Michael Forbes
mforbes@illinois.edu

2019-10-10.4 →
2019-10-15.2 ←
2019-10-15.1
CS473

CS473 Algorithms: Lecture 14

logistics: - pset 5 due W 10

last time: - network flow - edge - path
- network cuts

today: - flow algo

---

def: A single-source/sink network is a directed graph $G=(V,E)$
w/ source $s \in V$, sink $t \in V$ ($s \neq t$), capacities $c: E \to \mathbb{R}_{\geq 0}$ ← general
$c: E \to \mathbb{Z}_{\geq 0}$ ← this course [we'll see why today]

An (s,t)-flow is $f: E \to \mathbb{R}_{\geq 0}$ [f non integral even if c integral]
- capacity constraints: $\forall e$ $0 \leq f(e) \leq c(e)$
- conservation constraints: $f^{in}(v) = \sum_{u \to v} f(u \to v)$

$f^{out}(v) = \sum_{v \to u} f(v \to u)$

$f(v) = f^{out}(v) - f^{in}(v)$

then $\forall v \in V \setminus \{s,t\}$: $f(v) = 0$

The value of f is $|f| = f(s)$

lem: $= -f(t)$

Q: given network, find max flow

def: network $G=(V,E)$, $s \neq t \in V$, capacities c
An (s,t)-cut is a partition $V = S \sqcup T$ w/ - $s \in S$ - $t \in T$
The capacity of the cut is $|C| = c(S,T) = \sum_{u \in S, v \in T} c(u \to v)$ ← is 0 if no edge

Q: given network find min capacity cut. [min cut]

Prop: f (s,t) flow, C (s,t) cut $\Rightarrow |f| \leq |C|$ [no back edges]
$\Rightarrow$ max flow $\leq$ min cut [bottleneck argument]

thm: $=$ [will prove today]

rmk: - flows and cuts are dual objects [multiple perspectives] [linear programming]
- max flow = min cut is used in proof of correctness of algo

lem: wlog network G has no bidirected edges [simplifies notation]

Sketch: $u \underset{b}{\overset{a}{\rightleftarrows}} v \longmapsto$ $u \xrightarrow{a} \circ \xrightarrow{a} v$, $v \xrightarrow{b} \circ \xrightarrow{b} u$

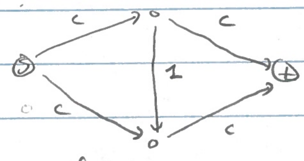
show that: - value preserving bijection of flows, efficiently computable
- " cuts "
$\Rightarrow$ max flow, min cut unchanged [value, algorithmically]

Q: what algorithmic paradigm to use?
divide and conquer? [not very natural here]
dynamic programming? [nothing known]
randomized algo? [sort of helpful in special cases]

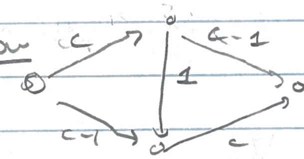
Michael Forbes
mforbes@illinois.edu
2019-10-15.2 $\leftarrow$ 2019-10-15.1
$\rightarrow$ 2019-10-15.3
cs473

greedy?

capacities:



flow



max flow = 2c

flow value 2c-1, is local optimum!
not global

idea: allow flow to reverse in greedy update.

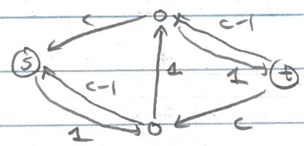
def: network $G=(V,E)$, $s\neq t\in V$, capacities c, flow f in G

residual capacities

The residual network $G_f=(V,E_f)$ with capacities c_f, where E_f, c_f

- forward edges: for $(u,v)\in E$ if $f(u\to v) < c(u\to v)$, add (u,v) to E_f
  $c_f(u,v) = c(u\to v) - f(u,v) > 0$
- backward edges: for $(u,v)\in E$ if $f(u\to v) > 0$ [how much flow to reverse]
  add (v,u) to E_f, where $c_f(v\to u) = f(u\to v)$

ex:



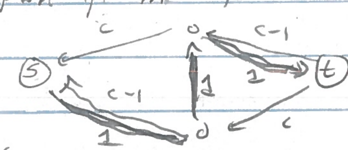
rmk: - wlog G has no bidirected edges $\Rightarrow$ G_f may have bidirected edges, but no parallel edges

