Introduction to Algorithms
Lecture 12 MAX FLOW

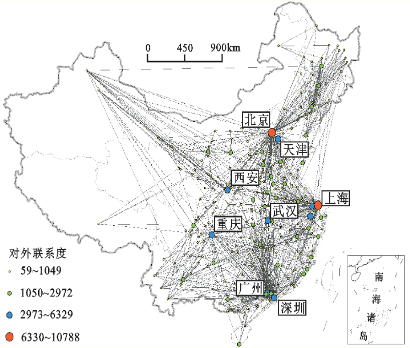
Xue Chen
xuechen1989@ustc.edu.cn
2025 spring in

Outline

Introduction

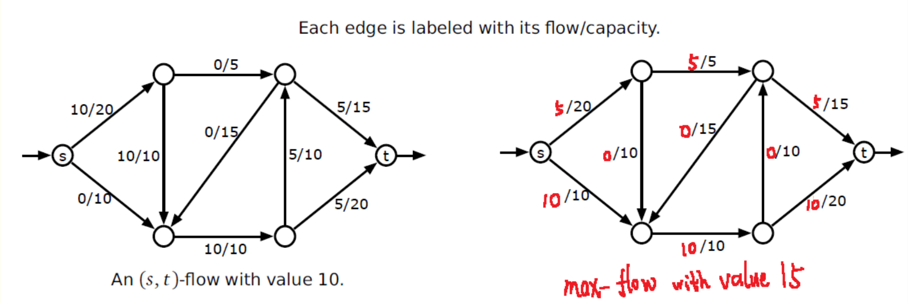
Flow networks are fundamental problems in many areas

1. Resource allocation and scheduling
2. Network routing
3. Traffic control



Overview

Basic problem: Given each edge's limit, schedule the max amount of flows from s to t

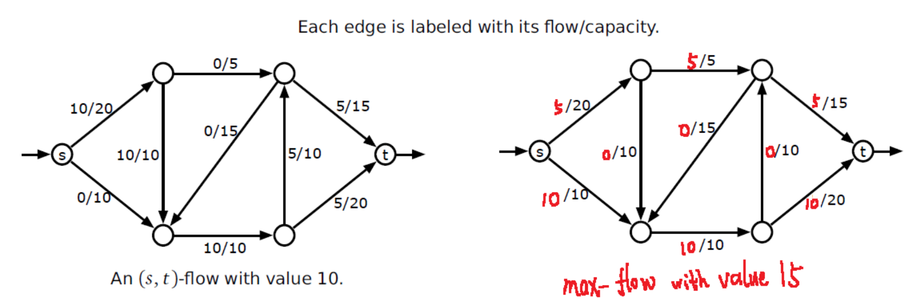


Lots of interesting algorithms:

1. How to compute a flow? How to prove the flow is max?

Overview

Basic problem: Given each edge's limit, schedule the max amount of flows from s to t

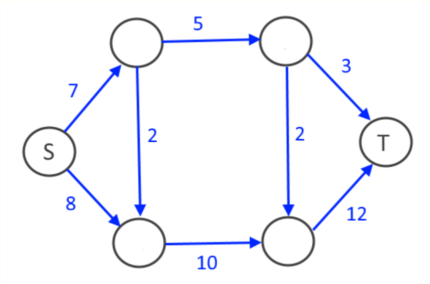


Lots of interesting algorithms:

1. How to compute a flow? How to prove the flow is max?
2. Duality: Max-Flow Min-Cut Theorem
3. Min-cut has lots of applications in data science ...

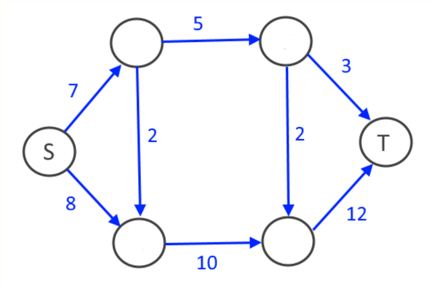
Formal Definition

A flow network $G = (V, E)$ is a directed graph where each edge (u, v) has a capacity $c(u, v) \geqslant 0$.



