Network Flow

Design and Analysis of Algorithms

Andrei Bulatov

Flow Networks

Think of a graph as system of pipes

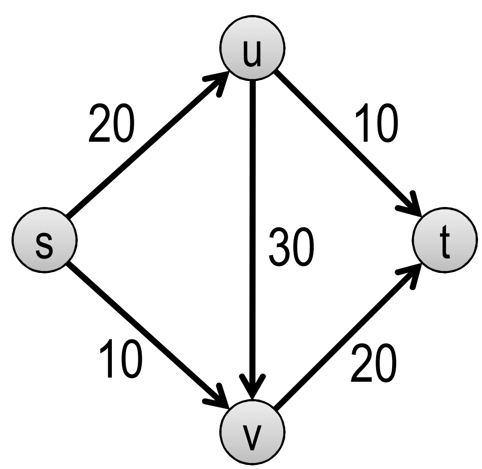
We use this system to pump water from the source s to sink t

Every pipe/edge has limited capacity

Flow occurs when we pump water through the system.

A flow is amount of water flowing through each pipe

How much water can we pump through the system without blowing up any pipes?



The Formalism

Flow Networks:

- a digraph G = (V;E)
- every edge e has capacity c_e, a nonnegative number
- there is a single source node s ∈ V
- there is a single sink node t ∈ V

Nodes other than s and t are called internal

Flows and Flow Networks

A flow network is a digraph with a unique source and sink nodes

Arcs have capacities

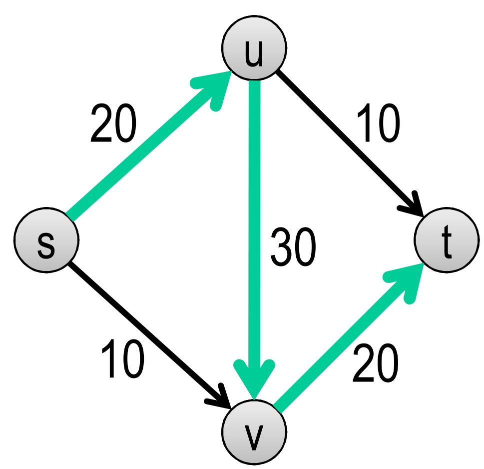
A flow is a function f: $E \to \mathbb{R}^+$ such that

(1) (Capacity condition) For each $e \in E$, we have $0 \le f(e) \le c_e$

(2) (Conservation condition, Kirchhoff principle)

for each node except s and t $\sum_{e \text{ into } v} f(e) = \sum_{e \text{ out of } v} f(e)$

The value of the flow is $\sum_{e \text{ out of } s} f(e)$



The Problem

The Maximum Flow Problem

Instance:

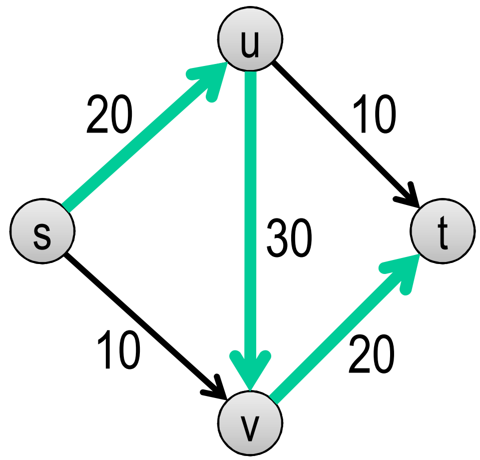
A flow network G, s, t

Objective:

Find a flow of maximal value.

Algorithm: Simple Flows and Residual Graph

Consider a flow network



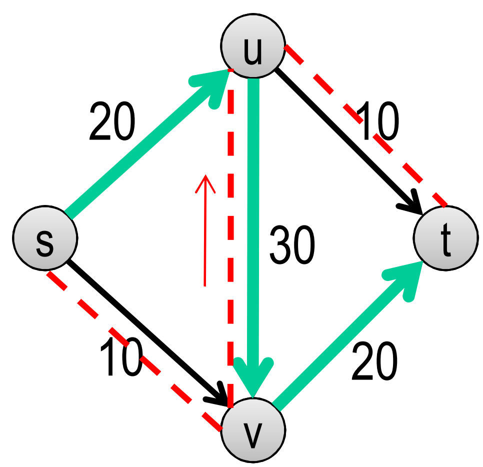
Natural idea:

