Improved flow algorithms

Peter Hajnal

Bolyai Institute, University of Szeged, Hungary

2023 fall

Residual network

Definition: Residual network, $\mathcal{R} = \mathcal{R}_f$

Let $\mathcal{H} = (\overrightarrow{G}, s, t, c)$ be a network and f is a flow in it.

Residual network

Definition: Residual network, $\mathcal{R} = \mathcal{R}_f$

Let $\mathcal{H} = (\overrightarrow{G}, s, t, c)$ be a network and f is a flow in it.
The graph of $\mathcal{R}$ is denoted as $\overrightarrow{G}_r$. Its vertex set is $V(\overrightarrow{G})$, the sourse/sink pair is s/t.

Residual network

Definition: Residual network, $\mathcal{R} = \mathcal{R}_f$

Let $\mathcal{H} = (\overrightarrow{G}, s, t, c)$ be a network and f is a flow in it.
The graph of $\mathcal{R}$ is denoted as $\overrightarrow{G}_r$. Its vertex set is $V(\overrightarrow{G})$, the sourse/sink pair is s/t. To see the edges and capacities we do the following for each edge $e = \overrightarrow{uv} \in E(\overrightarrow{G})$:

(i) If $0 < f(e) < c(e)$, then we take an edge $e_r^+ = \overrightarrow{uv}$ with capacity $c(e) - f(e)$, furthermore we take an edge $e_r^- = \overrightarrow{vu}$ with capacity $f(e)$.

(ii) If $0 = f(e) < c(e)$, then we take and edge $e_r^+ = \overrightarrow{uv}$ with capacity $c(e) - f(e)$.

(iii) If $0 < f(e) = c(e)$, then we introduce an edge $e_r^- = \overrightarrow{vu}$ with capacity $f(e)$.

Residual graph and augmenting paths

Residual graph and augmenting paths

- There is a bijection between augmenting paths of the original network and st directed paths in the residual network.

An other view of Ford-Fulkerson algorithm

An other view of Ford-Fulkerson algorithm

Ford—Fulkerson for finding an augmenting path

(1) Build the residual graph $\overrightarrow{G}_r$.

(2) Find a directed st-path in $\overrightarrow{G}_r$.

An other view of Ford-Fulkerson algorithm

Ford—Fulkerson for finding an augmenting path

(1) Build the residual graph $\overrightarrow{G}_r$.
(2) Find a directed st-path in $\overrightarrow{G}_r$.

The search of Ford and Fulkerson was „naive".

Naive search for directed path

Naive search for directed path

- Ford and Fulkerson introduced the sets

$B_{\text{forward}} = \{x \in V - S : \text{there is a vertex } y \in S \text{ s.t. } \overrightarrow{yx} \in E \text{ and } f(\overrightarrow{yx}) < c$

and

$B_{\text{backward}} = \{x \in V - S : \text{there is a vertex } y \in S \text{ s.t. } \overrightarrow{xy} \in E \text{ and } f(\overrightarrow{xy}) >$

Naive search for directed path

- Ford and Fulkerson introduced the sets

$B_{\text{forward}} = \{x \in V - S : \text{there is a vertex } y \in S \text{ s.t. } \overrightarrow{yx} \in E \text{ and } f(\overrightarrow{yx}) < c$

and

$B_{\text{backward}} = \{x \in V - S : \text{there is a vertex } y \in S \text{ s.t. } \overrightarrow{xy} \in E \text{ and } f(\overrightarrow{xy}) >$

- These are the outgoing edges for vertex set S in the residual graph.

Naive search for directed path

- Ford and Fulkerson introduced the sets

$B_{\text{forward}} = \{x \in V - S : \text{there is a vertex } y \in S \text{ s.t. } \overrightarrow{yx} \in E \text{ and } f(\overrightarrow{yx}) < c$

and

$B_{\text{backward}} = \{x \in V - S : \text{there is a vertex } y \in S \text{ s.t. } \overrightarrow{xy} \in E \text{ and } f(\overrightarrow{xy}) >$

- These are the outgoing edges for vertex set S in the residual graph.
- As soon we find a vertex of these sets we extend S with it.

