Advanced Core in Algorithm Design #9
算法設計要論 第 9 回

Yasushi Kawase
河瀬 康志

Dec. 7th, 2021

last update: 1:10pm, December 10, 2021

Schedule

Lec. #	Date	Topics
1	10/5	Introduction, Stable matching
2	10/12	Basics of Algorithm Analysis, Graphs
3	10/19	Greedy Algorithms (1/2)
4	10/26	Greedy Algorithms (2/2)
5	11/2	Divide and Conquer (1/2)
6	11/9	Divide and Conquer (2/2)
7	11/16	Dynamic Programming (1/2)
8	11/30	Dynamic Programming (2/2)
9	12/7	Network Flow (1/2)
10	12/14	Network Flow (2/2)
11	12/21	NP and Computational Intractability
12	1/4	Approximation Algorithms (1/2)
13	1/11	Approximation Algorithms (2/2)
14	1/18	Final Examination

Outline

1. Max-flow and Min-cut Problems
2. Augmenting path algorithm
3. Capacity-scaling algorithm

Flow Network

Flow network (G, s, t, c)

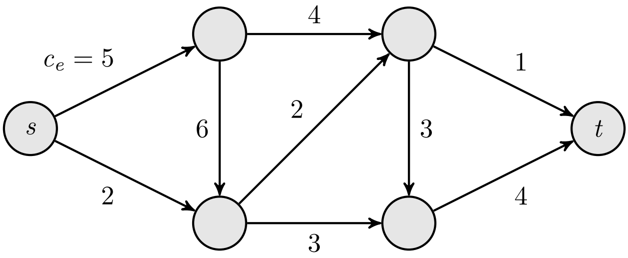
- directed graph $G = (V, E)$ with source $s \in V$ and sink $t \in V$
- capacity c_e for each $e \in E$

s–t flow f

Capacity constraint $0 \leq f_e \leq c_e$ for all $e \in E$

Conservation of flows $\sum_{u:\, e=(u,v)\in E} f_e = \sum_{u:\, (v,u)\in E} f_e$ $(\forall v \in V \setminus \{s, t\})$

Example



Flow Network

Flow network (G, s, t, c)

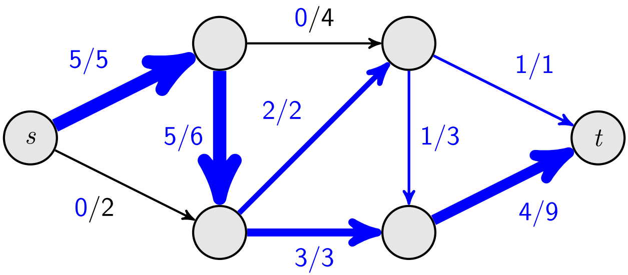
- directed graph $G = (V, E)$ with source $s \in V$ and sink $t \in V$
- capacity c_e for each $e \in E$

s–t flow f

Capacity constraint $0 \leq f_e \leq c_e$ for all $e \in E$

Conservation of flows $\sum_{u\colon e=(u,v)\in E} f_e = \sum_{u\colon (v,u)\in E} f_e$ $(\forall v \in V \setminus \{s, t\})$

Example



s–t flow

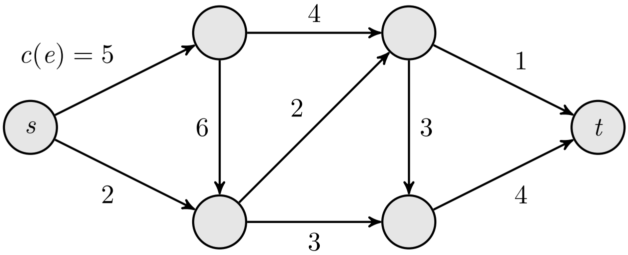
Max-flow problem

Problem

- Input: flow network (G, s, t, c)
- Goal: find an s–t flow of maximum value $\mathrm{val}(f)$

$\sum_{v\colon e=(s,v)\in E} f_e - \sum_{v\colon e=(v,s)\in E} f_e$

Example



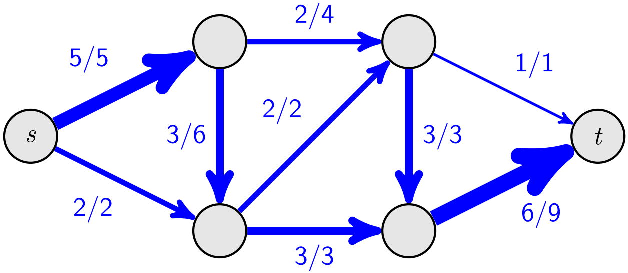
Max-flow problem

Problem

- Input: flow network (G, s, t, c)
- Goal: find an s–t flow of maximum value $\mathrm{val}(f)$