Formal Definition

A flow network $G = (V, E)$ is a directed graph where each edge (u, v) has a capacity $c(u, v) \geqslant 0$.

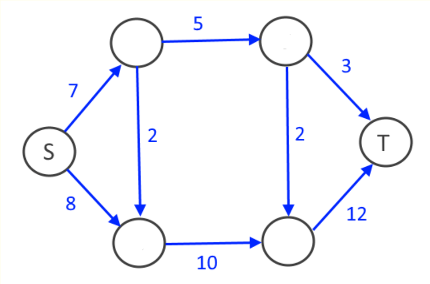


A flow $f : V \times V \to \mathbb{R}$ (with directions) satisfies:

1. Capacity constraint: $0 \leqslant f(u, v) \leqslant c(u, v)$
2. Flow conservation: For each $u \in V \setminus \{s, t\}$,
   flow-in $\sum_v f(v, u) =$ flow-out $\sum_v f(u, v)$

Formal Definition

A flow network $G = (V, E)$ is a directed graph where each edge (u, v) has a capacity $c(u, v) \geqslant 0$.

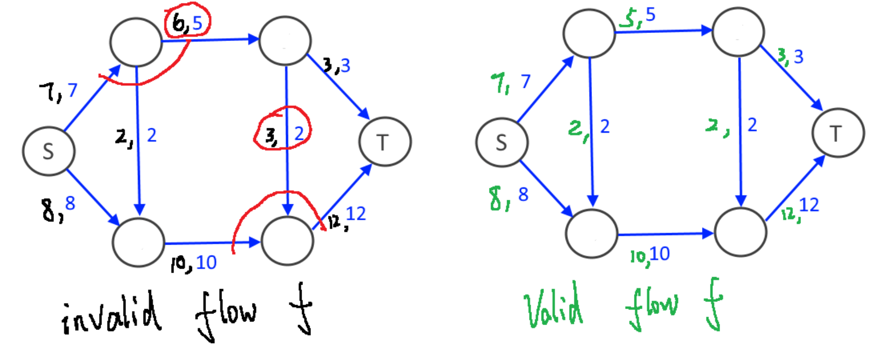


A flow $f : V \times V \to \mathbb{R}$ (with directions) satisfies:

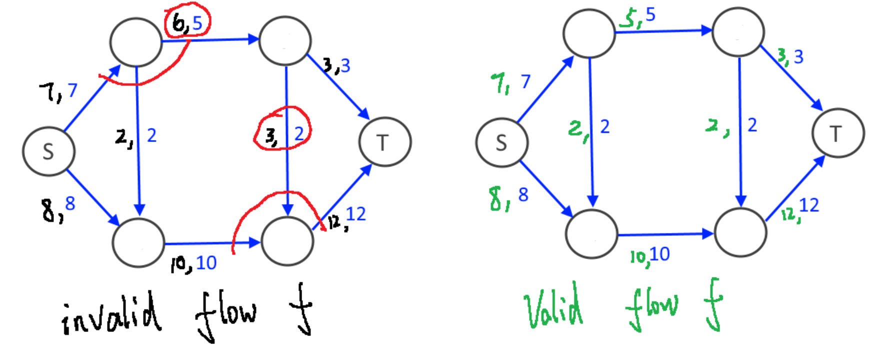
1. Capacity constraint: $0 \leqslant f(u, v) \leqslant c(u, v)$
2. Flow conservation: For each $u \in V \setminus \{s, t\}$,
   flow-in $\sum_v f(v, u) =$ flow-out $\sum_v f(u, v)$

Moreover, define the value of f as $|f| := \sum_v f(s, v) = \sum_v f(v, t)$.