- G has m edges $\Rightarrow$ G_f has $\leq 2m$ edges

def: network G, flow f in G, residual graph $G_f=(V,E_f)$

An augmenting path is a (simple) s-t path p in G_f

its capacity is $|p| = \min_{e\in p} c_f(e)$



lem: network G, flow f in G

p augmenting path in G_f $\Rightarrow$ $f+p$ is flow in G

w/ value $|f+p| = |f|+|p|$

pf: path p $s\rightsquigarrow t$ simple, assigns flow $|p|$ to each edge $e\in p$

flow $f+p$ defined by $f(u\to v) = \begin{cases} f(u\to v) & \text{if } (u,v),(v,u)\notin p \\ f(u\to v)+|p| & (u,v)\in p \Rightarrow (v,u)\notin p \text{ [p simple]} \\ f(u\to v)-|p| & (v,u)\in p \end{cases}$

capacity constraints: along $(u,v)\in E$

$(u,v),(v,u)\notin p$: not affected

$(u,v)\in p$ [forward edge]: $|p| \leq c_f(u\to v) = c(u\to v) - f(u\to v)$
$\Rightarrow 0 \leq f(u\to v)+|p| \leq c(u\to v)$

$(v,u)\in p$: [backward edge] $|p| \leq c_f(v\to u) = f(u\to v)$
$\Rightarrow 0 \leq f(u\to v) - |p| \leq f(u\to v) \leq c(u\to v)$

conservation: p is simple s-t path, hence if p visits $v\neq s,t$ p goes in then out exactly once

hence: $v\neq s,t$
$\begin{cases} (f+p)^{in}(v) = f^{in}(v),\ (f+p)^{out}(v) = f^{out}(v) & p \text{ skips } v \\ (f+p)^{in}(v) = f^{in}(v)+|p|,\ (f+p)^{out}(v) = f^{out}(v)+|p| & p \text{ visits } v \end{cases}$

$\Rightarrow$ conserved

value: p is simple s-t path hence

$(f+p)^{in}(s) = f^{in}(s)$

$(f+p)^{out}(s) = f^{out}(s)+|p|$

$\Rightarrow$ value increases by $|p|$ ∎

Michael Forbes
mforbes@illinois.edu
2019-10-15.2 ⇄ 2019-10-15.3
2019-10-15.4
cs473

algo (Ford Fulkerson 1956):

network $G=(V,E)$, $s\neq t\in V$, capacities c

$f(e)\leftarrow 0 \ \forall e\in E$

initialize G_f

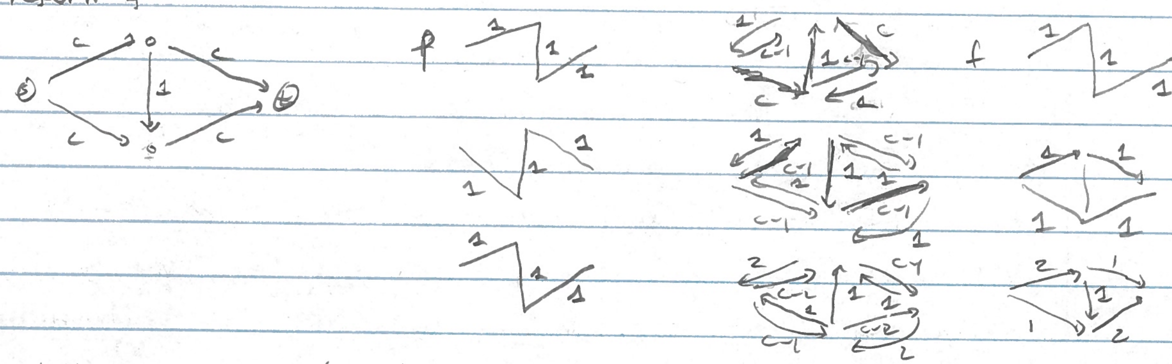
while augmenting simple s-t path p in G_f ← breadth/depth first search to find p, compute $|p| = \min_{e\in p} c_f(e)$ $O(m+n)$

$f \leftarrow f+p$ ← $O(n)$ additions/subtract

$G_f \leftarrow G_{f+p}$ ← $O(n)$ edges to update

return f

ex:



lem: each iteration takes $O(m+n)$ time

Q: termination? [integrality!]

lem: during algo, as $c: E\to\mathbb{Z}_{\geq 0}$
- $f: E\to\mathbb{Z}_{\geq 0}$ [f in general could be real]
- $c_f: E_f\to\mathbb{Z}_{\geq 0}$

