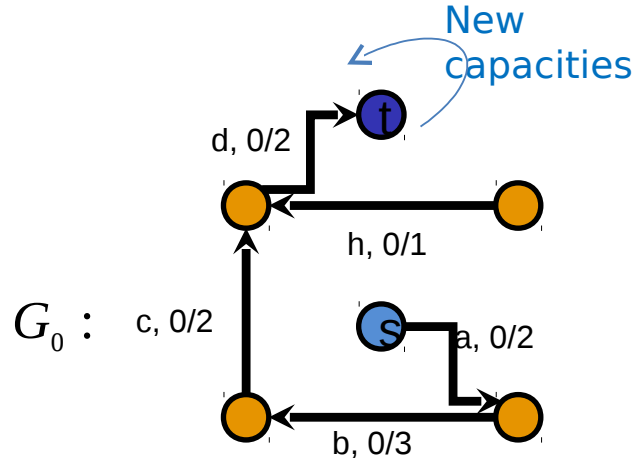
$$\eta = \begin{pmatrix} a = \Phi(f_2) - \Phi(f_1) = 1 - 2 = -1 \\ b = \Phi(f_\infty) - \Phi(f_1) = 0 - 2 = -2 \\ c = -1 \\ d = -1 \\ e = 1 \\ f = 1 \\ g = 1 \\ h = 0 \end{pmatrix}$$

- Input: $G, f, root$
- Calculate the dual Graph G^*
- Assign potential η to each face
- Calculate Leftmost Circulation η
- Calculate the residual capacities c_0
  - $\forall u \in E : c_0(u) = c(u) - \eta(u)$
  - residual, if $c_0(u) \neq 0$

original capacity

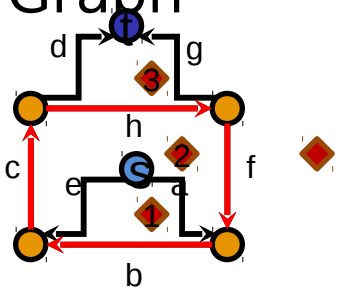


$$c_0 = c - \eta = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} - \begin{pmatrix} -1 \\ -2 \\ -1 \\ -1 \\ 1 \\ 1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} a & = 2 \\ b & = 3 \\ c & = 2 \\ d & = 2 \\ e & = 0 \\ f & = 0 \\ g & = 0 \\ h & = 1 \end{pmatrix}$$

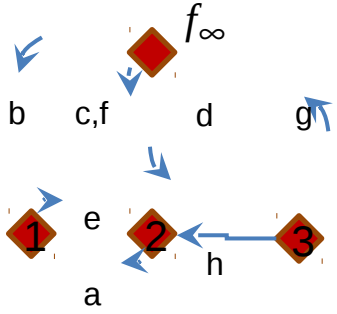


- Reminder: Leftmost Circulation to saturate clockwise cycles G'
- Take Graph

G':



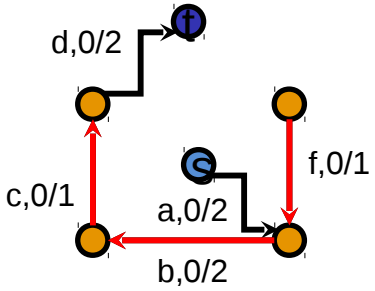
G'^{*}:



$$\Phi(f) = \begin{pmatrix} f_\infty = 0 \\ f_1 = 1 \\ f_2 = 0 \\ f_3 = 1 \end{pmatrix}$$

$$\eta = \begin{pmatrix} a = -1 \\ b = -1 \\ c = 0 \\ d = -1 \\ e = 1 \\ f = 0 \\ g = 1 \\ h = 1 \end{pmatrix} \qquad c_0 = c - \eta = \begin{pmatrix} a = 2 \\ b = 2 \\ c = 1 \\ d = 2 \\ e = 0 \\ f = 1 \\ g = 0 \\ h = 0 \end{pmatrix}$$

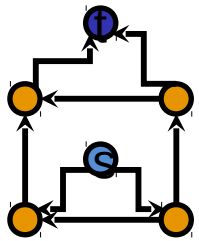
G_0':



- Saturation proven by: Khuller et al. 1993

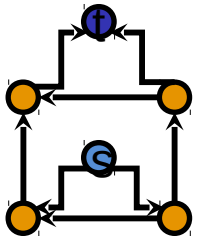
- Developed by Borradile & Klein 2006
- Consists of:

I. Preprocessing I:
   - Choose a face f_{root} incident to sink t

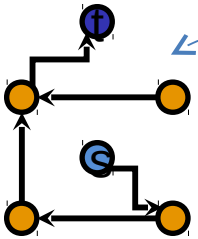


II. Leftmost Circulation η

$f_{root}, G:$



$G_0:$



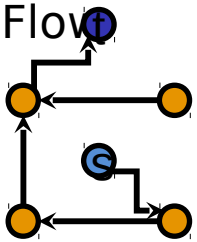
New capacities

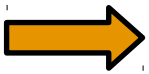
$, \eta$

Flow assignment for G

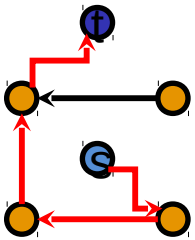
III. Leftmost Flow

$\eta, G_0:$



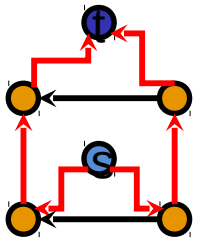


$G_0:$



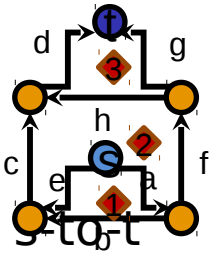
η

$G:$

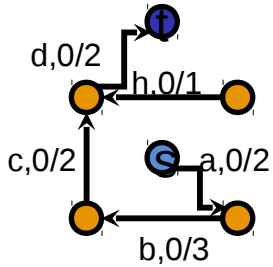


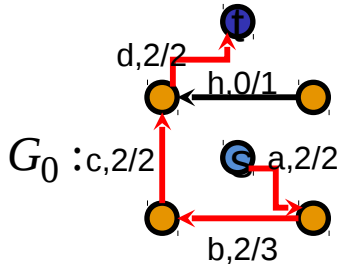
- Input: G_0, c_0, s, t, f_∞
  - Initialize flow **f**=0
  - While there is a residual s-to-t path, saturate the leftmost s-to-t path
  - Return **f**