Examples


S
T
7, 7
8, 8
6, 5
2, 2
3, 2
3, 3
10, 10
12, 12
invalid flow f
S
T
7, 7
8, 8
5, 5
2, 2
2, 2
3, 3
10, 10
12, 12
Valid flow f

Examples



Overview:

1. Ford-Fulkerson Method: Residue graph and augmenting path
2. Correctness: Max-Flow Min-Cut Theorem
3. Greedy Methods control time

Outline

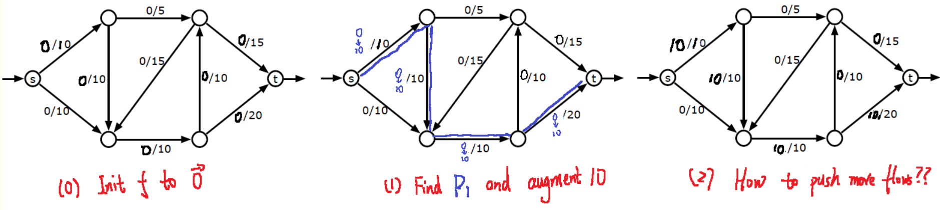
A general paradigm

procedure FORD-FULKERSON(G)
    $f : V \times V \to R$
    while $\exists$ augmenting paths in residual network G_f **do**
        Pick such a path P
        Augment/Push flow f on P

A general paradigm

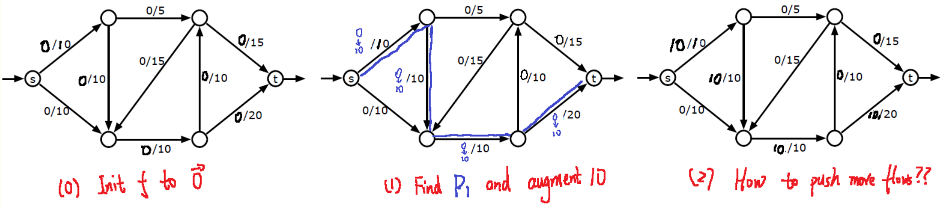
---

procedure FORD-FULKERSON(G)
    $f : V \times V \to R$
    while $\exists$ augmenting paths in residual network G_f **do**
        Pick such a path P
        Augment/Push flow f on P



A general paradigm

procedure FORD-FULKERSON(G)
  $f : V \times V \to R$
  while $\exists$ augmenting paths in residual network G_f **do**
    Pick such a path P
    Augment/Push flow f on P



1. However, the naive implementation does not work
2. Define augmenting path and residual graph formally

Capacities in Residual Graph G_f

Roughly, G_f consists of all edges with a capacitie$>$flow.

Two Types of Edges in G_f: For an original $(u, v) \in E$,

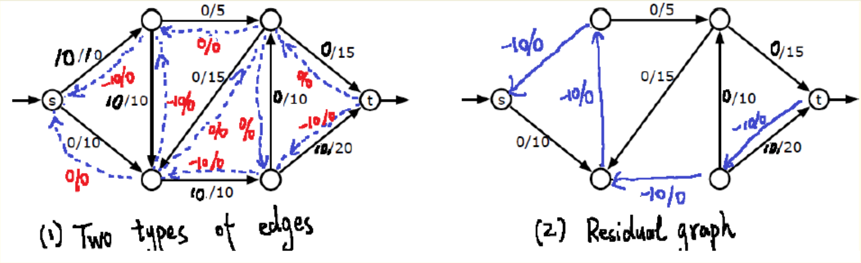
1. its residual capacity $c_f(u, v) = c(u, v) - f(u, v)$
2. its reverse has capacity $c_f(v, u) = f(u, v)$

Capacities in Residual Graph G_f

Roughly, G_f consists of all edges with a capacitie>flow.

Two Types of Edges in G_f: For an original $(u, v) \in E$,

1. its residual capacity $c_f(u, v) = c(u, v) - f(u, v)$
2. its reverse has capacity $c_f(v, u) = f(u, v)$

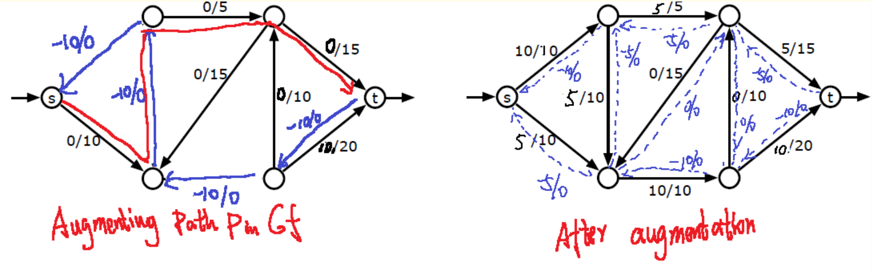


My implementation:

1. Set reverset capacity $c(v, u) = 0$ for any $(u, v) \in E$ but guarantee $f(v, u) = -f(u, v)$ for all edges at any moment
2. Then $c_f(u, v) = c(u, v) - f(u, v)$ for any u and v despite the direction

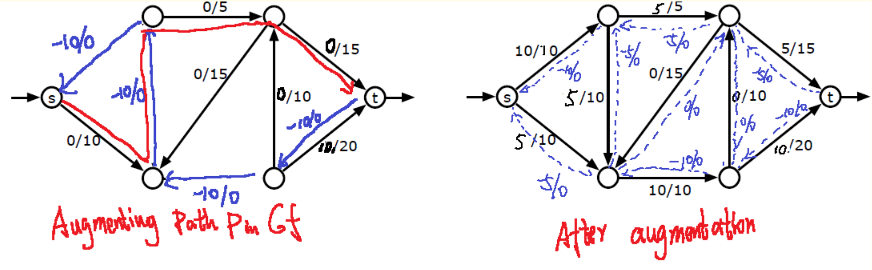
Augmenting Paths

An augmenting path P is a simple path $s \rightsquigarrow t$ in residual graph G_f



Augmenting Paths

An augmenting path P is a simple path $s \rightsquigarrow t$ in residual graph G_f

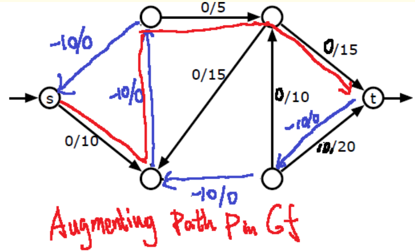


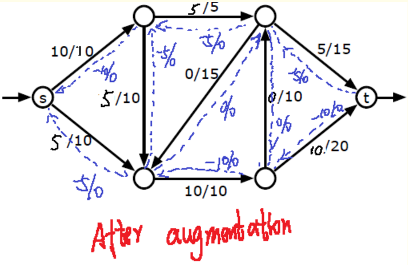
More about Augmenting a path

1. Define P's residual capacity as $c_f(P) = \min\{c_f(u, v) = c(u, v) - f(u, v) : (u, v) \in P\}$, i.e., max residual along P

Augmenting Paths

An augmenting path P is a simple path $s \rightsquigarrow t$ in residual graph G_f





More about Augmenting a path

1. Define P's residual capacity as $c_f(P) = \min\{c_f(u,v) = c(u,v) - f(u,v) : (u,v) \in P\}$, i.e., max residual along P
2. Then augment a flow of amount $c_f(P)$ along P
3. Increase $|f|$ by $c_f(P)$

Analysis

procedure BASIC-FORD-FULKERSON(G)
  $f : V \times V \to R$
  while $\exists$ augmenting paths in the residual network G_f **do**
    Pick such a path P
    Augment/Push flow f on P

Next

1. Correctness: Why does it find a maximum flow?
2. Running time: Is it polynomial in $n \cdot m$?

Outline

Introduction

Definition

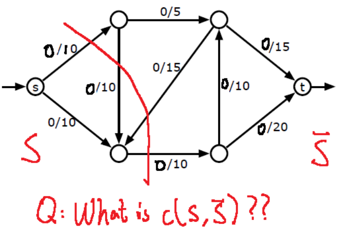
1. General Cut: Given a graph $G = (V, E)$, a cut is a partition $(S, \overline{S})$ where $S \subsetneq V$ and $S \neq \emptyset$
2. Cuts in flow network: only consider cut S where source $s \in S$ and sink $t \notin S$