Pf: by induction

base case: $f=0$, $c_f = c$ integral

induction: $|p| = \min_{e\in p} c_f(e) \in \mathbb{Z}_{\geq 0}$

$\Rightarrow f+p$ integral $\Rightarrow c_{f+p}$ integral as $c_{f+p}(u\to v) = \begin{cases} c(u\to v) - (f+p)(u\to v) \\ (f+p)(v\to u)\end{cases}$

Cor: algo takes $\leq \sum_{e\in E} c(e) = C$ iterations

Pf: initial flow value $=0$, all flows value $\leq \sum_{e\in E} c(e)$; each iteration increases flow value by $|p|\in\mathbb{Z}_{>0} \Rightarrow |p|\geq 1$] $\Rightarrow \leq C$ iterations

Cor: algo takes $O(m\cdot C)$ time

Q: what does termination "look like"? [no s-t path in G_f]

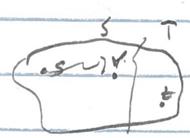
Prop: no $s\leadsto t$ path in G_f $\Rightarrow$ exists cut $V = S\sqcup T$ w/ $S\ni s$, $T\ni t$

$|f| = c(S,T)$

$\Rightarrow$ f is flow of max value [last time flow ≤ cut]

$S\sqcup T$ is cut of min value

Pf:



$S = \{v : s\leadsto v \text{ path}\} \ni s$, $\not\ni t$

$T = V\setminus S$

Michael Forbes
miforbes@illinois.edu
2019-10-15.4 ⇄ 2019-10-15.3 → 2019-10-17.1
CS473

no $s \rightsquigarrow t$ path in G_f ⇒ no edges $S \to T$ in G_f
$u \in S$, $v \in T$ ⇒ $c_f(u \to v) = 0$

$$\Rightarrow \begin{cases} f(u \to v) = c(u \to v) & (u,v) \in E \\ f(u \leftarrow v) = 0 & (v,u) \in E \end{cases}$$

lem. [last time] $|f| = \sum_{u \in S, v \in T} \underbrace{f(u \to v)}_{= c(u \to v)} - \sum_{u \in S, v \in T} \underbrace{f(u \leftarrow v)}_{= 0}$

$= \sum_{u \in S, v \in T} c(u \to v) = c(S,T)$

[correctness]
Cor: Ford Fulkerson terminates w/ max flow [terminates w/ no $s \to t$ path in G_f ⇒ $|f| = c(S,T)$ ⇒ max flow]

Cor: given max flow, can compute min cut in $O(m)$ time [is min cut as $t \in T$ as no augmenting path]
Pf: $S = \{v : s \rightsquigarrow v \text{ in } G_f\}$

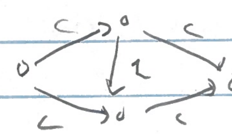
Cor: network G w/ integral capacities – $\max_f |f| = \min_{S,T} c(S,T)$
– some max flow is integral

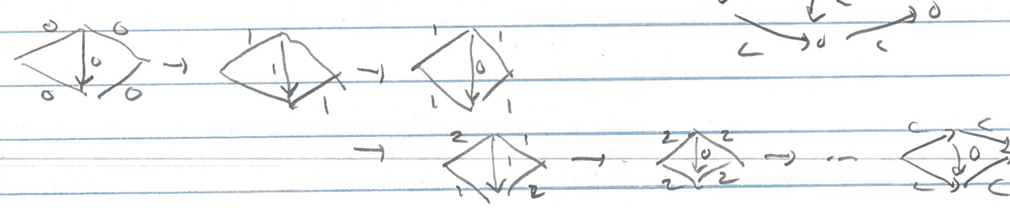
rmk – algorithmic proof, for integer capacities ← clearing denominators
can be extended to rational capacities in straightforward way

– irrational capacities: – Ford Fulkerson may not terminate, nor converge to max flow
– max flow = min cut still holds
prove via – better math
– better algo

– Ford Fulkerson actually runs in $O(m|f^*|)$ time ← max flow value
– ↳ $O(mC)$ time
↳ $\sum_e c(e)$ ← give $c(e)$ in binary representation
but input size is $O(m \lg C)$
hence: FF not "poly time" algo if C is large!

– FF can require $\Omega(C)$ iterations, eg





Q: polynomial time algo?

logistics: pset 5 due W10
today: Ford Fulkerson
next time: efficient flow algo
applications of max flow