G:



G_0 :

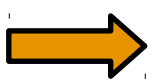




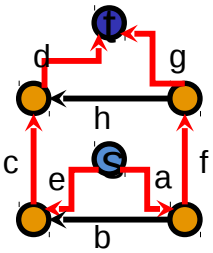
f=

$$\begin{pmatrix} a=2 \\ b=2 \\ c=2 \\ d=2 \\ e=0 \\ f=0 \\ g=0 \\ h=0 \end{pmatrix}$$

$$f_{\max} = \eta + \mathbf{f} = \begin{pmatrix} -1 \\ -2 \\ -1 \\ -1 \\ 1 \\ 1 \\ 1 \\ 0 \end{pmatrix} + \begin{pmatrix} 2 \\ 2 \\ 2 \\ 2 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} a=1 \\ b=0 \\ c=1 \\ d=1 \\ e=1 \\ f=1 \\ g=1 \\ h=0 \end{pmatrix}$$

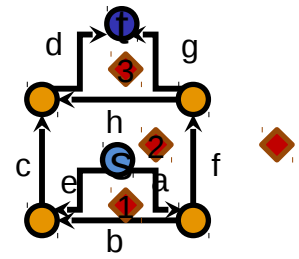


G:

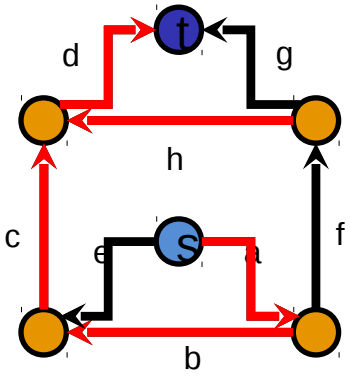


- Input: G, c_0, s, t, f_∞
  - Initialize flow f=0
  - Initialize T to be the right first search tree beginning from t
  - Initialize T* to consist of the edges of the dual G* which are not in T
  - Repeat FlowCalculation();

G:



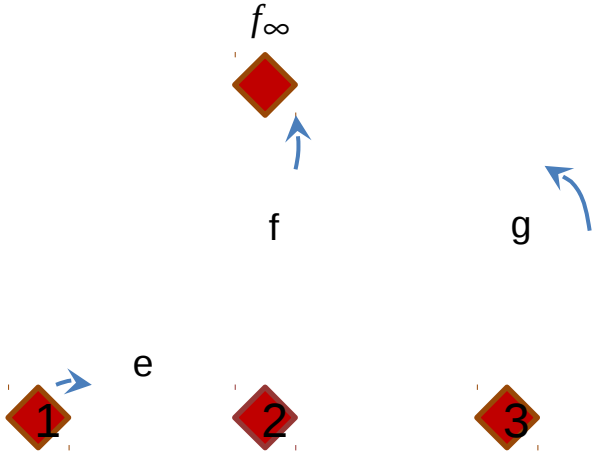
G:



f=

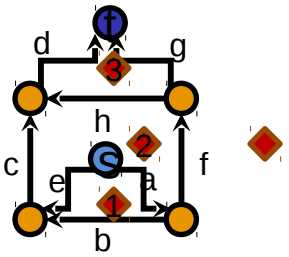
$$\begin{pmatrix} a=0 \\ b=0 \\ c=0 \\ d=0 \\ e=0 \\ f=0 \\ g=0 \\ h=0 \end{pmatrix}$$

T^*:

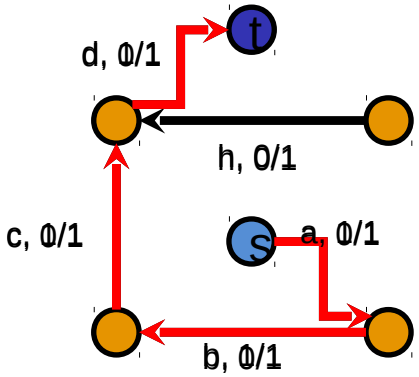


- Repeat:
  - If T[s] is residual, saturate T[s], modifying f

G:



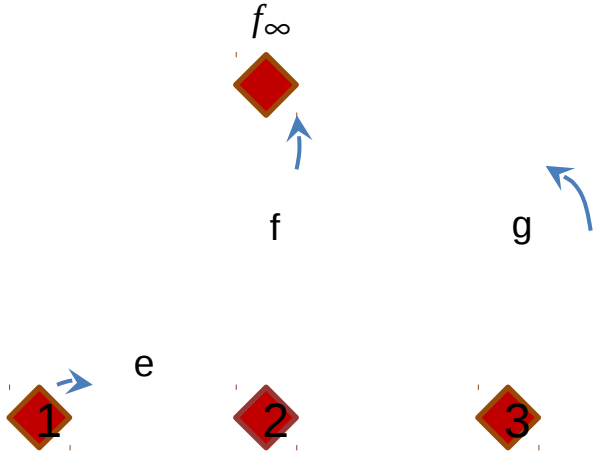
T:



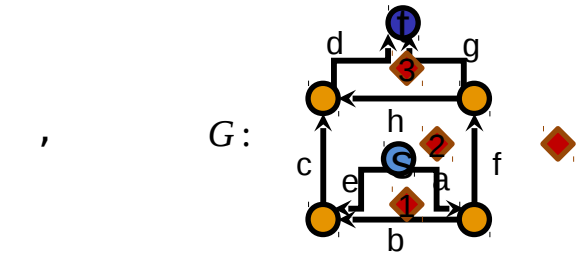
f=

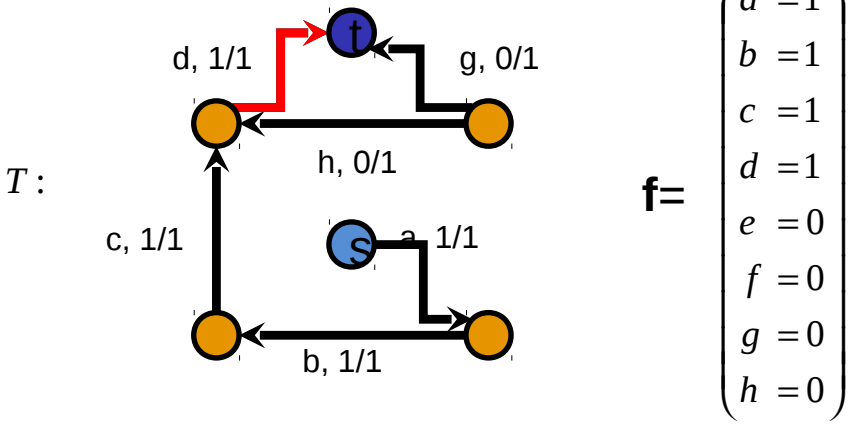
$$\begin{pmatrix} a = 0 \\ b = 0 \\ c = 0 \\ d = 0 \\ e = 0 \\ f = 0 \\ g = 0 \\ h = 0 \end{pmatrix}$$

T^*:



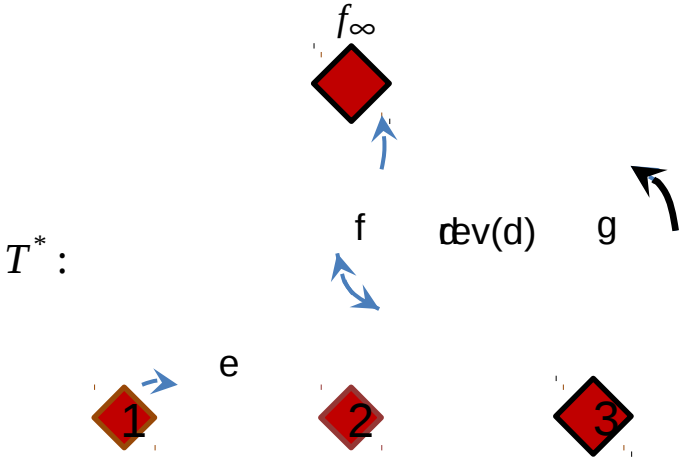
- Repeat:
  - If T[s] is residual, saturate T[s], modifying **f**
  - Let u be the last non-residual dart in T[s]
  - If $tail_{G^*}(u)$ is a descendent in T* of $head_{G^*}(u)$ return f
  - Let *v* be the parent dart in T* of $head_{G^*}(u)$
  - Eject *v* from T* and insert $rev(u)$ into T*
  - Eject *u* from T and insert *v* into T



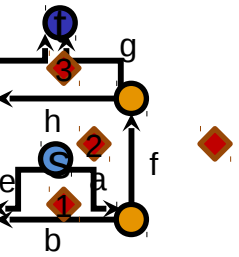


f=

$$\begin{pmatrix} a=1 \\ b=1 \\ c=1 \\ d=1 \\ e=0 \\ f=0 \\ g=0 \\ h=0 \end{pmatrix}$$

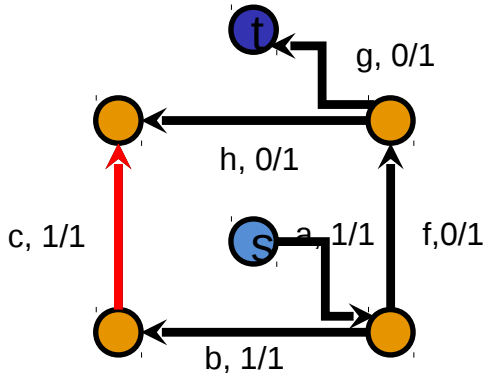


- Repeat:
  - If T[s] is residual, saturate T[s], modifying f
  - Let u be the last non-residual dart in T[s]
  - If $tail_{G^*}(u)$ is a descendent in T* of $head_{G^*}(u)$ return f
  - Let *v* be the parent dart in T* of $head_{G^*}(u)$
  - Eject *v* from T* and insert $rev(u)$ into T*
  - Eject *u* from T and insert *v* into T



G:

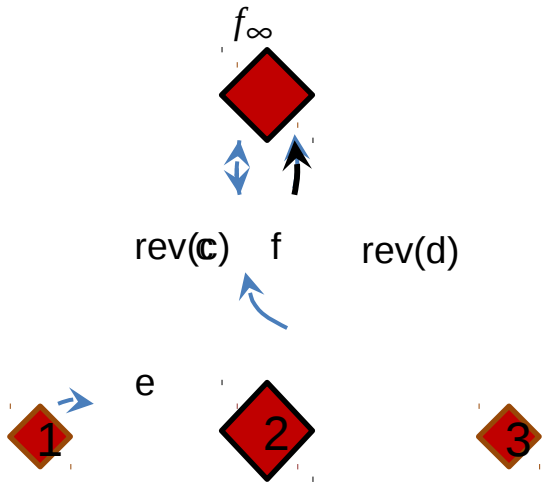
T:



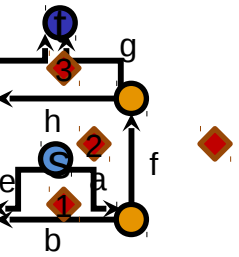
f=

$$\begin{pmatrix} a=1 \\ b=1 \\ c=1 \\ d=1 \\ e=0 \\ f=0 \\ g=0 \\ h=0 \end{pmatrix}$$

T^*:

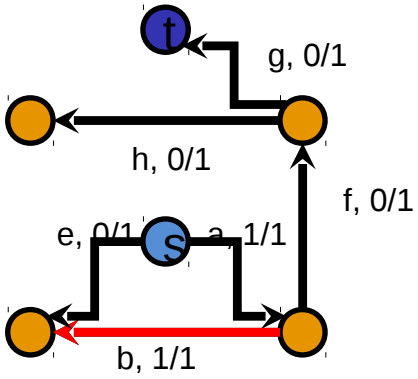


- Repeat:
  - If T[s] is residual, saturate T[s], modifying f
  - Let u be the last non-residual dart in T[s]
  - If $tail_{G^*}(u)$ is a descendent in T* of $head_{G^*}(u)$ return f
  - Let *v* be the parent dart in T* of $head_{G^*}(u)$
  - Eject *v* from T* and insert $rev(u)$ into T*
  - Eject *u* from T and insert *v* into T



