Algorithms and datastructures II

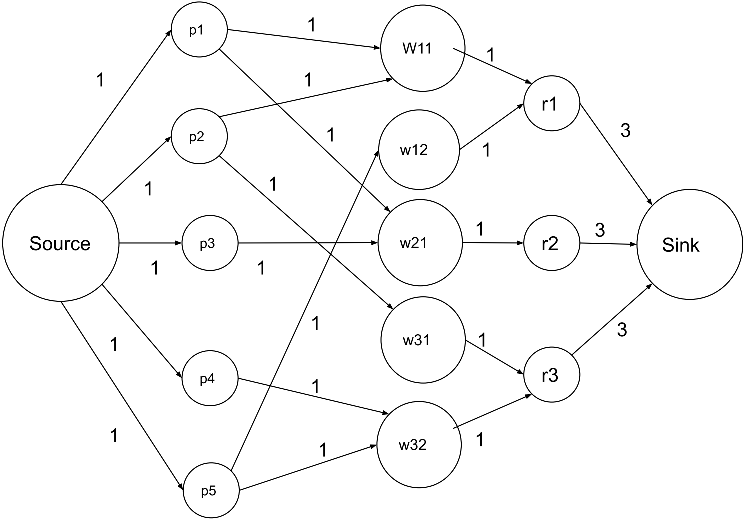
Lecture 3: network flows

Jan Hubička

Department of Applied Mathematics
Charles University
Prague

Oct 19 2020

Network flow



Network flow

Definition (Network)

Network is an 4-tuple $N = (V, E, s, t, c)$ where

1. (V, E) is a directed graph,
2. $s \in V$ is a source vertex,
3. $t \in V$ is a sink vertex,
4. $c : E \to \mathbb{R}_0^+$ is a function assigning every edge a capacity.

- $f^+(v) = \sum_{u,(u,v)\in E} f(u, v)$ (flow into a vertex)
- $f^-(v) = \sum_{u,(v,u)\in E} f(v, u)$ (flow out of a vertex)
- $f^\Delta(v) = f^+(v) - f^-(v)$ (surplus or excess)

Here $f : E \to \mathbb{R}_0^+$

Definition (Flow)

Function $f : E \to \mathbb{R}_0^+$ is flow if it satisfies

1. Capacity constraint: $(\forall_{e\in E}) : f(e) \leq c(e)$
2. Conservation of flows (Kirchoff's law): $(\forall_{v\in V\setminus\{s,t\}}) : f^\Delta(v) = 0$

Value of the flow: $|f| = f^\Delta(t)$.

Network flow

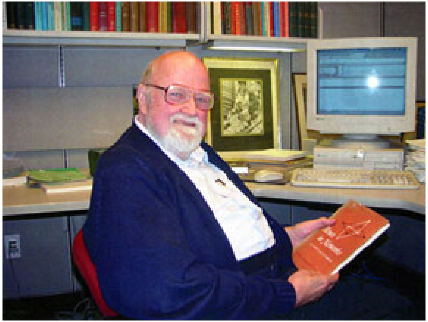
Network flow problem

Given network $N = (V, E, s, t, c)$ find flow f maximizing $|f|$ (a maximum flow).

Naive approach: Start with 0 flow and keep improving as long as there is path from source to sink that can be improved.

Today: we show that naive approach works and give an effective algorithm to solve this problem

Ford–Fulkerson algorithm, 1956



Ford–Fulkerson algorithm, 1956

Definition (Residual capacity)

$r(u, v) = c(u, v) - f(u, v) + f(v, u)$

Definition (Augmenting path)

A path in (V, E) is augmenting if every edge has non-zero residual capacity.

Ford–Fulkerson algorithm, 1956

Definition (Residual capacity)

$r(u, v) = c(u, v) - f(u, v) + f(v, u)$

Definition (Augmenting path)

A path in (V, E) is augmenting if every edge has non-zero residual capacity.