Introduction

Definition

1. General Cut: Given a graph $G = (V, E)$, a cut is a partition $(S, \overline{S})$ where $S \subsetneq V$ and $S \neq \emptyset$
2. Cuts in flow network: only consider cut S where source $s \in S$ and sink $t \notin S$
3. Cut Capacity: Given edge capacity c on E,

$$c(S, \overline{S}) = \sum_{(u,v) \in E : u \in S, v \in \overline{S}} c(u, v)$$

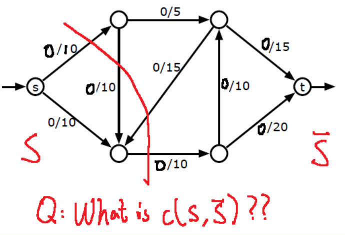


Introduction

Definition

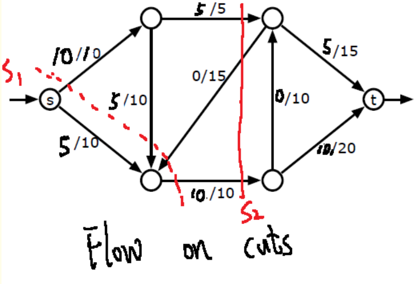
1. General Cut: Given a graph $G = (V, E)$, a cut is a partition $(S, \overline{S})$ where $S \subsetneq V$ and $S \neq \emptyset$
2. Cuts in flow network: only consider cut S where source $s \in S$ and sink $t \notin S$
3. Cut Capacity: Given edge capacity c on E,

$$c(S, \overline{S}) = \sum_{(u,v) \in E : u \in S, v \in \overline{S}} c(u, v)$$

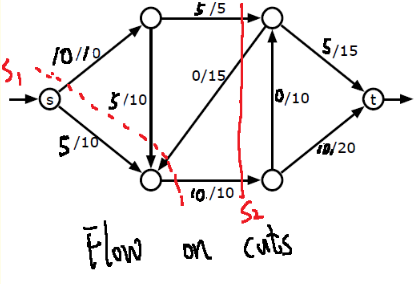


1. Graph cut is a fundamental object in CS: data mining, social networks, ...
2. Today, only consider S with minimum capacity — called min-cut

Basic Property



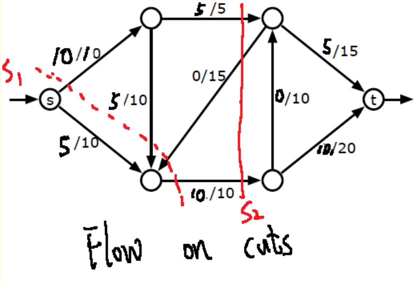
Basic Property



1. In contrast to $c(S, \overline{S}) = \sum_{(u,v) \in E \cap S \times \overline{S}} c(u, v)$, define the flow f on cut $(S, \overline{S})$ as

$$f(S, \overline{S}) = \sum_{(u,v) \in E \cap S \times \overline{S}} f(u, v) - \sum_{(u,v) \in E \cap \overline{S} \times S} f(u, v)$$

Basic Property

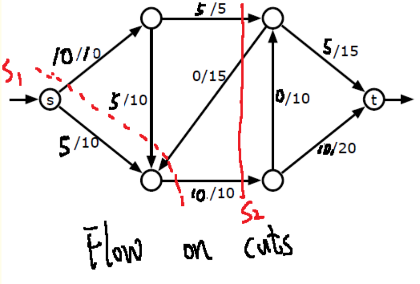


1. In contrast to $c(S,\overline{S}) = \sum_{(u,v)\in E\cap S\times\overline{S}} c(u,v)$, define the flow f on cut $(S,\overline{S})$ as

$$f(S,\overline{S}) = \sum_{(u,v)\in E\cap S\times\overline{S}} f(u,v) - \sum_{(u,v)\in E\cap \overline{S}\times S} f(u,v)$$

2. Lemma 26.5 in CLRS: For any flow f and cut $(S,\overline{S})$, $f(S,\overline{S}) = |f|$

Basic Property



1. In contrast to $c(S, \overline{S}) = \sum_{(u,v) \in E \cap S \times \overline{S}} c(u, v)$, define the flow f on cut $(S, \overline{S})$ as

$$f(S, \overline{S}) = \sum_{(u,v) \in E \cap S \times \overline{S}} f(u, v) - \sum_{(u,v) \in E \cap \overline{S} \times S} f(u, v)$$