G_0 :

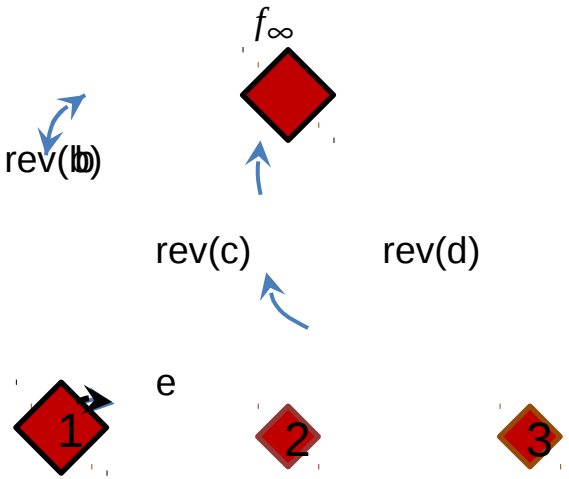
T :



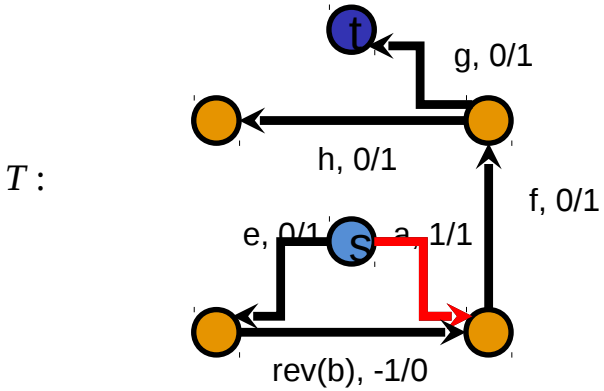
f=

$$\begin{pmatrix} a=1 \\ b=1 \\ c=1 \\ d=1 \\ e=0 \\ f=0 \\ g=0 \\ h=0 \end{pmatrix}$$

T^* :

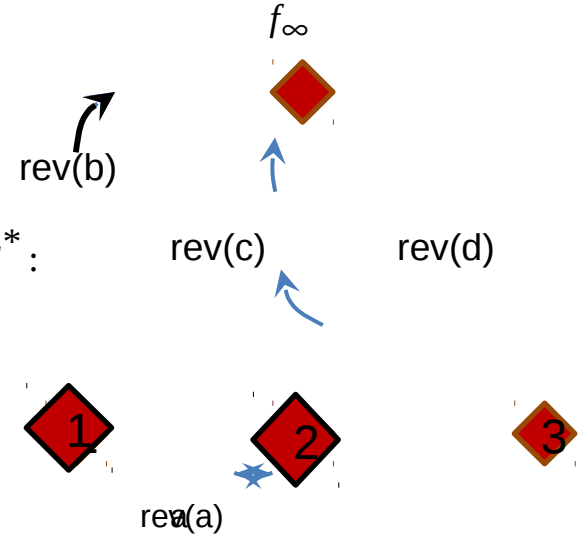


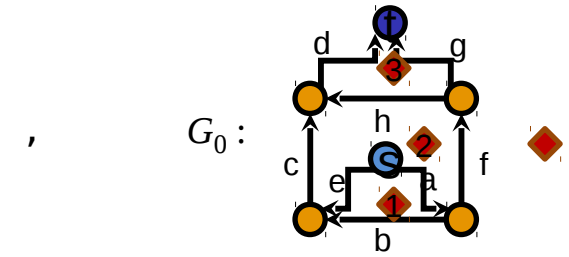
- Repeat:
  - If T[s] is residual, saturate T[s], modifying f
  - Let u be the last non-residual dart in T[s]
  - If $tail_{G^*}(u)$ is a descendent in T* of $head_{G^*}(u)$ return f
  - Let *v* be the parent dart in T* of $head_{G^*}(u)$
  - Eject *v* from T* and insert $rev(u)$ into T*
  - Eject *u* from T and insert *v* into T



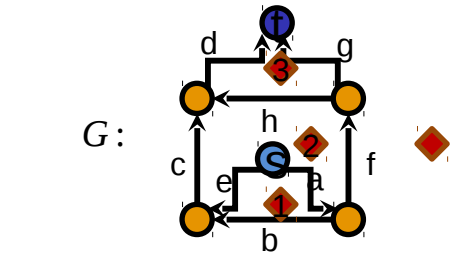
f=

$$\begin{pmatrix} a=1 \\ b=1 \\ c=1 \\ d=1 \\ e=0 \\ f=0 \\ g=0 \\ h=0 \end{pmatrix}$$



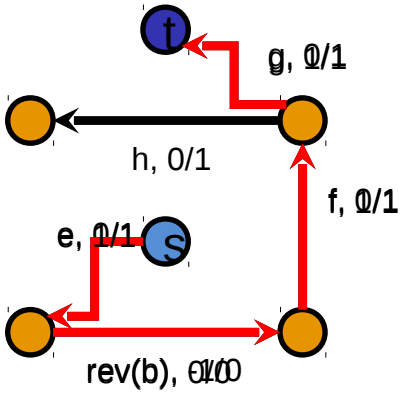


- Repeat:
  - If T[s] is residual, saturate T[s], modifying f
  - Let u be the last non-residual dart in T[s]
  - If $tail_{G^*}(u)$ is a descendent in T* of $head_{G^*}(u)$ return f
  - Let *v* be the parent dart in T* of $head_{G^*}(u)$
  - Eject *v* from T* and insert $rev(u)$ into T*
  - Eject *u* from T and insert *v* into T