FordFulkerson(V, E, s, t, c)

1. $f \leftarrow$ zero flow (or flow of your choice).
2. While there exists augmenting path P from s to t:
3. $\quad \epsilon \leftarrow \min_{e \in P} r(e)$.
4. $\quad$ For every $\{u, v\} \in P$:
5. $\quad\quad \delta \leftarrow \min(f(v, u), \epsilon)$.
6. $\quad\quad f(v, u) \leftarrow f(v, u) - \delta$.
7. $\quad\quad f(u, v) \leftarrow f(u, v) + \epsilon - \delta$.
8. Return f (maximum flow).

Ford–Fulkerson algorithm, 1956

Definition (Residual capacity)

$r(u, v) = c(u, v) - f(u, v) + f(v, u)$

Definition (Augmenting path)

A path in (V, E) is augmenting if every edge has non-zero residual capacity.

FordFulkerson(V, E, s, t, c)

1. $f \leftarrow$ zero flow (or flow of your choice).
2. While there exists augmenting path P from s to t:
3. $\quad \epsilon \leftarrow \min_{e \in P} r(e)$.
4. $\quad$ For every $\{u, v\} \in P$:
5. $\quad\quad \delta \leftarrow \min(f(v, u), \epsilon)$.
6. $\quad\quad f(v, u) \leftarrow f(v, u) - \delta$.
7. $\quad\quad f(u, v) \leftarrow f(u, v) + \epsilon - \delta$.
8. Return f (maximum flow).

Lemma

If FordFulkerson terminates, it returns a flow.

Ford–Fulkerson algorithm, 1956

Definition (Residual capacity)

$r(u, v) = c(u, v) - f(u, v) + f(v, u)$

Definition (Augmenting path)

A path in (V, E) is augmenting if every edge has non-zero residual capacity.

FordFulkerson(V, E, s, t, c)

1. $f \leftarrow$ zero flow (or flow of your choice).
2. While there exists augmenting path P from s to t:
3. $\quad \epsilon \leftarrow \min_{e \in P} r(e)$.
4. $\quad$ For every $\{u, v\} \in P$:
5. $\quad\quad \delta \leftarrow \min(f(v, u), \epsilon)$.
6. $\quad\quad f(v, u) \leftarrow f(v, u) - \delta$.
7. $\quad\quad f(u, v) \leftarrow f(u, v) + \epsilon - \delta$.
8. Return f (maximum flow).

Lemma

If FordFulkerson terminates, it returns a flow.

Invariant: f is a flow.

Ford–Fulkerson algorithm, 1956

Definition (Residual capacity)

$r(u, v) = c(u, v) - f(u, v) + f(v, u)$

Definition (Augumenting path)

A path in (V, E) is augmenting if every edge has non-zero residual capacity.

FordFulkerson(V, E, s, t, c)

1. $f \leftarrow$ zero flow (or flow of your choice).
2. While there exists augmenting path P from s to t:
3. $\epsilon \leftarrow \min_{e \in P} r(e)$.
4. For every $\{u, v\} \in P$:
5. $\delta \leftarrow \min(f(v, u), \epsilon)$.
6. $f(v, u) \leftarrow f(v, u) - \delta$.
7. $f(u, v) \leftarrow f(u, v) + \epsilon - \delta$.
8. Return f (maximum flow).

Lemma

If FordFulkerson terminates, it returns a flow.

Invariant: f is a flow.

Definition (elementary cut)

(X, Y) is an (elementary) cut of graph (V, E) if:

1. $X, Y \subseteq V$,
2. $X \cup Y = V$,
3. $X \cap Y = \emptyset$,
4. $s \in X$,
5. $t \in Y$.

Ford–Fulkerson algorithm, 1956

Definition (Residual capacity)

$r(u, v) = c(u, v) - f(u, v) + f(v, u)$

Definition (Augmenting path)

A path in (V, E) is augmenting if every edge has non-zero residual capacity.

FordFulkerson(V, E, s, t, c)

1. $f \leftarrow$ zero flow (or flow of your choice).
2. While there exists augmenting path P from s to t:
3. $\quad \epsilon \leftarrow \min_{e \in P} r(e)$.
4. $\quad$ For every $\{u, v\} \in P$:
5. $\quad\quad \delta \leftarrow \min(f(v, u), \epsilon)$.
6. $\quad\quad f(v, u) \leftarrow f(v, u) - \delta$.
7. $\quad\quad f(u, v) \leftarrow f(u, v) + \epsilon - \delta$.
8. Return f (maximum flow).