push a flow along a path

However, the flow cannot be improved this way,

but can be improved in

a different way

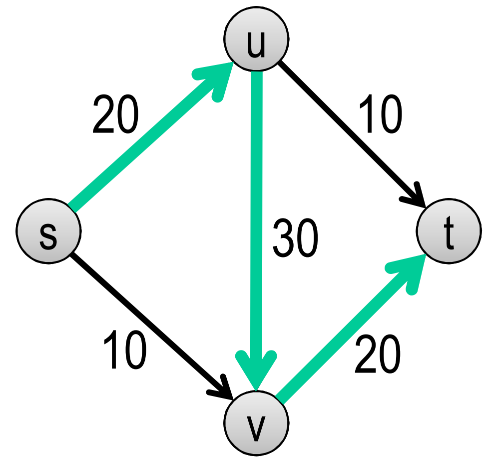


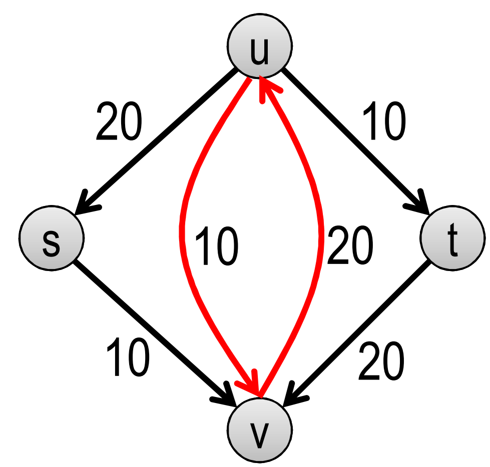
Residual Graph

Given a flow network G, and a flow f, construct the residual graph with respect to f

- the node set of G_f is the same as G
- for each edge e of G with $f(e) < c_e$ include e in G_f with capacity $c_e - f(e)$ (forward edge)
- for each edge e = (u,v) in G with f(e) > 0 include e' = (v,u) with capacity f(e) (backward edge)

Capacity of an edge in the residual graph is called residual capacity

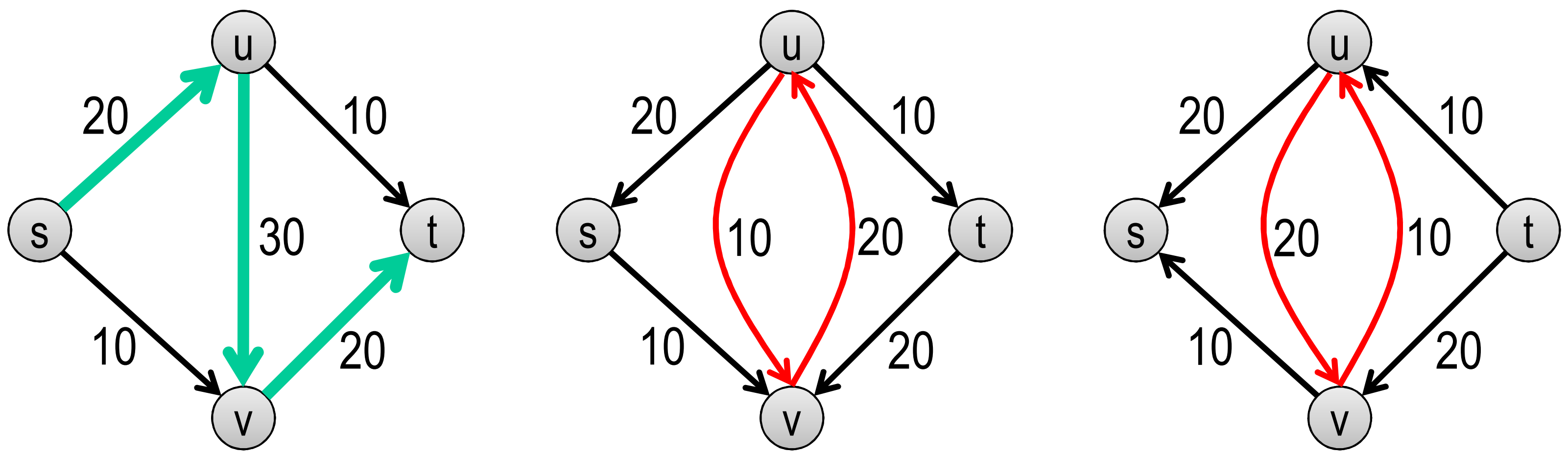




Residual Graph

Starting with the zero flow

- push a flow along (s,u), (u,v), (v,t) such that f(s,u) = f(u,v) = f(v,t) = 20
- construct the residual graph w.r.t. f
- push a flow along (s,v), (v,u), (u,t) s. t. g(s,v) = g(v,u) = g(u,t) = 10
- construct the residual graph w.r.t. g
- we cannot push any flow anymore.
- is f + g maximal?