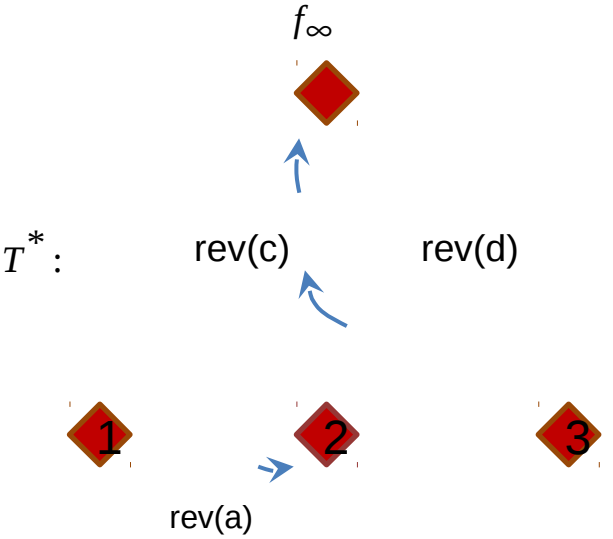


G:

T:

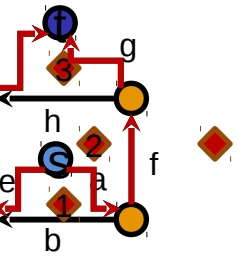


$$f = \begin{pmatrix} a = 1 \\ b = 0 \\ c = 1 \\ d = 1 \\ e = 0 \\ f = 0 \\ g = 0 \\ h = 0 \end{pmatrix}$$



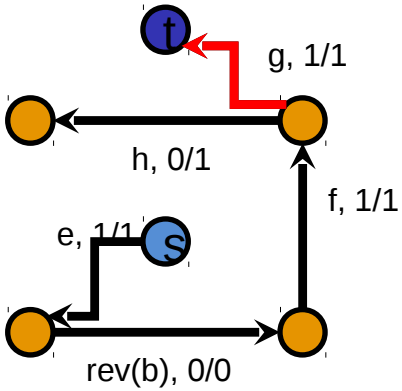
- Repeat:
  - If T[s] is residual, saturate T[s], modifying f
  - Let u be the last non-residual dart in T[s]
  - If $tail_{G^*}(u)$ is a descendent in T* of $head_{G^*}(u)$ return f
  - Let *v* be the parent dart in T* of $head_{G^*}(u)$
  - Eject *v* from T* and insert $rev(u)$ into T*
  - Eject *u* from T and insert *v* into T

ü



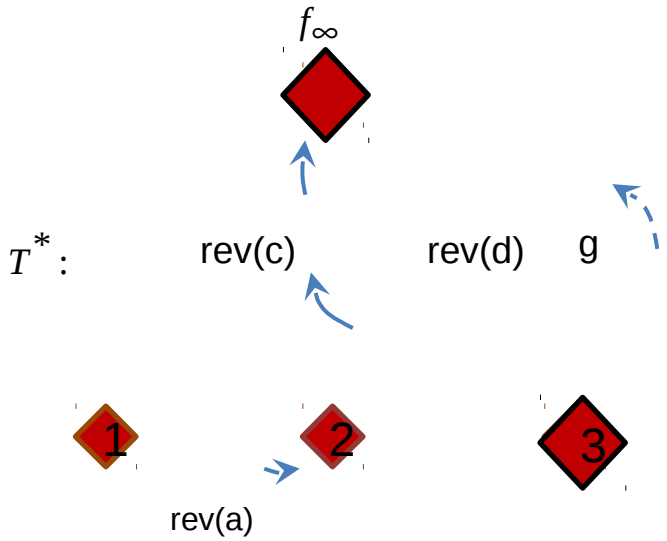
G :

T :



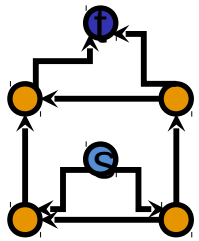
f=

$$\begin{pmatrix} a=1 \\ b=0 \\ c=1 \\ d=1 \\ e=1 \\ f=1 \\ g=1 \\ h=0 \end{pmatrix}$$

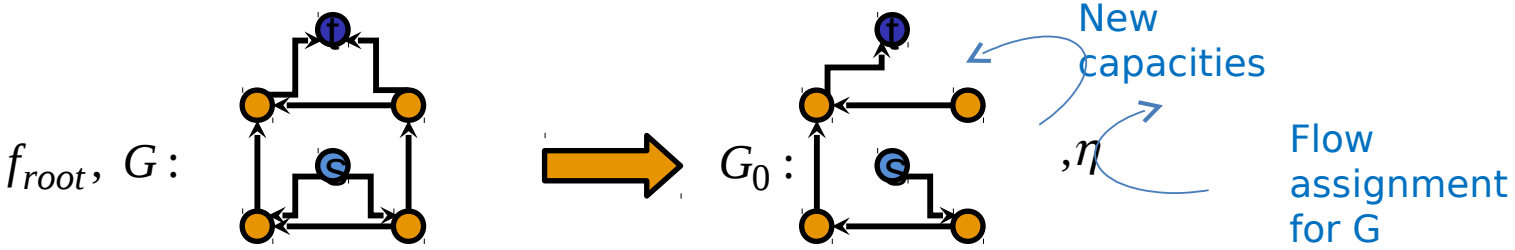


- Developed by Borradile & Klein 2006
- Consists of:

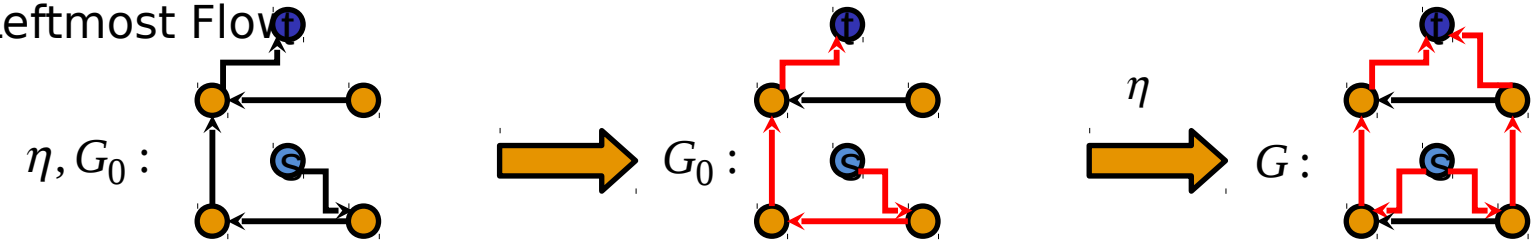
I. Preprocessing I:
   - Choose a face f_{root} incident to sink t