Lemma

If FordFulkerson terminates, it returns a flow.

Invariant: f is a flow.

Definition (elementary cut)

(X, Y) is an (elementary) cut of graph (V, E) if:

1. $X, Y \subseteq V$,
2. $X \cup Y = V$,
3. $X \cap Y = \emptyset$,
4. $s \in X$,
5. $t \in Y$.

$$E(X, Y) = E \cap \{X \times Y\}$$

$$f(X, Y) = \sum_{e \in E(X,Y)} f(e)$$

$$f^{\Delta}(X, Y) = f(X, Y) - f(Y, X)$$

Ford–Fulkerson algorithm, 1956

Observation A

If f is a flow and (X, Y) a cut, then $f^{\Delta}(X, Y) = |f|$.

Lemma

If FordFulkerson terminates, it returns a flow.

Invariant: f is a flow.

Definition (elementary cut)

(X, Y) is an (elementary) cut of graph (V, E) if:

1. $X, Y \subseteq V$,
2. $X \cup Y = V$,
3. $X \cap Y = \emptyset$,
4. $s \in X$,
5. $t \in Y$.

$$E(X, Y) = E \cap \{X \times Y\}$$

$$f(X, Y) = \sum_{e \in E(X,Y)} f(e)$$

$$f^{\Delta}(X, Y) = f(X, Y) - f(Y, X)$$

Ford–Fulkerson algorithm, 1956

Observation A

If f is a flow and (X, Y) a cut, then $f^{\Delta}(X, Y) = |f|$.

$$f^{\Delta}(X, Y) = \sum_{v \in X} f^{\Delta}(v) = f^{\Delta}(t) = |f|$$

Lemma

If FordFulkerson terminates, it returns a flow.

Invariant: f is a flow.

Definition (elementary cut)

(X, Y) is an (elementary) cut of graph (V, E) if:

1. $X, Y \subseteq V$,
2. $X \cup Y = V$,
3. $X \cap Y = \emptyset$,
4. $s \in X$,
5. $t \in Y$.

$$E(X, Y) = E \cap \{X \times Y\}$$

$$f(X, Y) = \sum_{e \in E(X,Y)} f(e)$$

$$f^{\Delta}(X, Y) = f(X, Y) - f(Y, X)$$

Ford–Fulkerson algorithm, 1956

Observation A

If f is a flow and (X, Y) a cut, then $f^{\Delta}(X, Y) = |f|$.

$$f^{\Delta}(X, Y) = \sum_{v \in X} f^{\Delta}(v) = f^{\Delta}(t) = |f|$$

Observation B

If f is a flow and (X, Y) a cut, then $|f| \leq c(X, Y)$.

Lemma

If FordFulkerson terminates, it returns a flow.

Invariant: f is a flow.

Definition (elementary cut)

(X, Y) is an (elementary) cut of graph (V, E) if:

1. $X, Y \subseteq V$,
2. $X \cup Y = V$,
3. $X \cap Y = \emptyset$,
4. $s \in X$,
5. $t \in Y$.

$$E(X, Y) = E \cap \{X \times Y\}$$

$$f(X, Y) = \sum_{e \in E(X,Y)} f(e)$$

$$f^{\Delta}(X, Y) = f(X, Y) - f(Y, X)$$

Ford–Fulkerson algorithm, 1956

Observation A

If f is a flow and (X, Y) a cut, then $f^{\Delta}(X, Y) = |f|$.

$$f^{\Delta}(X, Y) = \sum_{v \in X} f^{\Delta}(v) = f^{\Delta}(t) = |f|$$

Observation B

If f is a flow and (X, Y) a cut, then $|f| \leq c(X, Y)$.

$$|f| = f^{\Delta}(X, Y) = f(X, Y) - f(Y, X) \leq f(X, Y) \leq c(X, Y)$$

Lemma

If FordFulkerson terminates, it returns a flow.