Augmenting a Flow

Let P be an s-t path in G_f

bottleneck(P,f) denotes the minimal residual capacity of the edges of P

Augment(f,P)

```
set b:=bottleneck(P,f)
for each edge (u,v)∈P do
   if e=(u,v) is a forward edge then
     increase f(e) by b
   else decrease f(e) by b
endfor
return f
```

Any s-t path in G_f is called an augmenting path

Augmenting a Flow (cntd)

Let f’ be the function obtained after augmenting

Lemma

f’ is a flow

Proof

Capacity condition:

It suffices to consider arcs of P

Let $e = (u,v) \in P$

By construction bottleneck(P,f) is at most the residual capacity of e

If e is a forward edge, then

$$0 \le f(e) \le f'(e) = f(e) + \text{bottleneck}(P, f) \le f(e) + (c_e - f(e)) = c_e$$

Augmenting a Flow (cntd)

Proof (cntd)

If e is a backward edge, then

$$c_e \geq f(e) \geq f'(e) = f(e) - \text{bottleneck}(P, f) \geq f(e) - f(e) = 0$$

Conservation condition:

It suffices to observe that for every node the additional amount of flow, 0 or bottleneck(P,f) entering the node equals the additional amount of flow, 0 or bottleneck(P,f), leaving the node.

QED

Algorithm Ford-Falkerson

Max-Flow(G)

```
set f(e):=0 for all e in G
while there is an s-t path in the residual graph G_f do
   let P be a simple s-t path in G_f
   set f':=Augment(f,P)
   set G_f := G_f'
   set f:=f'
endwhile
return f
```

Termination

We find a parameter that increases every time Augment is applied.

Clearly, it is the value, v(f), of the flow

Lemma

At every stage of the algorithm, the flow values are integers

Lemma

Let f be a flow in G, and let P be a simple s-t path in G_f . Then v(f') = v(f) + bottleneck(P,f). Since bottleneck(P,f) > 0, we have v(f') > v(f).

Proof

The first arc of P leaves s, and P does not revisit s again.

Moreover, it is a forward arc. Hence v(f') = v(f) + bottleneck(P,f) > v(f)

Termination (cntd)

Corollary

Let C be the total capacity of arcs leaving s, i.e. $C = \sum_{e \text{ out of } s} c_e$

Then if all capacities in the flow network are integers , Ford-Falkerson terminates in at most C iterations of the while loop.

Proof

Since all capacities are integer, every iteration increases the value by at least 1.

QED

Running Time

Theorem

If all the capacities are integers then the Ford-Falkerson algorithm can be implemented to run in O(mC) time

Proof

The algorithm executes the while loop at most C times.

The residual graph G_f contains at most 2m edges.