The idea of Edmonds and Karp: Breadth first search

The idea of Edmonds and Karp: Breadth first search

Edmonds-Karp algorithm for finding augmenting path

Given a network $\mathcal{H}$ and a flow f in it.

(R) `Building the residual graph:` Build the residual graph $\overrightarrow{G}_r$.

(I) `Initialization:` Let $S_0 = \{s\}$, $i = 0$,
// $S = S_0 \cup \ldots \cup S_i$, the set vertices x's that can be reach by a directed sx-path of length at most i.

(E) `Extension:` Let S_{i+1} the set of out-neighbors of S_i.

- If $t \in S_{i+1}$, then we have a directed st-path, i.e. we found an f-augmenting path in the original network: Successful search.
- If $S_{i+1} = \emptyset$, then we stop: Unsuccessful search.
- If $t \notin S_{i+1} \neq \emptyset$, then $i \leftarrow i + 1$ and back to (E).

The properties of the breadth first search

The properties of the breadth first search

($\star$) In the case of unsuccessful search we have $t \notin S$, and there is no edge leading from S to $\overline{S} = V(G) - S$.

The properties of the breadth first search

($\star$) In the case of unsuccessful search we have $t \notin S$, and there is no edge leading from S to $\overline{S} = V(G) - S$.

- ($\star$) guarantees that there is no directed *st*-path.

The properties of the breadth first search

($\star$) In the case of unsuccessful search we have $t \notin S$, and there is no edge leading from S to $\overline{S} = V(G) - S$.

- ($\star$) guarantees that there is no directed st-path.

($\star\star$) In the case $\overrightarrow{xy} \in E(\overrightarrow{G})$, $x \in S_i$, and $x \in S_j$ we have $j \leq i + 1$.

The properties of the breadth first search

($\star$) In the case of unsuccessful search we have $t \notin S$, and there is no edge leading from S to $\overline{S} = V(G) - S$.

- ($\star$) guarantees that there is no directed st-path.

($\star\star$) In the case $\overrightarrow{xy} \in E(\overrightarrow{G})$, $x \in S_i$, and $x \in S_j$ we have $j \leq i + 1$.

- ($\star\star$) guarantees that we find the shortest augmenting path.

The properties of the breadth first search

($\star$) In the case of unsuccessful search we have $t \notin S$, and there is no edge leading from S to $\overline{S} = V(G) - S$.

- ($\star$) guarantees that there is no directed st-path.

($\star\star$) In the case $\overrightarrow{xy} \in E(\overrightarrow{G})$, $x \in S_i$, and $x \in S_j$ we have $j \leq i + 1$.

- ($\star\star$) guarantees that we find the shortest augmenting path.

Analysis of breadth first search

One can implement breadth first search with worst case complexity

$$\mathcal{O}(|E| + |V|).$$

Old vs new

Old vs new

- The Ford-Fulkerson algorithm is looking for an optimal flow. In the case of exact real arithmetic cycling is a real danger.

Old vs new

- The Ford-Fulkerson algorithm is looking for an optimal flow. In the case of exact real arithmetic cycling is a real danger.

- Edmonds and Karp suggested using breadth first search and proved that in this case there is polynomial upper bound on the number of augmentations in terms of $|E|$ and $|V|$.

The essential part of $\overrightarrow{G}_r$ from the point of breadth first search

The essential part of $\overrightarrow{G}_r$ from the point of breadth first search

Definition

Assume that we have a successful search in the residual graph ($t \in S_\ell$). Let $\overrightarrow{G}_r^0$ be the subgraph of the residual graph that contains exactly those vertices and edges that are on a shortest $\overrightarrow{st}$-path. The vertices is denoted as $S_0^0 \cup S_1^0 \cup S_2^0 \cup \ldots \cup S_\ell^0$, where $S_i^0 \subset S_i$, $S_0^0 = \{s\}$, $S_\ell^0 = \{t\}$.

The essential part of $\overrightarrow{G}_r$ from the point of breadth first search

Definition

Assume that we have a successful search in the residual graph ($t \in S_\ell$). Let $\overrightarrow{G}_r^0$ be the subgraph of the residual graph that contains exactly those vertices and edges that are on a shortest $\overrightarrow{st}$-path. The vertices is denoted as $S_0^0 \cup S_1^0 \cup S_2^0 \cup \ldots \cup S_\ell^0$, where $S_i^0 \subset S_i$, $S_0^0 = \{s\}$, $S_\ell^0 = \{t\}$.