Invariant: f is a flow.

Definition (elementary cut)

(X, Y) is an (elementary) cut of graph (V, E) if:

1. $X, Y \subseteq V$,
2. $X \cup Y = V$,
3. $X \cap Y = \emptyset$,
4. $s \in X$,
5. $t \in Y$.

$$E(X, Y) = E \cap \{X \times Y\}$$

$$f(X, Y) = \sum_{e \in E(X, Y)} f(e)$$

$$f^{\Delta}(X, Y) = f(X, Y) - f(Y, X)$$

Ford–Fulkerson algorithm, 1956

Observation A

If f is a flow and (X, Y) a cut, then $f^{\Delta}(X, Y) = |f|$.

$$f^{\Delta}(X, Y) = \sum_{v \in X} f^{\Delta}(v) = f^{\Delta}(t) = |f|$$

Observation B

If f is a flow and (X, Y) a cut, then $|f| \leq c(X, Y)$.

$$|f| = f^{\Delta}(X, Y) = f(X, Y) - f(Y, X) \leq f(X, Y) \leq c(X, Y)$$

Observation C

If f is a flow and (X, Y) a cut, $|f| = c(X, Y)$ then $|f|$ is maximum and $c(X, Y)$ is minimum possible.

Lemma

If FordFulkerson terminates, it returns a flow.

Invariant: f is a flow.

Definition (elementary cut)

(X, Y) is an (elementary) cut of graph (V, E) if:

1. $X, Y \subseteq V$,
2. $X \cup Y = V$,
3. $X \cap Y = \emptyset$,
4. $s \in X$,
5. $t \in Y$.

$$E(X, Y) = E \cap \{X \times Y\}$$

$$f(X, Y) = \sum_{e \in E(X, Y)} f(e)$$

$$f^{\Delta}(X, Y) = f(X, Y) - f(Y, X)$$

Ford–Fulkerson algorithm, 1956

Observation A

If f is a flow and (X, Y) a cut, then $f^{\Delta}(X, Y) = |f|$.

$$f^{\Delta}(X, Y) = \sum_{v \in X} f^{\Delta}(v) = f^{\Delta}(t) = |f|$$

Observation B

If f is a flow and (X, Y) a cut, then $|f| \leq c(X, Y)$.

$$|f| = f^{\Delta}(X, Y) = f(X, Y) - f(Y, X) \leq f(X, Y) \leq c(X, Y)$$

Observation C

If f is a flow and (X, Y) a cut, $|f| = c(X, Y)$ then $|f|$ is maximum and $c(X, Y)$ is minimum possible.

Lemma

If FordFulkerson terminates, it returns a flow.

Invariant: f is a flow.

Definition (elementary cut)

(X, Y) is an (elementary) cut of graph (V, E) if:

1. $X, Y \subseteq V$,
2. $X \cup Y = V$,
3. $X \cap Y = \emptyset$,
4. $s \in X$,
5. $t \in Y$.

$$E(X, Y) = E \cap \{X \times Y\}$$

$$f(X, Y) = \sum_{e \in E(X, Y)} f(e)$$

$$f^{\Delta}(X, Y) = f(X, Y) - f(Y, X)$$

Ford–Fulkerson algorithm, 1956

Observation A

If f is a flow and (X, Y) a cut, then $f^{\Delta}(X, Y) = |f|$.

$$f^{\Delta}(X, Y) = \sum_{v \in X} f^{\Delta}(v) = f^{\Delta}(t) = |f|$$

Observation B

If f is a flow and (X, Y) a cut, then $|f| \leq c(X, Y)$.

$$|f| = f^{\Delta}(X, Y) = f(X, Y) - f(Y, X) \leq f(X, Y) \leq c(X, Y)$$

Observation C

If f is a flow and (X, Y) a cut, $|f| = c(X, Y)$ then $|f|$ is maximum and $c(X, Y)$ is minimum possible.

Lemma

If FordFulkerson terminates, it returns a flow.