$\sum_{v\colon e=(s,v)\in E} f_e - \sum_{v\colon e=(v,s)\in E} f_e$

Example



$\mathrm{val}(f) = 7$

Min-cut Problem

s–t cut

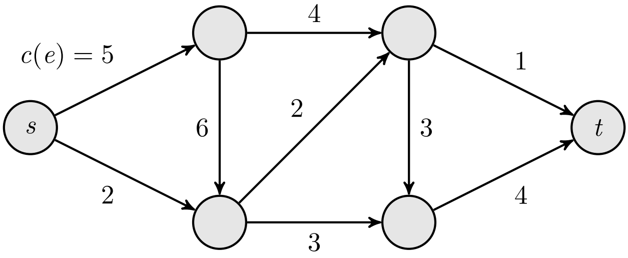
a partition (S, T) of the vertices with $s \in S$ and $t \in T$

Problem

- Input: flow network (G, s, t, c)
- Goal: find an s–t cut of minimum capacity $\mathrm{cap}(S)$

$\sum_{e=(u,v)\in E:\, u\in S,\, v\notin S} c_e$

Example



Min-cut Problem

s–t cut

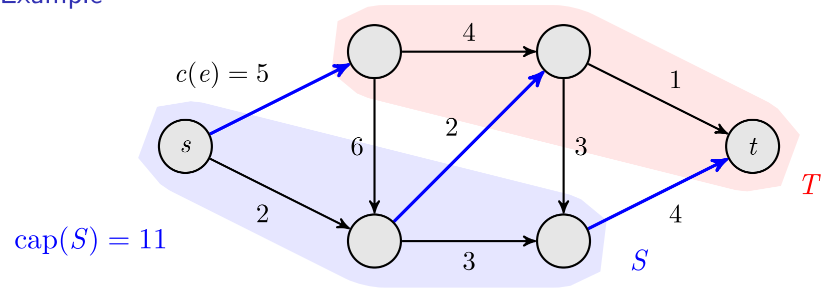
a partition (S, T) of the vertices with $s \in S$ and $t \in T$

Problem

- Input: flow network (G, s, t, c)
- Goal: find an s–t cut of minimum capacity $\mathrm{cap}(S)$

$\sum_{e=(u,v)\in E:\, u\in S,\, v\notin S} c_e$

Example



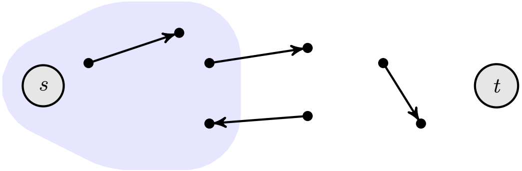
Weak duality

Lemma

$\mathrm{val}(f) \leq \mathrm{cap}(S)$ for any flow f and cut (S, T)

Proof

$$
\begin{aligned}
\mathrm{val}(f) &= \sum_{v \in S} \Bigl[\overbrace{\textstyle\sum_{u:\, e=(v,u) \in E} f_e}^{\text{flow out of } v} - \overbrace{\textstyle\sum_{u:\, e=(u,v) \in E} f_e}^{\text{flow into } v} \Bigr] \\
&= \sum_{e:\ \text{out of } S} f_e - \sum_{e:\ \text{into } S} f_e \\
&\leq \sum_{e:\ \text{out of } S} f_e \leq \sum_{e:\ \text{out of } S} c_e = c(S)
\end{aligned}
$$



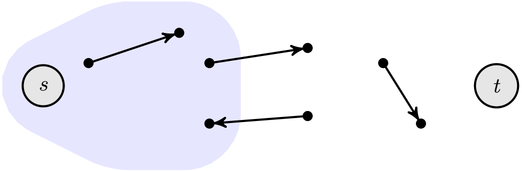
Weak duality

Lemma

$\mathrm{val}(f) \le \mathrm{cap}(S)$ for any flow f and cut (S, T)