- Each edge $\overrightarrow{uv}$ of $\overrightarrow{G}_r^0$ has a unique i such that $u \in S_i^0$, and $v \in S_{i+1}^0$. ℓ is the minimal length of augmenting paths.

The essential part of $\overrightarrow{G}_r$ from the point of breadth first search

Definition

Assume that we have a successful search in the residual graph ($t \in S_\ell$). Let $\overrightarrow{G}_r^0$ be the subgraph of the residual graph that contains exactly those vertices and edges that are on a shortest $\overrightarrow{st}$-path. The vertices is denoted as $S_0^0 \cup S_1^0 \cup S_2^0 \cup \ldots \cup S_\ell^0$, where $S_i^0 \subset S_i$, $S_0^0 = \{s\}$, $S_\ell^0 = \{t\}$.

- Each edge $\overrightarrow{uv}$ of $\overrightarrow{G}_r^0$ has a unique i such that $u \in S_i^0$, and $v \in S_{i+1}^0$. ℓ is the minimal length of augmenting paths.
- The sets S_i^0 are called layers.

The essential part of $\overrightarrow{G}_r$ from the point of breadth first search

Definition

Assume that we have a successful search in the residual graph ($t \in S_\ell$). Let $\overrightarrow{G}_r^0$ be the subgraph of the residual graph that contains exactly those vertices and edges that are on a shortest $\overrightarrow{st}$-path. The vertices is denoted as $S_0^0 \cup S_1^0 \cup S_2^0 \cup \ldots \cup S_\ell^0$, where $S_i^0 \subset S_i$, $S_0^0 = \{s\}$, $S_\ell^0 = \{t\}$.

- Each edge $\overrightarrow{uv}$ of $\overrightarrow{G}_r^0$ has a unique i such that $u \in S_i^0$, and $v \in S_{i+1}^0$. ℓ is the minimal length of augmenting paths.
- The sets S_i^0 are called layers. $\overrightarrow{G}_r^0$ is a layered graph.

The essential part of $\overrightarrow{G}_r$ from the point of breadth first search

Definition

Assume that we have a successful search in the residual graph ($t \in S_\ell$). Let $\overrightarrow{G}_r^0$ be the subgraph of the residual graph that contains exactly those vertices and edges that are on a shortest $\overrightarrow{st}$-path. The vertices is denoted as $S_0^0 \cup S_1^0 \cup S_2^0 \cup \ldots \cup S_\ell^0$, where $S_i^0 \subset S_i$, $S_0^0 = \{s\}$, $S_\ell^0 = \{t\}$.

- Each edge $\overrightarrow{uv}$ of $\overrightarrow{G}_r^0$ has a unique i such that $u \in S_i^0$, and $v \in S_{i+1}^0$. ℓ is the minimal length of augmenting paths.
- The sets S_i^0 are called layers. $\overrightarrow{G}_r^0$ is a layered graph.
- Any minimal length augmenting path reaches all layers, following the $S_0^0 \to S_1^0 \to S_3^0 \to \ldots \to S_\ell^0$ order.

Analysis of the Edmonds-Karp algorithm: Phases

Analysis of the Edmonds-Karp algorithm: Phases

Definition: Phases

Analysis of the Edmonds-Karp algorithm: Phases

Definition: Phases

The first augmentation opens the first phase.

Analysis of the Edmonds-Karp algorithm: Phases

Definition: Phases

The first augmentation opens the first phase.

After that, in the case of the actual augmentation we compare the length of actual the augmenting path with the length of the previous one.

Analysis of the Edmonds-Karp algorithm: Phases

Definition: Phases

The first augmentation opens the first phase.

After that, in the case of the actual augmentation we compare the length of actual the augmenting path with the length of the previous one.

If the two lengths are equal, then we continue the phase.

Analysis of the Edmonds-Karp algorithm: Phases

Definition: Phases

The first augmentation opens the first phase.

After that, in the case of the actual augmentation we compare the length of actual the augmenting path with the length of the previous one.