Lemma

If FordFulkerson terminates it returns a maximum flow

Ford–Fulkerson algorithm, 1956

Observation A

If f is a flow and (X, Y) a cut, then $f^{\Delta}(X, Y) = |f|$.

$$f^{\Delta}(X, Y) = \sum_{v \in X} f^{\Delta}(v) = f^{\Delta}(t) = |f|$$

Observation B

If f is a flow and (X, Y) a cut, then $|f| \leq c(X, Y)$.

$$|f| = f^{\Delta}(X, Y) = f(X, Y) - f(Y, X) \leq f(X, Y) \leq c(X, Y)$$

Observation C

If f is a flow and (X, Y) a cut, $|f| = c(X, Y)$ then $|f|$ is maximum and $c(X, Y)$ is minimum possible.

Lemma

If FordFulkerson terminates, it returns a flow.

Lemma

If FordFulkerson terminates it returns a maximum flow

Proof.

$X = \{v \in V : \text{there exists augumenting path } s \text{ to } v\}$
Because algorithm terminates, we know that $t \notin X$.

□

Ford–Fulkerson algorithm, 1956

Observation A

If f is a flow and (X, Y) a cut, then $f^{\Delta}(X, Y) = |f|$.

$$f^{\Delta}(X, Y) = \sum_{v \in X} f^{\Delta}(v) = f^{\Delta}(t) = |f|$$

Observation B

If f is a flow and (X, Y) a cut, then $|f| \leq c(X, Y)$.

$$|f| = f^{\Delta}(X, Y) = f(X, Y) - f(Y, X) \leq f(X, Y) \leq c(X, Y)$$

Observation C

If f is a flow and (X, Y) a cut, $|f| = c(X, Y)$ then $|f|$ is maximum and $c(X, Y)$ is minimum possible.

Lemma

If FordFulkerson terminates, it returns a flow.

Lemma

If FordFulkerson terminates it returns a maximum flow

Proof.

$X = \{v \in V :$ there exists augumenting path s to $v\}$
Because algorithm terminates, we know that $t \notin X$.
Put $Y = V \setminus X$.

□

Ford–Fulkerson algorithm, 1956

Observation A

If f is a flow and (X, Y) a cut, then $f^{\Delta}(X, Y) = |f|$.

$$f^{\Delta}(X, Y) = \sum_{v \in X} f^{\Delta}(v) = f^{\Delta}(t) = |f|$$

Observation B

If f is a flow and (X, Y) a cut, then $|f| \leq c(X, Y)$.

$$|f| = f^{\Delta}(X, Y) = f(X, Y) - f(Y, X) \leq f(X, Y) \leq c(X, Y)$$

Observation C

If f is a flow and (X, Y) a cut, $|f| = c(X, Y)$ then $|f|$ is maximum and $c(X, Y)$ is minimum possible.

Lemma

If FordFulkerson terminates, it returns a flow.

Lemma

If FordFulkerson terminates it returns a maximum flow

Proof.

$X = \{v \in V : \text{there exists augumenting path } s \text{ to } v\}$
Because algorithm terminates, we know that $t \notin X$.
Put $Y = V \setminus X$.
(X, Y) is a cut and by Observation C f is maximum flow.
□

Ford–Fulkerson algorithm, 1956

Observation A

If f is a flow and (X, Y) a cut, then $f^{\Delta}(X, Y) = |f|$.

$$f^{\Delta}(X, Y) = \sum_{v \in X} f^{\Delta}(v) = f^{\Delta}(t) = |f|$$

Observation B

If f is a flow and (X, Y) a cut, then $|f| \leq c(X, Y)$.

$$|f| = f^{\Delta}(X, Y) = f(X, Y) - f(Y, X) \leq f(X, Y) \leq c(X, Y)$$

Observation C

If f is a flow and (X, Y) a cut, $|f| = c(X, Y)$ then $|f|$ is maximum and $c(X, Y)$ is minimum possible.

Lemma

If FordFulkerson terminates, it returns a flow.

Lemma

If FordFulkerson terminates it returns a maximum flow

Proof.

$X = \{v \in V : \text{there exists augumenting path } s \text{ to } v\}$
Because algorithm terminates, we know that $t \notin X$.
Put $Y = V \setminus X$.
(X, Y) is a cut and by Observation C f is maximum flow.
□

Observation

If capacities are integers, FordFulkerson terminates

Homework: Time complexity when all edges have capacity 1.