2. Lemma 26.5 in CLRS: For any flow f and cut $(S, \overline{S})$, $f(S, \overline{S}) = |f|$
3. Corollary 26.6 in CLRS: $|f| \leqslant c(S, \overline{S})$ for any f and S

Main Theorem

Max-Flow Min-Cut Theorem (THM 26.6 in CLRS)

If f is a flow, the following 3 conditions are equivalent

1. f is a maximum flow
2. No augmenting path in G_f
3. $|f| = c(S, \overline{S})$ for some cut S

Main Theorem

Max-Flow Min-Cut Theorem (THM 26.6 in CLRS)

If f is a flow, the following 3 conditions are equivalent

1. f is a maximum flow
2. No augmenting path in G_f
3. $|f| = c(S, \overline{S})$ for some cut S

- $(1) \Rightarrow (2)$: Otherwise $\exists P$ s.t. augmenting P makes a larger flow

Main Theorem

Max-Flow Min-Cut Theorem (THM 26.6 in CLRS)

If f is a flow, the following 3 conditions are equivalent

1. f is a maximum flow
2. No augmenting path in G_f
3. $|f| = c(S, \overline{S})$ for some cut S

- $(1) \Rightarrow (2)$: Otherwise $\exists P$ s.t. augmenting P makes a larger flow
- $(2) \Rightarrow (3)$: Consider $S = \{v : s \rightsquigarrow v \text{ in } G_f\}$ after the Big while loop

Main Theorem

Max-Flow Min-Cut Theorem (THM 26.6 in CLRS)

If f is a flow, the following 3 conditions are equivalent

1. f is a maximum flow
2. No augmenting path in G_f
3. $|f| = c(S, \overline{S})$ for some cut S

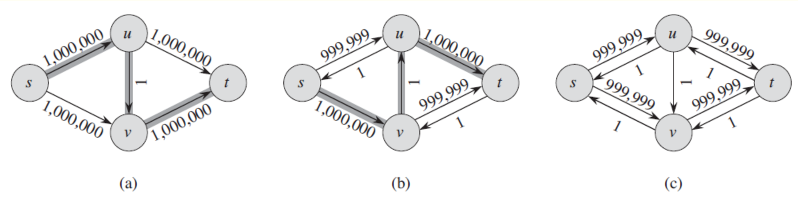
- $(1) \Rightarrow (2)$: Otherwise $\exists P$ s.t. augmenting P makes a larger flow
- $(2) \Rightarrow (3)$: Consider $S = \{v : s \rightsquigarrow v \text{ in } G_f\}$ after the Big while loop
- $(3) \Rightarrow (1)$: from Corollary 26.6, $|f| \leqslant c(S, \overline{S})$ for any S.

Outline

Time Complexity

One more issue

Basic version of Ford-Fulkerson Algorithm does not guarantee a polynomial-time.

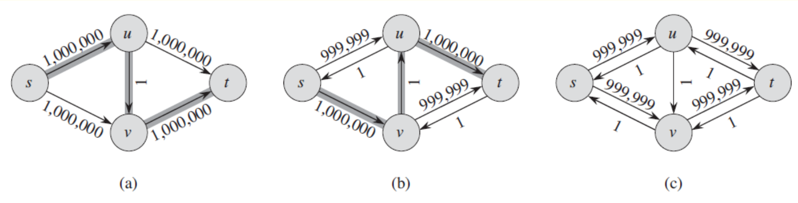


Example: Figure 26.7 from CLRS

Time Complexity

One more issue

Basic version of Ford-Fulkerson Algorithm does not guarantee a polynomial-time.