If the two lengths are equal, then we continue the phase. The actual augmentation continues the phase of the previous one.

Analysis of the Edmonds-Karp algorithm: Phases

Definition: Phases

The first augmentation opens the first phase.

After that, in the case of the actual augmentation we compare the length of actual the augmenting path with the length of the previous one.

If the two lengths are equal, then we continue the phase. The actual augmentation continues the phase of the previous one.

If the two lengths are different, then we close the phase.

Analysis of the Edmonds-Karp algorithm: Phases

Definition: Phases

The first augmentation opens the first phase.

After that, in the case of the actual augmentation we compare the length of actual the augmenting path with the length of the previous one.

If the two lengths are equal, then we continue the phase. The actual augmentation continues the phase of the previous one.

If the two lengths are different, then we close the phase. The actual augmentation opens a new phase.

Analysis of the Edmonds-Karp algorithm: Phases

Definition: Phases

The first augmentation opens the first phase.

After that, in the case of the actual augmentation we compare the length of actual the augmenting path with the length of the previous one.

If the two lengths are equal, then we continue the phase. The actual augmentation continues the phase of the previous one.

If the two lengths are different, then we close the phase. The actual augmentation opens a new phase.

The unsuccessful search closes the last phase.

Theorem and its Corollary

Theorem and its Corollary

Theorem (Edmonds–Karp)

(i) In one phase the length of the augmenting paths remain the same. During several phases the length is strictly increasing.

(ii) In one phase the edge set of $\overrightarrow{G}_r^0$ is strictly decreasing.

Theorem and its Corollary

Theorem (Edmonds–Karp)

(i) In one phase the length of the augmenting paths remain the same. During several phases the length is strictly increasing.

(ii) In one phase the edge set of $\overrightarrow{G}_r^0$ is strictly decreasing.

Corollary

A run of the Edmonds-Karp algorithm consist of at most $|V|$ phases.

Theorem and its Corollary

Theorem (Edmonds–Karp)

(i) In one phase the length of the augmenting paths remain the same. During several phases the length is strictly increasing.

(ii) In one phase the edge set of $\overrightarrow{G}_r^0$ is strictly decreasing.

Corollary

A run of the Edmonds-Karp algorithm consist of at most $|V|$ phases. Each phase contains at most $|E|$ augmentations.

Theorem and its Corollary

Theorem (Edmonds–Karp)

(i) In one phase the length of the augmenting paths remain the same. During several phases the length is strictly increasing.

(ii) In one phase the edge set of $\overrightarrow{G}_r^0$ is strictly decreasing.

Corollary

A run of the Edmonds-Karp algorithm consist of at most $|V|$ phases. Each phase contains at most $|E|$ augmentations. Hence the algorithm find the optimal flow after at most $|V| \cdot |E|$ augmentations.

Theorem and its Corollary

Theorem (Edmonds–Karp)

(i) In one phase the length of the augmenting paths remain the same. During several phases the length is strictly increasing.

(ii) In one phase the edge set of $\overrightarrow{G}_r^0$ is strictly decreasing.

Corollary

A run of the Edmonds-Karp algorithm consist of at most $|V|$ phases. Each phase contains at most $|E|$ augmentations. Hence the algorithm find the optimal flow after at most $|V| \cdot |E|$ augmentations.

Specially the running time of the algorithm is

$$|V||E|\mathcal{O}(|E| + |V|) = \mathcal{O}(|V||E|^2).$$

The proof of the theorem: (i)

The proof of the theorem: (i)

One of the bread first search builds the sets S_i's, and finds n augmenting path going through the vertices $v_0 = s, v_1 \in S_i, \ldots, v_{\ell-1} \in S_{\ell-1}, v_\ell = t$.

The proof of the theorem: (i)

One of the bread first search builds the sets S_i's, and finds n augmenting path going through the vertices $v_0 = s, v_1 \in S_i, \ldots, v_{\ell-1} \in S_{\ell-1}, v_\ell = t$.

This is a vertex set of directed path $\overrightarrow{G_r^0}$.