II. Leftmost Circulation η



III. Leftmost Flow



- Unusability Theorem:
  - Each dart can be inserted only once
  - *O(m)* Iterations

$$\Rightarrow O(3m) = O(m)$$

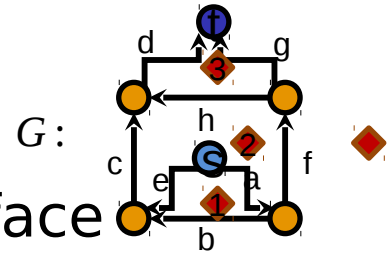
- Complexity
  - Initialization
    - T : dynamic tree data structure
    - T*: Euler-tour tree data structure
  - Loop
    - if T[s] is residual, saturate T[s], modifying f
    - Let u be the last non-residual dart in T[s]
    - If $tail_{G^*}(u)$ is a descendent in T* of $head_{G^*}(u)$, return f
    - Let *v* be the parent dart in T* of $head_{G^*}(u)$
    - Eject *v* from T* and insert $rev(u)$ into T*
    - Eject *u* from T and insert *v* into T

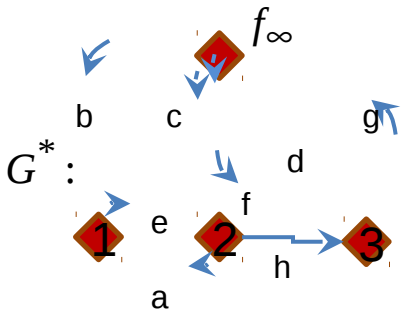
$$\Rightarrow O(\log n)$$

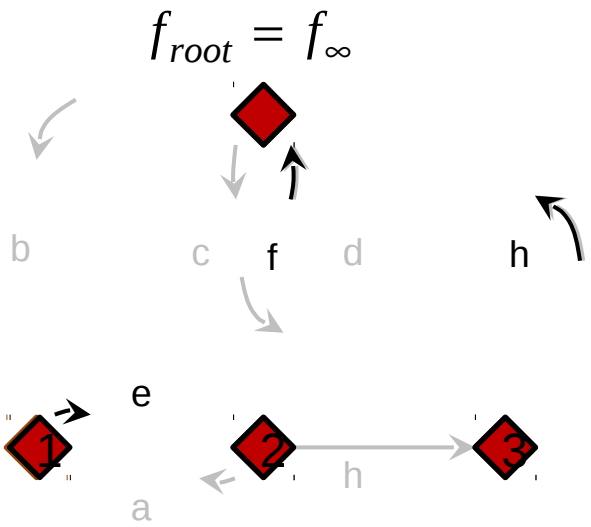
$$\Rightarrow O(n \log n)$$

Thank you for your attention

- Input: G, f_{root}
- Calculate the dual Graph G^*
- Assign potential to each face
  - Calculate shortest path (SSSP) in G^*
  - From each $v \in V^*$ to f_{root}
  - Where capacity = distance



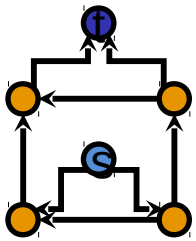




$$\Phi(f) = \begin{pmatrix} f_\infty = \mathbf{0} \\ f_1 = 2 \\ f_2 = 1 \\ f_3 = 1 \end{pmatrix}$$

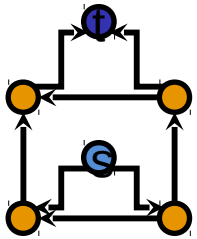
- Developed by Borradile & Klein 2006
- Consists of:

I. Preprocessing I:
   - Choose a face f_{root} incident to sink t



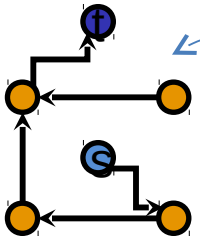
II. Preprocessing II: Leftmost Circulation

f_{root}, G:



f_{root}

G_0:

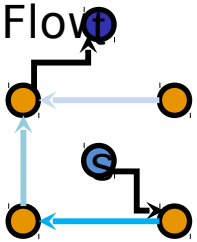


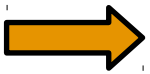
New capacities

, x

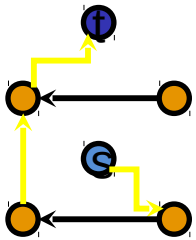
III. Leftmost Flow

G_0:



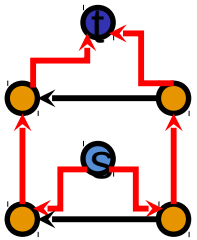


G_0:



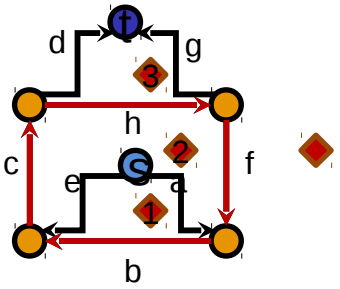
x

G:

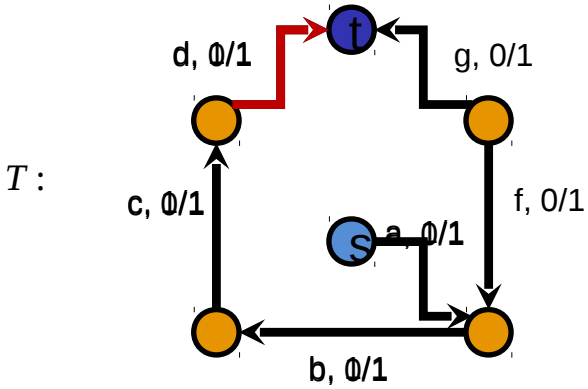


- Cyclic Graph G':
- Repeat:
  - If T[s] is residual, saturate T[s], modifying f
  - Let u be the last non-residual dart in in T[s]
  - If $tail_{G^*}(u)$ is a descendent in T^* of $head_{G^*}(u)$ return f
  - Let *v* be the parent dart in T^* of $head_{G^*}(u)$
  - Eject *v* from T^* and insert $rev(u)$ into T^*
  - Eject *u* from T^* and insert *v* into T^*