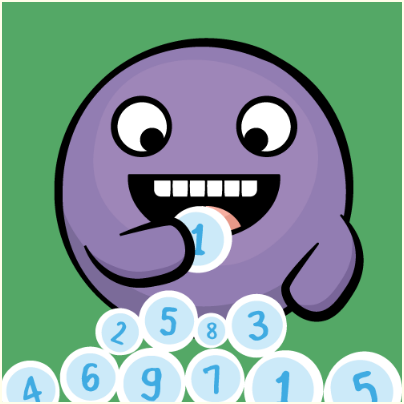


Example: Figure 26.7 from CLRS

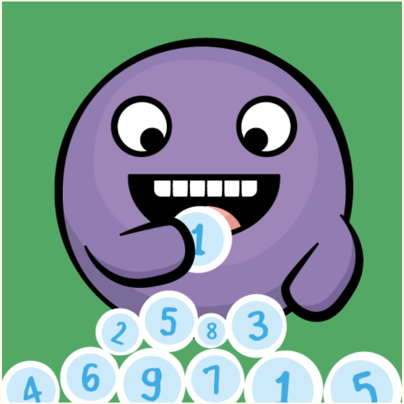
Question

How to improve running time?

Two greedy approaches

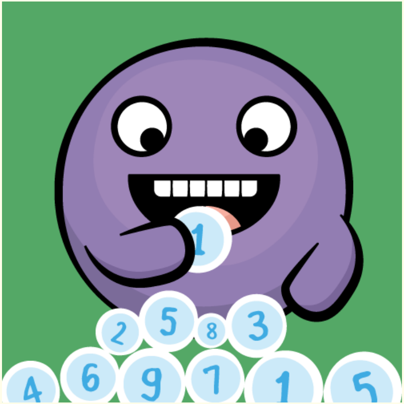


Two greedy approaches



1. Finding the shortest augmenting path (i.e. with fewest edges) leads to at most nm paths, called Edmonds-Karp

Two greedy approaches



1. Finding the shortest augmenting path (i.e. with fewest edges) leads to at most nm paths, called Edmonds-Karp
2. Finding the fattest augmenting path (i.e. with largest residual) leads to at most $m \cdot \log |f^*|$ paths

Greedy I

---

procedure EDMONDS-KARP(G)

  $f : V \times V \to R$

  while $\exists$ augmenting paths in the residual network G_f **do**

    Find augmenting path P with fewest edges

        //Question: How to do it? Time?

    Augment/Push flow f on P

Greedy I

procedure EDMONDS-KARP(G)

    $f : V \times V \rightarrow R$

    while $\exists$ augmenting paths in the residual network G_f **do**

        Find augmenting path P with fewest edges

            //Question: How to do it? Time?

        Augment/Push flow f on P

Theorem 26.7 in CLRS

For any v, the shortest-path distance $\delta_f(s, v)$ in the residual network G_f increases monotonically with each flow.

Analysis of Greedy I

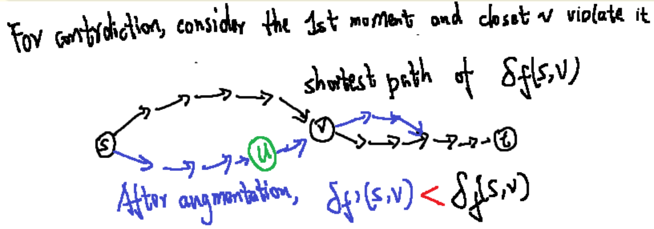
Theorem 26.7 in CLRS

For any v, the shortest-path distance $\delta_f(s, v)$ in the residual network G_f increases monotonically with each flow.

Analysis of Greedy I

Theorem 26.7 in CLRS

For any v, the shortest-path distance $\delta_f(s, v)$ in the residual network G_f increases monotonically with each flow.

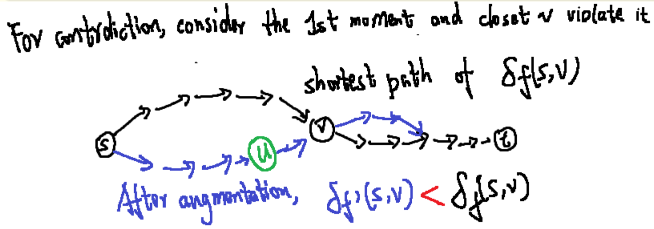


1. By def, $\delta_{f'}(s, u) = \delta_{f'}(s, v) - 1$
2. By our choice of v, u is good $\Rightarrow \delta_{f'}(s, u) \geqslant \delta_f(s, u)$

Analysis of Greedy I

Theorem 26.7 in CLRS

For any v, the shortest-path distance $\delta_f(s, v)$ in the residual network G_f increases monotonically with each flow.