Proof

$$
\begin{aligned}
\mathrm{val}(f) &= \sum_{v \in S} \Bigl[\overbrace{\textstyle\sum_{u:\ e=(v,u)\in E} f_e}^{\text{flow out of } v} - \overbrace{\textstyle\sum_{u:\ e=(u,v)\in E} f_e}^{\text{flow into } v} \Bigr] \\
&= \sum_{e:\ \text{out of } S} f_e - \sum_{e:\ \text{into } S} f_e \\
&\le \sum_{e:\ \text{out of } S} f_e \le \sum_{e:\ \text{out of } S} c_e = c(S)
\end{aligned}
$$



we will see $\max_f \mathrm{val}(f) = \min_{(S,T)} \mathrm{cap}(S)$

Outline

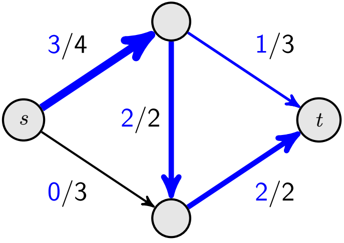
Residual network

Residual network (G_f, s, t, c_f) of G w.r.t. f

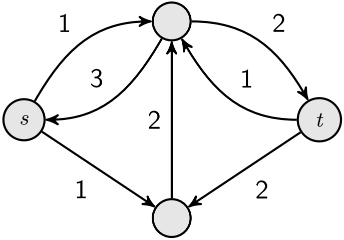
reverse edge of e

residual graph $G_f = (V, E_f)$, $E_f = \{e \mid e \in E,\ f_e < c_e\} \cup \{\bar{e} \mid e \in E,\ f_e > 0\}$

residual capacity $c_f(e) = \begin{cases} c_e - f_e & \text{if } e \in E \\ f_e & \text{if } \bar{e} \in E \end{cases}$



original network and flow

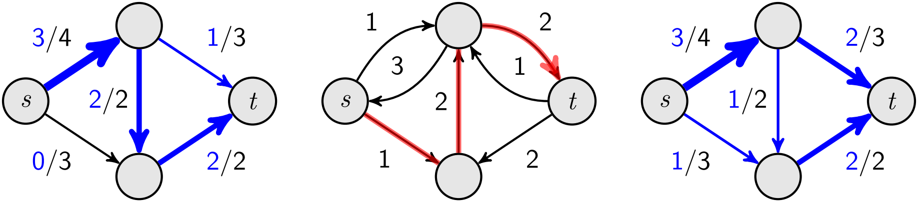


residual network

Augmenting path

- augmenting path: a simple s–t path in the residual network G_f
- bottleneck capacity: the minimum residual capacity of

bottleneck capacity $= 1$



original network and flow

residual network and augmenting path

new flow

Ford–Fulkerson algorithm

$\mathrm{Augment}(f, c, P)$

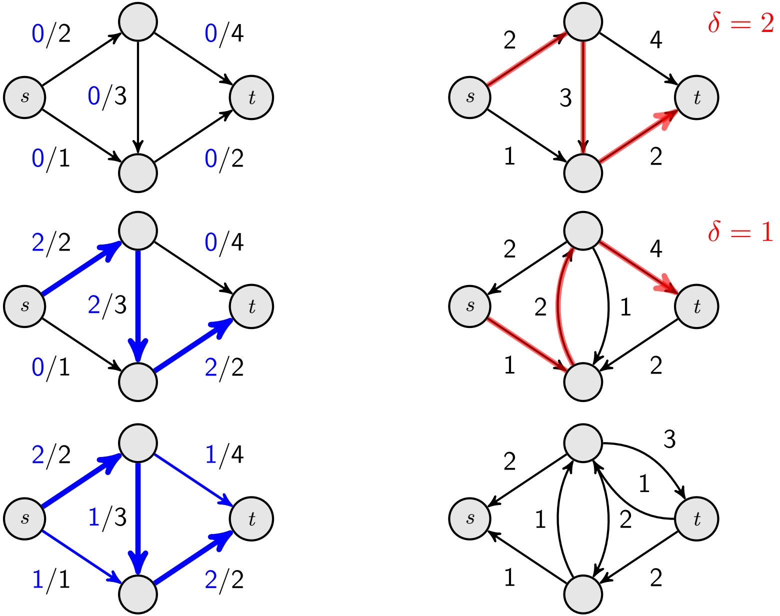
$\delta \leftarrow$bottleneck capacity of augmenting path P;
foreach $e \in P$ **do**
  if $e \in E$ **then** $f_e \leftarrow f_e + \delta$;
  else $f_{\bar{e}} \leftarrow f_{\bar{e}} - \delta$;
Return f;