The proof of the theorem: (i)

One of the bread first search builds the sets S_i's, and finds n augmenting path going through the vertices $v_0 = s, v_1 \in S_i, \ldots, v_{\ell-1} \in S_{\ell-1}, v_\ell = t$.

This is a vertex set of directed path $\overrightarrow{G_r^0}$.

We compute δ and update f to f^+.

The proof of the theorem: (i)

One of the bread first search builds the sets S_i's, and finds n augmenting path going through the vertices $v_0 = s, v_1 \in S_i, \ldots, v_{\ell-1} \in S_{\ell-1}, v_\ell = t$.

This is a vertex set of directed path $\overrightarrow{G_r^0}$.

We compute δ and update f to f^+.

δ is chosen such a way that the flow at certain edges (at least at one) will be increased to the capacity or reduced to 0.

The proof of the theorem: (i)

One of the bread first search builds the sets S_i's, and finds n augmenting path going through the vertices $v_0 = s, v_1 \in S_i, \dots, v_{\ell-1} \in S_{\ell-1}, v_\ell = t$.

This is a vertex set of directed path $\overrightarrow{G_r^0}$.

We compute δ and update f to f^+.

δ is chosen such a way that the flow at certain edges (at least at one) will be increased to the capacity or reduced to 0. The next generation of the reduced graph this results some loss, concerning the set of edges.

The proof of the theorem: (i)

One of the bread first search builds the sets S_i's, and finds n augmenting path going through the vertices $v_0 = s, v_1 \in S_i, \ldots, v_{\ell-1} \in S_{\ell-1}, v_\ell = t$.

This is a vertex set of directed path $\overrightarrow{G_r^0}$.

We compute δ and update f to f^+.

δ is chosen such a way that the flow at certain edges (at least at one) will be increased to the capacity or reduced to 0. The next generation of the reduced graph this results some loss, concerning the set of edges.

At the same time some flow might be reduced from the value of capacity, and some flow might be increased from 0.

The proof of the theorem: (i)

One of the bread first search builds the sets S_i's, and finds n augmenting path going through the vertices $v_0 = s, v_1 \in S_i, \ldots, v_{\ell-1} \in S_{\ell-1}, v_\ell = t$.

This is a vertex set of directed path $\overrightarrow{G_r^0}$.

We compute δ and update f to f^+.

δ is chosen such a way that the flow at certain edges (at least at one) will be increased to the capacity or reduced to 0. The next generation of the reduced graph this results some loss, concerning the set of edges.

At the same time some flow might be reduced from the value of capacity, and some flow might be increased from 0. This results some extra edges when generating the next reduced network.

The proof of the theorem: (i), the main observation

The proof of the theorem: (i), the main observation

The Main Observation

The proof of the theorem: (i), the main observation

The Main Observation

(i) The edges where the flow is changed are $\overrightarrow{S_i S_{i+1}}$-edges.

The proof of the theorem: (i), the main observation

The Main Observation

(i) The edges where the flow is changed are $\overrightarrow{S_i S_{i+1}}$-edges.

(ii) The extra edges after generating the new reduced network will be $\overrightarrow{S_{i+1} S_i}$-edges.

The proof of the theorem: (i), the main observation

The Main Observation

(i) The edges where the flow is changed are $\overrightarrow{S_i S_{i+1}}$-edges.

(ii) The extra edges after generating the new reduced network will be $\overrightarrow{S_{i+1} S_i}$-edges.

- Specially we maintain the property $(\star\star)$.

The proof of the theorem: (i), the main observation

The Main Observation

(i) The edges where the flow is changed are $\overrightarrow{S_i S_{i+1}}$-edges.

(ii) The extra edges after generating the new reduced network will be $\overrightarrow{S_{i+1} S_i}$-edges.

- Specially we maintain the property $(\star\star)$.

- Specially the length of the shortest augmenting path can't be decreased.

The proof of the theorem: (ii)

The proof of the theorem: (ii)

- The extra edges won't appear in G_r^0.

The proof of the theorem: (ii)

- The extra edges won't appear in G_r^0.
- The loss in edges will be seen in G_r^0.

Break

The weakness of Edmonds-Karp algorithm

The weakness of Edmonds-Karp algorithm

- For each augmentation they run a new breadth first search.