Using BFS we find an s-t path in it in O(m + n) = O(m) time

Augmenting takes O(n) = O(m) time

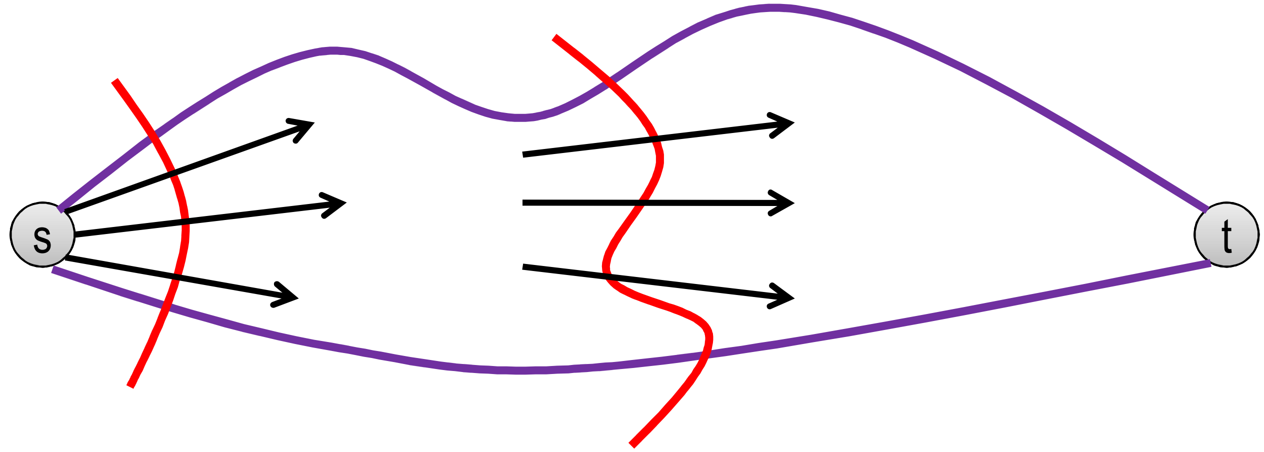
QED

Ford-Falkerson: Analysis

Theorem

If all the capacities are integers then the Ford-Falkerson algorithm finds a maximal flow.

Cuts



A cut is a partition of G into two sets, A and B, so that $s \in A$ and $t \in B$

The capacity of the cut is $c(A,B) = \sum_{e \text{ out of } A} c_e$

Also $f^{out}(A) = \sum_{e \text{ out of } A} f(e)$, $\quad f^{in}(A) = \sum_{e \text{ in } A} f(e)$

Lemma

For any flow f we have $v(f) = f^{out}(A) - f^{in}(A)$

Cuts and Flow Value

Proof

By definition $v(f) = f^{out}(s)$

Since $f^{in}(s) = 0$ we also have $v(f) = f^{out}(s) - f^{in}(s)$

Furthermore, $f^{out}(v) - f^{in}(v) = 0$ for v ≠ s,t

Thus $v(f) = \sum_{v \in A} f^{out}(v) - f^{in}(v)$

If both ends of e belong to A, it contributes 0 to the sum above

If the beginning of e is in A, it contributes positivly

If the end of e is in A, it contributes negatively

Hence

$$v(f) = \sum_{v \in A} f^{out}(v) - f^{in}(v) = \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in } A} f(e) = f^{out}(A) - f^{in}(A)$$

Cuts and Flow Value

Corollary

Let f be a flow and (A,B) a cut. Then $v(f) = f^{in}(t) - f^{out}(t)$

Corollary

Let f be a flow, and (A,B) a cut. Then $v(f) \leq c(A,B)$

Max Flow vs. Min Cut

Let f be the flow returned by the Ford-Falkerson algortihm.

We find a cut (A,B) such that v(f) = c(A,B)

By the Corollary above this means that v(f) is maximal possible, and that c(A,B) is the value of the maximal flow

Lemma

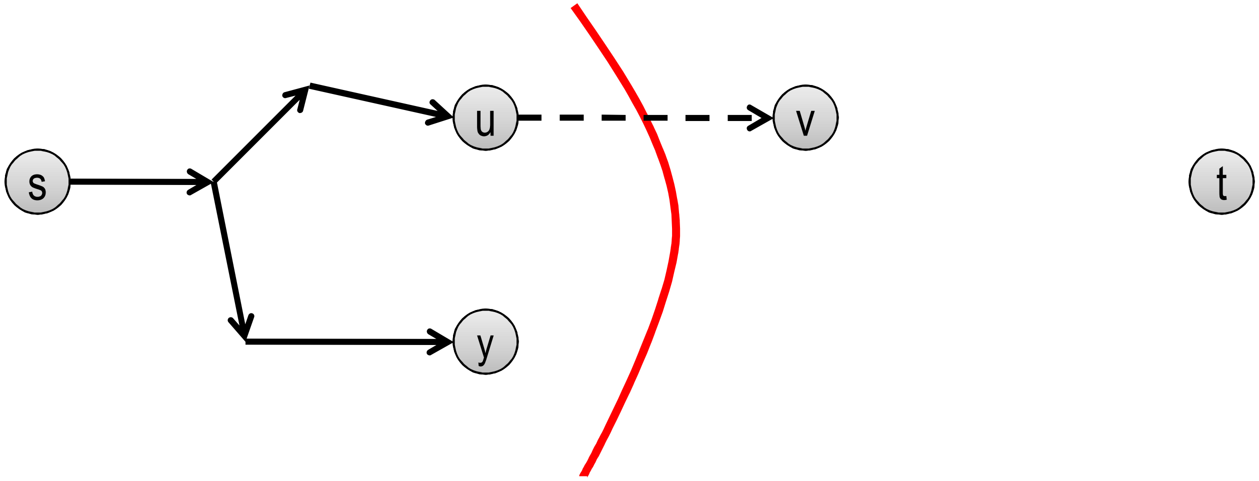
Let f be a flow such that there is no s-t path in the residual graph G_f
Then there is a cut (A,B) in G such that v(f) = c(A,B)