Output of $\mathrm{Augment}(f, c, P)$ is a flow

Ford–Fulkerson algorithm

$f_e \leftarrow 0$ for each $e \in E$;
$G_f \leftarrow$ residual network of G with respect to flow f;
while $\exists$ *an s–t path P in G_f* **do**
  $f \leftarrow \mathrm{Augment}(f, c, P)$;
  Update G_f;

Example



0/2
0/4
s
0/3
t
0/1
0/2
2
4
$\delta = 2$
s
3
t
1
2
2/2
0/4
s
2/3
t
0/1
2/2
2
4
$\delta = 1$
s
2
1
t
1
2
2/2
1/4
s
1/3
t
1/1
2/2
2
3
1
s
1
2
t
1
2

Termination and Running time

Suppose that $c_e \in \mathbb{Z}_{++}$ $(\forall e \in E)$

Observation

$\forall$ iterations, the flow value f_e and the residual capacity of G_f are integral

Observation

$\forall$ iterations, $\mathrm{val}(f)$ increases at least 1

Theorem

- Ford–Fulkerson algorithm terminates in at most $C := \sum_{e \in E} c_e$ steps
- Ford–Fulkerson algorithm can be implemented to run in $\mathrm{O}(mC)$ time

s–t path can be found in $\mathrm{O}(m)$ time by BFS or DFS

Correctness

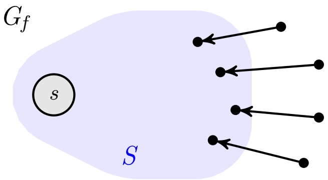
Theorem

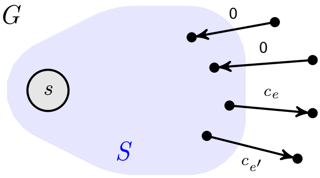
Ford–Fulkerson algorithm outputs a max-flow

Proof

- When it terminates, $\nexists$ s–t path in G_f
- Let S be the set of vertices reachable from s in G_f ($s \in S$ and $t \notin S$)
- $\mathrm{val}(f) = \sum_{e:\ \text{out of } S} f_e - \sum_{e:\ \text{into } S} f_e = \sum_{e:\ \text{out of } S} c_e = \mathrm{cap}(S)$
- By the weak duality, f is a max-flow and $(S, V \setminus S)$ is a min-cut

$\mathrm{val}(f') \le \mathrm{cap}(A)$



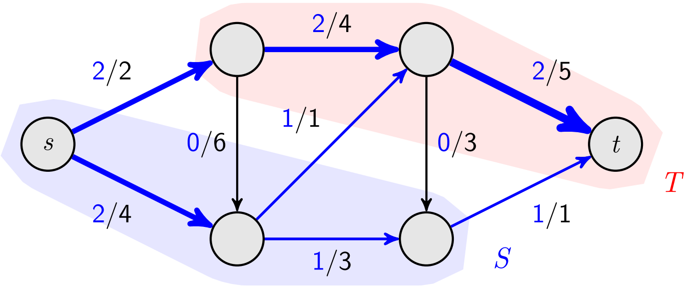


Max-flow Min-cut theorem

Theorem

$\max_{f:\,\mathsf{flow}} \mathrm{val}(f) = \min_{(S,T):\mathsf{cut}} \mathrm{cap}(S)$

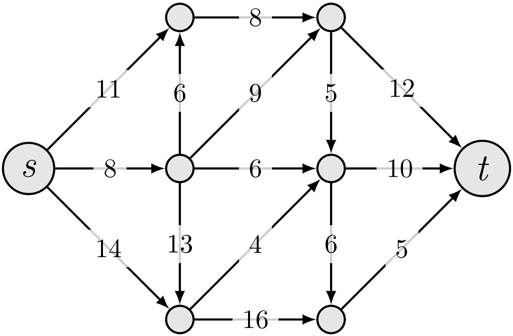
Example



$\mathrm{val}(f) = \mathrm{cap}(S) = 4$

Quiz

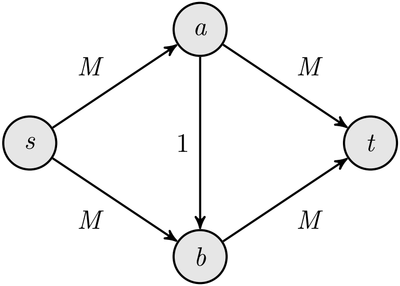
What is the maximum value of s–t flow?