G' :

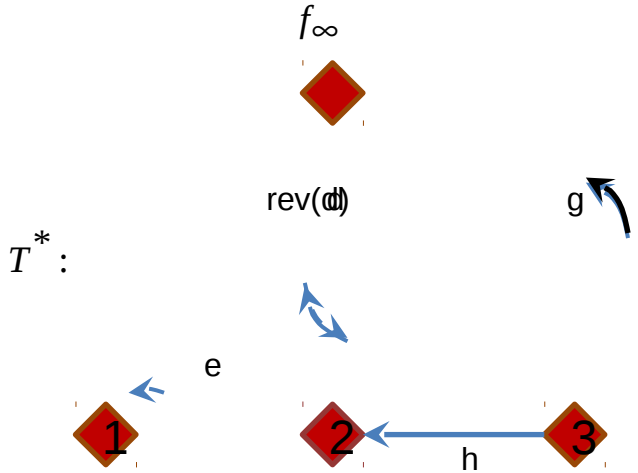


,

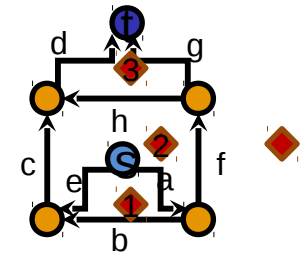


f=

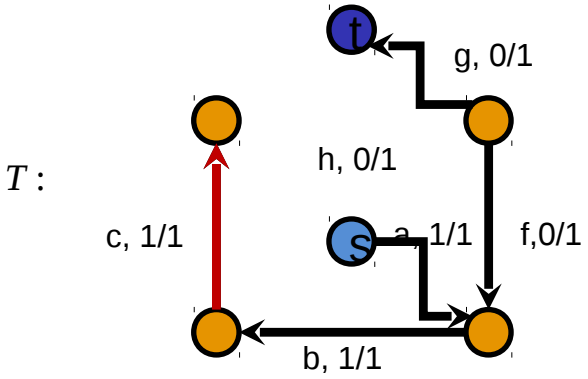
$$\begin{pmatrix} a=1 \\ b=1 \\ c=1 \\ d=1 \\ e=0 \\ f=0 \\ g=0 \\ h=0 \end{pmatrix}$$



- Repeat:
  - If T[s] is residual, saturate T[s], modifying f
  - Let u be the last non-residual dart in in T[s]
  - If $tail_{G^*}(u)$ is a descendent in T* of $head_{G^*}(u)$ return f
  - Let *v* be the parent dart in T* of $head_{G^*}(u)$
  - Eject *v* from T* and insert $rev(u)$ into T*
  - Eject *u* from T and insert *v* into T



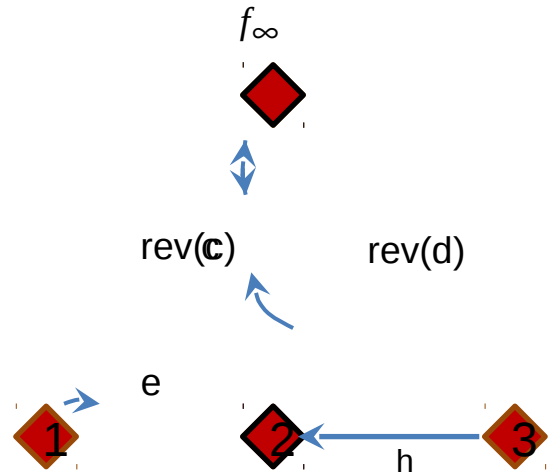
G :



T :

f=

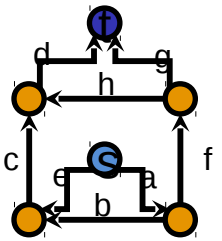
$$\begin{pmatrix} a & =1 \\ b & =1 \\ c & =1 \\ d & =1 \\ e & =0 \\ f & =0 \\ g & =0 \\ h & =0 \end{pmatrix}$$



T^* :

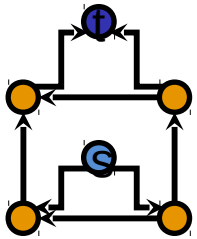
Algorithm's Basics

- Developed by Borradile & Klein 2006
- Consists of:
  - I. Preprocessing I:
    - Choose a face f_{root} incident to sink t

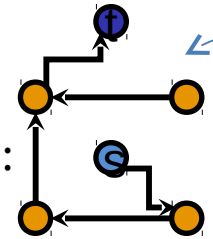


  - II. Preprocessing II: Leftmost Circulation

f_{root}, G:



f_{root}



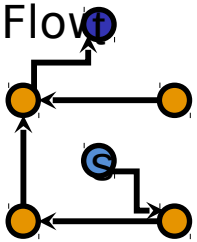
G_0 :

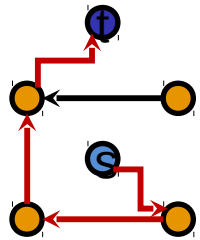
New capacities

, x

  - III. Leftmost Flow

G_0 :

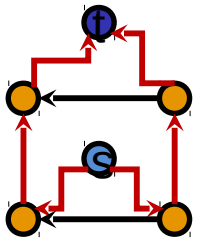




G_0 :