Yefim Dinitz

Construction of layered residual network

Definition (Reall: Residual capacity)

$r(u, v) = c(u, v) - f(u, v) + f(v, u)$.

Construction of layered residual network

Definition (Reall: Residual capacity)

$r(u, v) = c(u, v) - f(u, v) + f(v, u)$.

Definition (Residual network)

Let $N = (V, E, s, t, c)$ be a network and f a flow. Then the residual network is network defined as:

$$R(S, f) = (V, E, s, t, r).$$

Construction of layered residual network

Definition (Reall: Residual capacity)

$r(u, v) = c(u, v) - f(u, v) + f(v, u)$.

Definition (Residual network)

Let $N = (V, E, s, t, c)$ be a network and f a flow. Then the residual network is network defined as:

$$R(S, f) = (V, E, s, t, r).$$

Definition (Blocking flow)

Flow f is blocking flow if for every (oriented) path from s to t contains edge e with $f(e) = c(e)$.

Idea: finding a blocking flow is easier than maximum flow.

Construction of layered residual network

Definition (Reall: Residual capacity)

$r(u, v) = c(u, v) - f(u, v) + f(v, u)$.

Definition (Residual network)

Let $N = (V, E, s, t, c)$ be a network and f a flow. Then the residual network is network defined as:

$$R(S, f) = (V, E, s, t, r).$$

Definition (Blocking flow)

Flow f is blocking flow if for every (oriented) path from s to t contains edge e with $f(e) = c(e)$.

Idea: finding a blocking flow is easier than maximum flow.

Definition (Layered network)

Network is layered if it consists only of vertices and edges that belong to some shortest path from s to t.

Construction of layered residual network

Definition (Reall: Residual capacity)

$r(u, v) = c(u, v) - f(u, v) + f(v, u)$.

Definition (Residual network)

Let $N = (V, E, s, t, c)$ be a network and f a flow. Then the residual network is network defined as:

$$R(S, f) = (V, E, s, t, r).$$

Definition (Blocking flow)

Flow f is blocking flow if for every (oriented) path from s to t contains edge e with $f(e) = c(e)$.

Idea: finding a blocking flow is easier than maximum flow.

Definition (Layered network)

Network is layered if it consists only of vertices and edges that belong to some shortest path from s to t.

LayeredNetwork($R = (V, E, s, t, r)$)

1. Using BFS(s) determine layers in (V, E).
2. Remove all layers after t.
3. Remove edges inside layers and edges going backwards.

Construction of layered residual network

Definition (Reall: Residual capacity)

$r(u, v) = c(u, v) - f(u, v) + f(v, u).$

Definition (Residual network)

Let $N = (V, E, s, t, c)$ be a network and f a flow. Then the residual network is network defined as:

$$R(S, f) = (V, E, s, t, r).$$

Definition (Blocking flow)

Flow f is blocking flow if for every (oriented) path from s to t contains edge e with $f(e) = c(e)$.

Idea: finding a blocking flow is easier than maximum flow.

Definition (Layered network)

Network is layered if it consists only of vertices and edges that belong to some shortest path from s to t.

LayeredNetwork($R = (V, E, s, t, r)$)

1. Using BFS(s) determine layers in (V, E).
2. Remove all layers after t.
3. Remove edges inside layers and edges going backwards.
4. Remove dead ends:
5. $Q = \{v \neq z : deg^{out}(v) = 0\}$
6. While there exists $v \leftarrow Dequeue(Q)$:
7. Remove v and all edges containing v.
8. If removal of some edge decreased deg^{out} of some vertex to 0, add it to Q.
9. Return R.

Constructing blocking flow of layered network

BlockingFlow (Layered $R = (V, E, s, t, r)$)

1. $g \leftarrow$ zero flow.
2. While there exists oriented path P from s to t:
3. $\quad\epsilon \leftarrow \min_{e \in P}(r(e) - g(e))$.
4. $\quad$For every $e \in P : g(e) \leftarrow g(e) + \epsilon$.
5. $\quad$Remove from E all $e \in P$ such that $g(e) = r(e)$.
6. $\quad$Remove dead ends like in previous algorithm.
7. Return g.