Outline

Bad example

Ford–Fulkerson is too slow (exponential time w.r.t. input size)



- $s \to a \to b \to t$
- $s \to b \to a \to t$
- $s \to a \to b \to t$
- $s \to b \to a \to t$
- $\vdots$

we'd like to choose "good" augmenting path

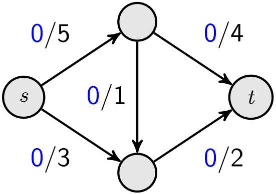
Capacity scaling

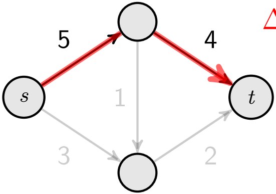
- Choosing augmenting paths with large bottleneck capacity
- $G_f(\Delta)$: subgraph of G_f consisting only of edges e with $c_f(e) \geq \Delta$

Scaling algorithm

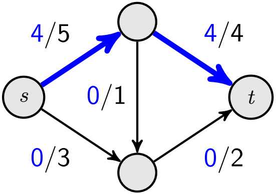
```
Δ ← largest power of 2 that is no larger than max_{e: out of s} c_e;
G_f ← residual network of G with respect to flow f;
while Δ ≥ 1 do
    while ∃ an s–t path P in G_f(Δ) do
        f ← Augment(f, c, P);
        Update G_f(Δ);
    Δ ← Δ/2;
```

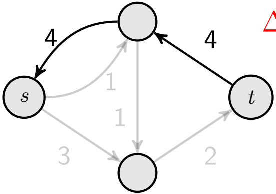
Example



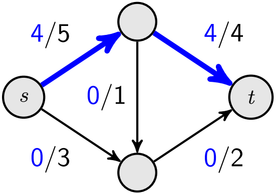


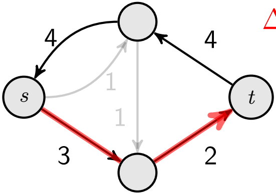
$\Delta = 4,\ \delta = 4$





$\Delta = 4$





$\Delta = 2,\ \delta = 2$

Analyzing the algorithm

Lemma

$\max_{e:\text{ out of } s} c_e$

The number of scaling phases is $1 + \lfloor \log_2 C \rfloor$

$\because \Delta = 2^{\lfloor \log_2 C \rfloor}, 2^{\lfloor \log_2 C \rfloor - 1}, \ldots, 2^0$

Lemma

the set of vertices reachable from s in $G_f(\Delta)$

At the end of a Δ-scaling phase, $\mathrm{cap}(S) \leq \mathrm{val}(f) + m\Delta$

see next slide

Lemma

The number of augmentations in each scaling phase is at most $2m$

- at the beginning of Δ-scaling phase, max-flow$\leq \mathrm{val}(f) + m(2\Delta)$
- each augmentation increases $\mathrm{val}(f)$ by at least Δ

Theorem

weakly polynomial time

The scaling algorithm can be implemented to run in $\mathrm{O}(m^2 \log C)$ time

finding an augmenting path takes $\mathrm{O}(m)$ time

Proof of the lemma

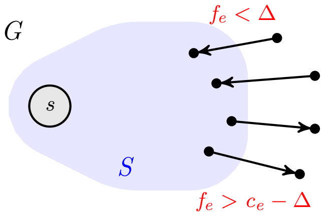
Lemma

the set of vertices reachable from s in $G_f(\Delta)$

At the end of a Δ-scaling phase, $\mathrm{cap}(S) \leq \mathrm{val}(f) + m\Delta$

Proof

$$\begin{aligned}
\mathrm{val}(f) &= \sum_{e:\ \text{out of } S} f_e - \sum_{e:\ \text{into } S} f_e \\
&\geq \sum_{e:\ \text{out of } S} (c_e - \Delta) - \sum_{e:\ \text{into } S} \Delta \\
&\geq \mathrm{cap}(S) - m\Delta
\end{aligned}$$



Summary

- Ford–Fulkerson algorithm $\longrightarrow$ $\mathrm{O}(mC)$ time (pseudo polynomial-time)
- Scaling algorithm $\longrightarrow$ $\mathrm{O}(m^2 \log C)$ time (weak polynomial-time)

Next week

- Edmonds–Karp algorithm $\longrightarrow$ $\mathrm{O}(m^2 n)$ time (strong polynomial-time)