The weakness of Edmonds-Karp algorithm

- For each augmentation they run a new breadth first search.
- The monotonicity of $\overrightarrow{G}_r^0$ is just a theoretical observation.

The weakness of Edmonds-Karp algorithm

- For each augmentation they run a new breadth first search.
- The monotonicity of $\overrightarrow{G}_r^0$ is just a theoretical observation. It has no algorithmic value.

Dinic's idea

Dinic's idea

Keeps the structure of Edmonds-Karp algorithm.

Dinic's idea

Keeps the structure of Edmonds-Karp algorithm.

At the beginning of each phase compute the graph $\overrightarrow{G_r^0}$.

Dinic's idea

Keeps the structure of Edmonds-Karp algorithm.

At the beginning of each phase compute the graph $\overrightarrow{G_r^0}$.

In a phase don't repeat this computation.

Dinic's idea

Keeps the structure of Edmonds-Karp algorithm.

At the beginning of each phase compute the graph $\overrightarrow{G_r^0}$.

In a phase don't repeat this computation. In stead of that "only" update the previous graph $\overrightarrow{G_r^0}$.

Dinic's idea

Keeps the structure of Edmonds-Karp algorithm.

At the beginning of each phase compute the graph $\overrightarrow{G_r^0}$.

In a phase don't repeat this computation. In stead of that "only" update the previous graph $\overrightarrow{G_r^0}$.

Knowing the graph $\overrightarrow{G_r^0}$ we can compute a directed $\overrightarrow{st}$ path/augmenting path in $\mathcal{O}(|V|)$ time.

Computing $\overrightarrow{G}_r^0$

Computing $\overrightarrow{G}_r^0$

- Run the breadth first search algorithms until S_ℓ is found ($t \in S_\ell$).

Computing $\overrightarrow{G}_r^0$

- Run the breadth first search algorithms until S_ℓ is found ($t \in S_\ell$).
- Let $\overrightarrow{G}_r^{00}$ be the graph formed by the vertex set $S_0 \cup S_1 \cup \ldots \cup S_\ell$ and edges of type $\overrightarrow{S_i S_{i+1}}$.

Computing $\overrightarrow{G}_r^0$

- Run the breadth first search algorithms until S_ℓ is found ($t \in S_\ell$).
- Let $\overrightarrow{G}_r^{00}$ be the graph formed by the vertex set $S_0 \cup S_1 \cup \ldots \cup S_\ell$ and edges of type $\overrightarrow{S_i S_{i+1}}$.
- Run a "backward" breadth first search starting from t in $\overrightarrow{G}_r^{00}$.

Computing $\overrightarrow{G}_r^0$

- Run the breadth first search algorithms until S_ℓ is found ($t \in S_\ell$).
- Let $\overrightarrow{G}_r^{00}$ be the graph formed by the vertex set $S_0 \cup S_1 \cup \ldots \cup S_\ell$ and edges of type $\overrightarrow{S_i S_{i+1}}$.
- Run a "backward" breadth first search starting from t in $\overrightarrow{G}_r^{00}$.
- Let $S_i^0 \subset S_i$ the set of labeled vertices.

Computing $\overrightarrow{G}_r^0$

- Run the breadth first search algorithms until S_ℓ is found ($t \in S_\ell$).
- Let $\overrightarrow{G}_r^{00}$ be the graph formed by the vertex set $S_0 \cup S_1 \cup \ldots \cup S_\ell$ and edges of type $\overrightarrow{S_i S_{i+1}}$.
- Run a "backward" breadth first search starting from t in $\overrightarrow{G}_r^{00}$.
- Let $S_i^0 \subset S_i$ the set of labeled vertices. ${}^0_\ell = \{t\}$. $\overrightarrow{G}_r^0$ is formed by the vertex set $S_0^0 \cup S_1^0 \cup \ldots \cup S_\ell^0$ and edges of type $\overrightarrow{S_i^0 S_{i+1}^0}$.