1. By def, $\delta_{f'}(s, u) = \delta_{f'}(s, v) - 1$
2. By our choice of v, u is good $\Rightarrow \delta_{f'}(s, u) \geqslant \delta_f(s, u)$
3. Key claim: $(u, v) \notin G_f$

Analysis of Greedy I

Theorem 26.7 in CLRS

For any v, the shortest-path distance $\delta_f(s, v)$ in the residual network G_f increases monotonically with each flow.

1. By def, $\delta_{f'}(s, u) = \delta_{f'}(s, v) - 1$
2. By our choice of v, u is good $\Rightarrow \delta_{f'}(s, u) \geqslant \delta_f(s, u)$
3. Key claim: $(u, v) \notin G_f$
4. Only way that $(u, v) \notin G_f$ but $(u, v) \in G_{f'}$ is that the shortest augmenting path in G_f has the reverse (v, u) — leads to a contradiction

Analysis of Greedy I

Theorem 26.7 in CLRS

For any v, the shortest-path distance $\delta_f(s, v)$ in the residual network G_f increases monotonically with each flow.

Theorem 26.8 in CLRS

The Edmonds-Karp ALGO augments at most $O(nm)$ paths.

Analysis of Greedy II

procedure GREEDYII(G)

$f : V \times V \to R$

while $\exists$ augmenting paths in the residual network G_f **do**

Find augmenting path P with largest residual capacity

//Question: How to do it? Time?

Augment/Push flow f on P

Analysis of Greedy II

---

procedure GREEDYII(G)

$f : V \times V \to R$

while $\exists$ augmenting paths in the residual network G_f **do**

Find augmenting path P with largest residual capacity

//Question: How to do it? Time?

Augment/Push flow f on P

THM: Running time of Greedy II

The above greedy approaches finds at most $m \cdot \ln |f^*|$ paths where f^* is the amount of max-flow.

Analysis of Greedy II

THM: Running time of Greedy II

The above greedy approaches finds at most $m \log |f^*|$ paths where f^* is the amount of max-flow.

Analysis of Greedy II

THM: Running time of Greedy II

The above greedy approaches finds at most $m \log |f^*|$ paths where f^* is the amount of max-flow.

1. Given f^*, consider the best way to decompose it — how many augmenting paths?

Analysis of Greedy II

THM: Running time of Greedy II

The above greedy approaches finds at most $m \log |f^*|$ paths where f^* is the amount of max-flow.

1. Given f^*, consider the best way to decompose it — how many augmenting paths?
2. Consider each unit flow in f^* as an element, reformulate the problem as set cover
3. Question: What are the subsets here?

Analysis of Greedy II

THM: Running time of Greedy II

The above greedy approaches finds at most $m \log |f^*|$ paths where f^* is the amount of max-flow.

1. Given f^*, consider the best way to decompose it — how many augmenting paths?
2. Consider each unit flow in f^* as an element, reformulate the problem as set cover
3. Question: What are the subsets here?
4. Recall that the greedy method in set cover needs $OPT \cdot \log |U|$ where OPT is the best solution and $|U|$ is # elements

Summary

1. Ford-Fulkerson Method
2. Residue graphs and augmenting paths: reverse edges!

Summary

1. Ford-Fulkerson Method
2. Residue graphs and augmenting paths: reverse edges!
3. Two greedy algorithms implement Ford-Fulkerson method

Summary

1. Ford-Fulkerson Method
2. Residue graphs and augmenting paths: reverse edges!
3. Two greedy algorithms implement Ford-Fulkerson method
4. Many extensions: Max-Matching, Min-cost Max-Flow, ...

Questions?