Proof

Let A be the set of all vertices v such that v is reachable from s in G_f

Let B be the remaining vertices

Max Flow vs. Min Cut (cntd)



First, show that (A,B) is a cut

Obviously, s ∈ A

Since there is no s-t path in G_f we have t ∉ A

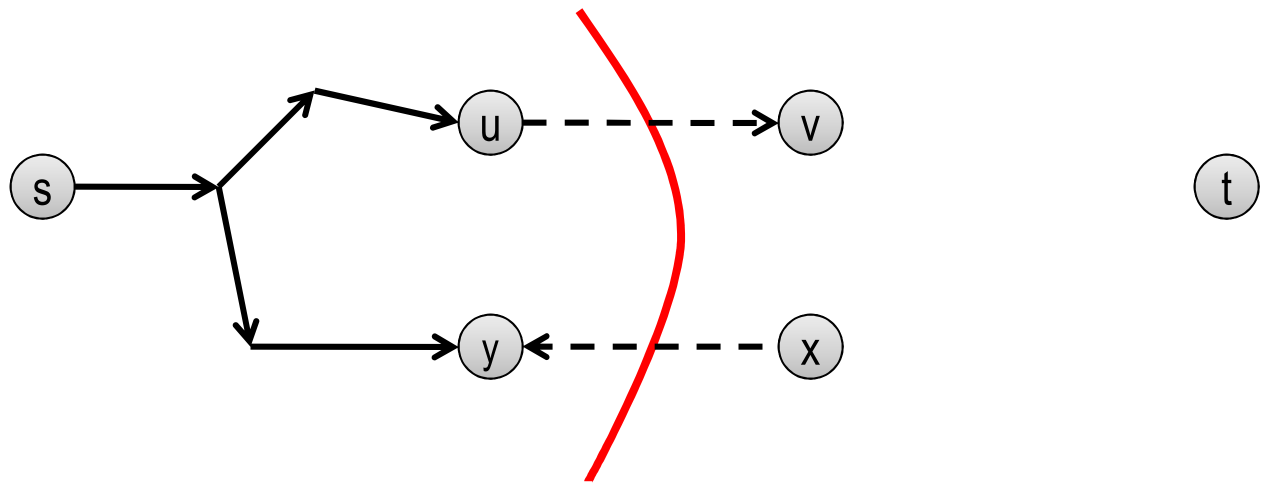
Second, suppose that e =(u,v) is an edge in G, for which u ∈ A , v ∈ B

Then $f(e) = c_e$.

Indeed, otherwise e would be a forward edge in G_f

A contradiction with the choice of A

Max Flow vs. Min Cut (cntd)



Third, suppose that e' =(x,y) is an edge in G, for which $x \in B$, $y \in A$

Then f(e) = 0

Indeed, otherwise the edge e'' = (y,x) would be a backward edge in G_f

A contradiction with the choice of A

Thus $v(f) = f^{out}(A) - f^{in}(A) = \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in } A} f(e)$

$$= \sum_{e \text{ out of } A} c_e - 0 = c(A,B)$$

Max Flow vs. Min Cut (cntd)

Corollary

The flow returned by the Ford-Falkerson algorithm is a maximal flow

Corollary

In every flow network the maximum value of a flow equals the minimum capacity of a cut