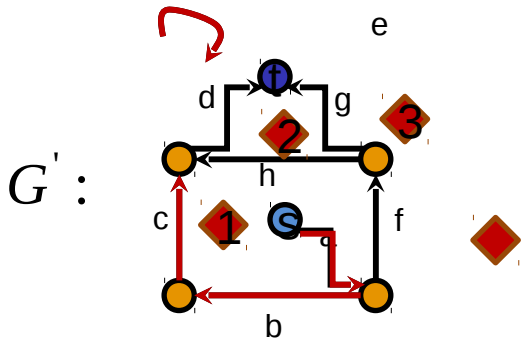
x

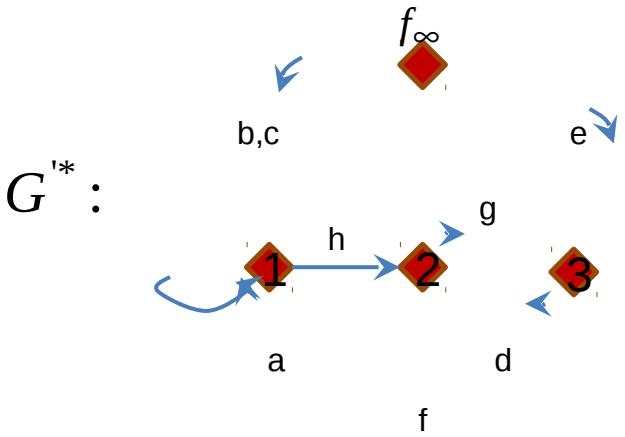
G :

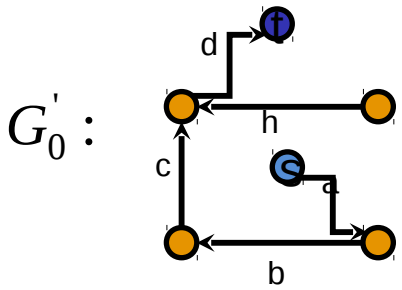


Leftmost Circulation & Cycles

- Reminder: No cycles in Leftmost Flow
- Take Graph G'



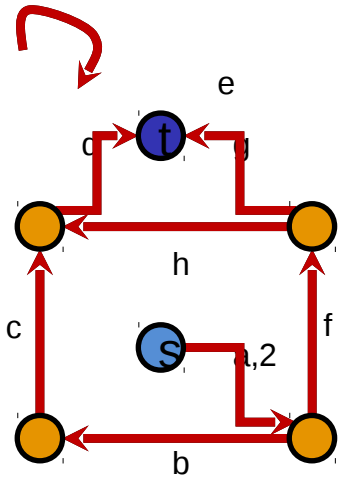




Leftmost

- Problem clockwise cycles

G' :



$$flow^1_{\max} = 2$$

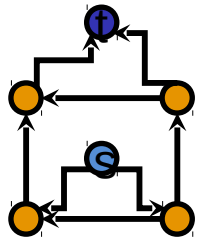
$$flow^2_{\max} = 1$$

- Appear always as leftmost path
- Flow assignment not correct for cyclic graphs
- Preprocessing necessary

Algorithm's Basics

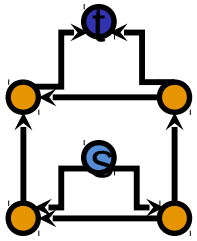
- Developed by Borradile & Klein 2006
- Consists of:

I. Preprocessing I:
   - Choose a face f_{root} incident to sink t

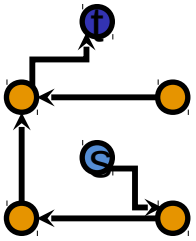


II. Leftmost Circulation η

$f_{root}, G:$



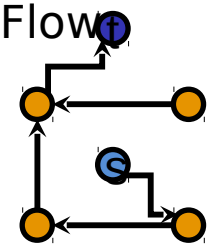
$G_0:$

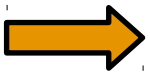


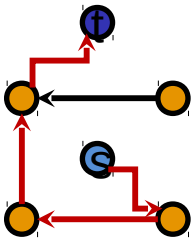
$,\eta$

III. Leftmost Flow

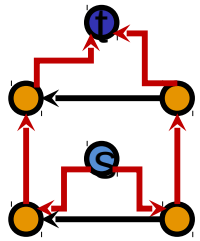
$\eta, G_0:$







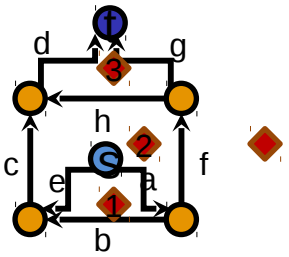
$+\eta$



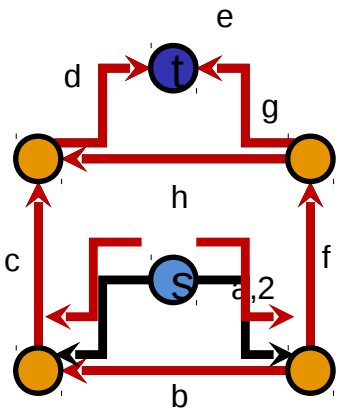
2nd Example

- Repeat:
  - If T[s] is residual, saturate T[s], modifying f
  - Let d be the last non-residual dart in in T[s]
  - If tail_G*(d) is a descendent in T* of head_G*(d), return f
  - Let e be the parent dart in T* of of head_G*(d)
  - Eject e from T* and insert d into T*
  - Eject d from T and insert e into T

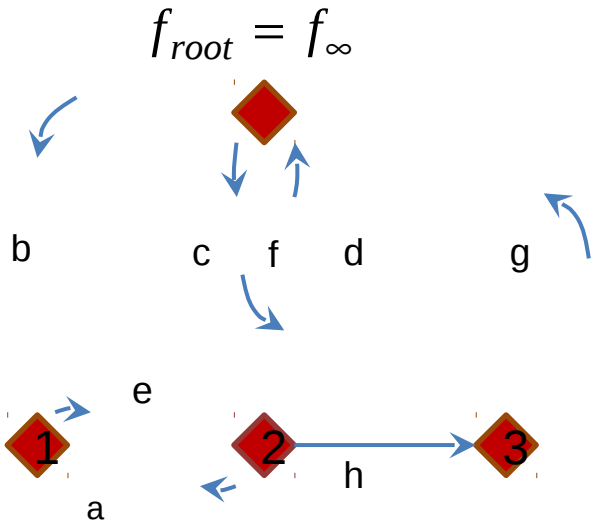
G:



T:



T^*:



Introduction