Constructing blocking flow of layered network

BlockingFlow (Layered $R = (V, E, s, t, r)$)

1. $g \leftarrow$ zero flow.
2. While there exists oriented path P from s to t:
3. $\epsilon \leftarrow \min_{e \in P}(r(e) - g(e))$.
4.   For every $e \in P : g(e) \leftarrow g(e) + \epsilon$.
5.   Remove from E all $e \in P$ such that $g(e) = r(e)$.
6.   Remove dead ends like in previous algorithm.
7. Return g.

Time complexity: $O(|V||E|)$

Dinic algorithm

Dinic (V, E, s, t, c)

1. $f \leftarrow$ zero flow.
2. Repeat:
3. Build residual network R and remove all edges e with $r(e) = 0$.
4. $\ell \leftarrow$length of the shortest oriented path from s to t in R.
5. If there is no such path return f.
6. $L \leftarrow$LayeredNetwork (R).
7. $g \leftarrow$BlockingFlow (L).
8. Improve flow f using g.

Dinic algorithm

Dinic (V, E, s, t, c)

1. $f \leftarrow$ zero flow.
2. Repeat:
3. Build residual network R and remove all edges e with $r(e) = 0$.
4. $\ell \leftarrow$length of the shortest oriented path from s to t in R.
5. If there is no such path return f.
6. $L \leftarrow$LayeredNetwork (R).
7. $g \leftarrow$BlockingFlow (L).
8. Improve flow f using g.

Lemma (On improving flows)

For every flow f in network $N = (V, E, s, t, c)$ and flow g in $R(S, f) = (V, E, s, t, r)$ one can construct in time $O(m)$ flow h in S such that $|h| = |f| + |g|$.

Proof.

For every (u, v) such that $\{u, v\} \in E$ compute $h^*(u, v) = f(u, v) - f(v, u) + g(u, v) - g(v, u)$.
If $h^*(u, v) \geq 0$ put $h(u, v) = h^*(u, v)$ and $h(v, u) = 0$. □

Dinic algorithm

Dinic (V, E, s, t, c)

1. $f \leftarrow$ zero flow.
2. Repeat:
3. Build residual network R and remove all edges e with $r(e) = 0$.
4. $\ell \leftarrow$length of the shortest oriented path from s to t in R.
5. If there is no such path return f.
6. $L \leftarrow$LayeredNetwork (R).
7. $g \leftarrow$BlockingFlow (L).
8. Improve flow f using g.

Lemma

Every iteration of the loop increases ℓ by at least 1.

Loop iterates at most $|V|$ times. It follows that the runtime of algorithm is $O(|V|^2|E|)$.

Dinic algorithm

Lemma

Every iteration of the loop increases ℓ by at least 1.

Proof.

Denote by R_i the residual network R at iteration i.
What is difference of R_{i+1} and R_i?

Dinic algorithm

Lemma

Every iteration of the loop increases ℓ by at least 1.

Proof.

Denote by R_i the residual network R at iteration i.
What is difference of R_{i+1} and R_i?
We removed edge from every path of length ℓ, but new edges can appear.

Dinic algorithm

Lemma

Every iteration of the loop increases ℓ by at least 1.

Proof.

Denote by R_i the residual network R at iteration i.
What is difference of R_{i+1} and R_i?
We removed edge from every path of length ℓ, but new edges can appear.
Those edges appear only on paths of length $\ell + 2$. □

Dinic algorithm

Lemma

Every iteration of the loop increases ℓ by at least 1.

Proof.

Denote by R_i the residual network R at iteration i.
What is difference of R_{i+1} and R_i?
We removed edge from every path of length ℓ, but new edges can appear.
Those edges appear only on paths of length $\ell + 2$. □

Theorem

Dinic algorithm will terminate in time $O(|V|^2|E|)$ and will return maximum flow.

Maximality of flow returned follows from the same analysis as in Ford–Fulkerson's algorithm.