Computing $\overrightarrow{G}_r^0$

- Run the breadth first search algorithms until S_ℓ is found ($t \in S_\ell$).
- Let $\overrightarrow{G}_r^{00}$ be the graph formed by the vertex set $S_0 \cup S_1 \cup \ldots \cup S_\ell$ and edges of type $\overrightarrow{S_i S_{i+1}}$.
- Run a "backward" breadth first search starting from t in $\overrightarrow{G}_r^{00}$.
- Let $S_i^0 \subset S_i$ the set of labeled vertices. ${}_\ell^0 = \{t\}$. $\overrightarrow{G}_r^0$ is formed by the vertex set $S_0^0 \cup S_1^0 \cup \ldots \cup S_\ell^0$ and edges of type $\overrightarrow{S_i^0 S_{i+1}^0}$.
-

Lemma of Dinic

$\overrightarrow{G}_r^0$ can be computed in time $\mathcal{O}(|E|)$.

Computing $\overrightarrow{G}_r^0$

- Run the breadth first search algorithms until S_ℓ is found ($t \in S_\ell$).
- Let $\overrightarrow{G}_r^{00}$ be the graph formed by the vertex set $S_0 \cup S_1 \cup \ldots \cup S_\ell$ and edges of type $\overrightarrow{S_i S_{i+1}}$.
- Run a "backward" breadth first search starting from t in $\overrightarrow{G}_r^{00}$.
- Let $S_i^0 \subset S_i$ the set of labeled vertices. ${}^0_\ell = \{t\}$. $\overrightarrow{G}_r^0$ is formed by the vertex set $S_0^0 \cup S_1^0 \cup \ldots \cup S_\ell^0$ and edges of type $\overrightarrow{S_i^0 S_{i+1}^0}$.
-

Lemma of Dinic

$\overrightarrow{G}_r^0$ can be computed in time $\mathcal{O}(|E|)$.

We assume that $\overrightarrow{G}$ is connected (the undirected sense), hence $|E| \geq |V| - 1$.

Theorem of Dinic

Theorem of Dinic

Theorem of Dinic

Within a phase all the updates of $\overrightarrow{G_r^0}$ can be done in time $\mathcal{O}(|E|)$.

Theorem of Dinic

Theorem of Dinic

Within a phase all the updates of $\overrightarrow{G_r^0}$ can be done in time $\mathcal{O}(|E|)$.

We discuss the proof in a later unit.

The profit

The profit

We know that the number of phases is at most $|V|$.

The profit

We know that the number of phases is at most $|V|$.

In one phase we have at most $|E|$ augmentation.

The profit

We know that the number of phases is at most $|V|$.

In one phase we have at most $|E|$ augmentation.

The cost of building of the graph $\overrightarrow{G}_r^0$ is $\mathcal{O}(|E|)$ in each phase.

The profit

We know that the number of phases is at most $|V|$.

In one phase we have at most $|E|$ augmentation.

The cost of building of the graph $\overrightarrow{G}_r^0$ is $\mathcal{O}(|E|)$ in each phase.

Finding each augmenting path and each augmentation requires time $\mathcal{O}(|V|)$.

The profit

We know that the number of phases is at most $|V|$.

In one phase we have at most $|E|$ augmentation.

The cost of building of the graph $\overrightarrow{G}_r^0$ is $\mathcal{O}(|E|)$ in each phase.

Finding each augmenting path and each augmentation requires time $\mathcal{O}(|V|)$.

All the update costs for $\overrightarrow{G}_r^0$ requires time $\mathcal{O}(|E|)$ in each phase.

The profit

We know that the number of phases is at most $|V|$.

In one phase we have at most $|E|$ augmentation.

The cost of building of the graph $\overrightarrow{G}_r^0$ is $\mathcal{O}(|E|)$ in each phase.

Finding each augmenting path and each augmentation requires time $\mathcal{O}(|V|)$.

All the update costs for $\overrightarrow{G}_r^0$ requires time $\mathcal{O}(|E|)$ in each phase.

The cost of the whole algorithm is

$$|V|\left(\mathcal{O}(|E|)+\mathcal{O}(|E|)\cdot\mathcal{O}(|V|)+\mathcal{O}(|E|)\right)=\mathcal{O}(|V|^2|E|).$$

Summary

Summary

Theorem of Dinic

The Dinic' algorithm find the optimal flow in

$$\mathcal{O}(|V|^2|E|)$$

steps.

That"s the end!

Thank you for your attention!