Corollary

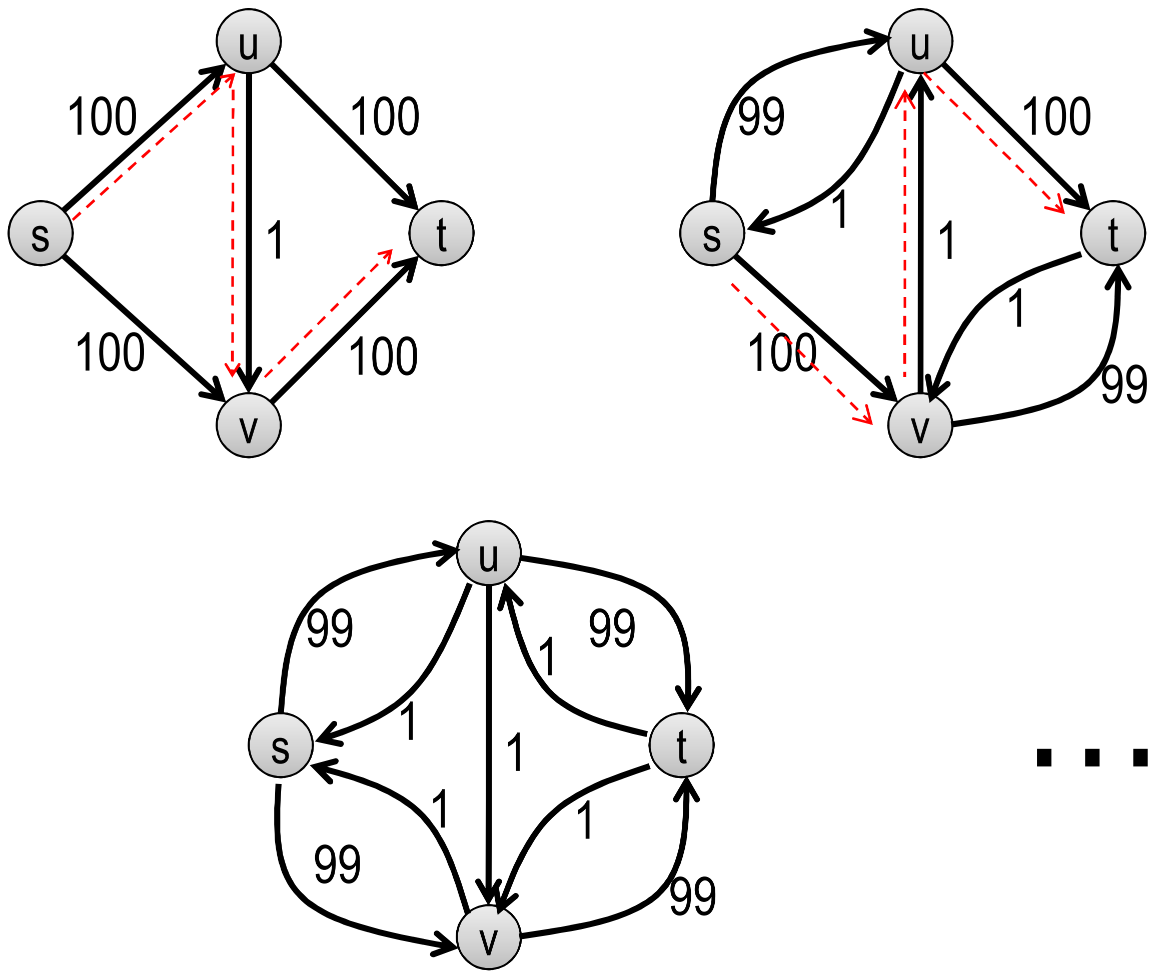
Given a flow of maximal value, we can compute a cut of minimum capacity in $O(m)$ time

Corollary

If all capacities in a flow network are integers, then there is a maximum flow f for which every $f(e)$ is an integer

Faster Flow

Selecting a good augmenting path makes a big difference



. . .

Faster Flow

We need to select augmenting paths with large bottleneck

It can be a difficult problem

Use scaling parameter Δ

Let $G_f(\Delta)$ denote the subgraph of the residual graph consisting only of edges with residual capacity at least Δ

Faster Ford-Falkerson

Scaling Max-Flow(G)

```
set f(e):=0 for all e in G
set Δ:=maximal power of 2 such that Δ ≤ max_{e out of s} c_e
while Δ≥1 do
   while there is an s-t path in the residual graph
      G_f(Δ) do
     let P be a simple s-t path in G_f(Δ)
     set f':=Augment(f,P)
     set G_f := G_f'
     set f:=f'
   endwhile
   set Δ:=Δ/2
endwhile
return f
```

Faster Ford-Falkerson: Analysis

Theorem

The Scaling Max-Flow algorithm in a graph with m edges and integer capacities finds a maximum flow in at most $2m(1 + \lceil \log C \rceil)$ augmentations.

It can be implemented in at most $O(m^2 \log C)$ time