

Honors Algorithms
G22.3520-001 Fall 2007

Lecture 19

Network Flow

Flow Network:

- A directed graph $G = (V, E)$ (no self loops)
- $s, t \in V$, with $s \neq t$
- $s = \text{"source"}$, $t = \text{"sink"}$
- $c : V \times V \rightarrow \mathbb{R}_{\geq 0}$
- $c(u, v) = 0$ if $(u, v) \notin E$
- $c(u, v) = \text{"capacity"}$ of edge (u, v)

A *flow* for G is a function $f : V \times V \rightarrow \mathbb{R}_{\geq 0}$ such that:

1. **(capacity constraints)**

$$f(u, v) \leq c(u, v) \text{ for all } u, v \in V$$

2. **(conservation of flow)**

For all $u \in V \setminus \{s, t\}$:

$$\sum_{v \in V} f(v, u) = \sum_{v \in V} f(u, v)$$

flow into u = flow out of u

(like Kirchhoff's Law)

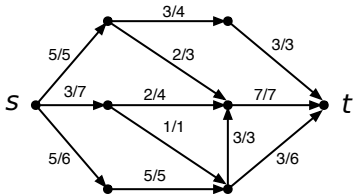
Summation notation:

- for $X, Y \subseteq V$: $f(X, Y) := \sum_{x \in X} \sum_{y \in Y} f(x, y)$
- Conservation of flow: $f(V, u) = f(u, V)$ for all $u \notin \{s, t\}$

The *value* of flow f : $|f| := f(s, V) - f(V, s) =$ “net flow out of s ”

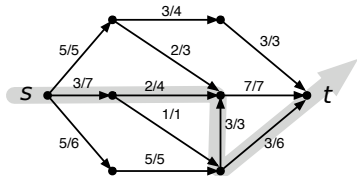
Problem: given a flow network (G, s, t, c) , find a flow of *maximum* value

Example:

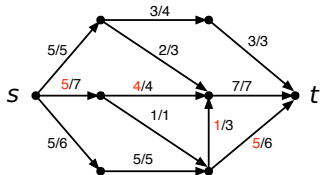


$$|f| = 13$$

However, f is not a maximum flow



push 2 units of flow along an *augmenting path*



$|f_{\text{new}}| = 15$
maximum flow

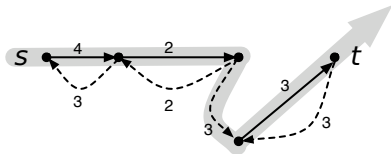
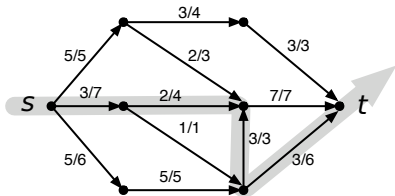
The residual graph

- Let f be a flow
- For $u, v \in V$, define
$$c'(u, v) := c(u, v) - f(u, v) + f(v, u)$$
- $c'(u, v)$ is the *residual capacity* from u to v
- we can increase the *net flow* from u to v (i.e., $f(u, v) - f(v, u)$) by $c'(u, v)$:
 - increase flow on the edge (u, v) by $c(u, v) - f(u, v)$
 - decrease flow on the edge (v, u) by $f(v, u)$

The residual graph (cont'd)

- Define the *residual graph* $G' = (V, E')$, where $E' := \{(u, v) : c'(u, v) > 0\}$
- An *augmenting path* is a simple path from s to t in G'
- If $p = \langle v_0, \dots, v_k \rangle$ is an augmenting path, the *residual capacity* of p ,
 $\Delta := \min\{c'(v_{i-1}, v_i) : i = 1 \dots k\}$
- If there is an augmenting path p with residual capacity Δ , we can increase the flow by Δ by *saturating* p : for each edge (u, v) in p , increase net flow from u to v by Δ

Example:



Ford-Fulkerson Algorithm:

$f \leftarrow 0$

while there is an augmenting path p do
 saturate p

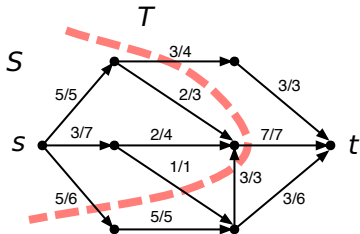
Questions:

- How to find augmenting path?
 - Build residual graph G' , and do DFS or BFS from s
- What is the time per loop iteration? $O(|V| + |E|)$
- Will the algorithm terminate? ???
- When the algorithm terminates, is the flow maximum? ???

Definitions:

- A *cut* is a pair (S, T) , where $s \in S, t \in T$,
 $S \cup T = V, S \cap T = \emptyset$
- The *net flow* across (S, T) is $f(S, T) - f(T, S)$
- The *capacity* of (S, T) is $c(S, T)$
- Clearly, net flow \leq capacity

Example:



$$\text{Net flow} = 5 + 1 - 3 + 7 + 3 = 13$$

$$\text{Capacity} = 6 + 1 + 7 + 4 = 18$$

Lemma 1: for all $X \subseteq V \setminus \{s, t\}$: $f(V, X) = f(X, V)$

Proof:

- $f(V, X) = \sum_{x \in X} f(V, x) = \sum_{x \in X} f(x, V) = f(X, V)$

Lemma 2:

- if $X, Y, Z \subseteq V$, with $X \cap Y = \emptyset$, then
 - $f(X \cup Y, Z) = f(X, Z) + f(Y, Z)$, and
 - $f(Z, X \cup Y) = f(Z, X) + f(Z, Y)$

Lemma 3:

- If (S, T) is a cut, and f is a flow, then $|f| = f(S, T) - f(T, S)$

Proof:

$$\begin{aligned} f(s, V) &= f(S, V) - f(S \setminus s, V) \quad (L2) \\ &= f(S, V) - f(V, S \setminus s) \quad (L1) \\ &= f(S, V) - (f(V, S) - f(V, s)) \quad (L2) \\ &= f(S, S) + f(S, T) - \\ &\quad (f(S, S) + f(T, S) - f(V, s)) \quad (L2) \\ &= f(S, T) - f(T, S) + f(V, s) \end{aligned}$$

Corollary: $|f| \leq c(S, T)$

Corollary: $|f| = f(V, t) - f(t, V) =$ “net flow into t ”

Augmenting Path Theorem:

- A flow f is maximum \Leftrightarrow it admits no augmenting path

Proof:

- We already observed " \Rightarrow "
- To prove " \Leftarrow ", assume f admits no augmenting path
- Define
$$S := \{u \in V : \text{there is a path from } s \text{ to } u \text{ in the residual graph}\}$$
- Set $T := V \setminus S$, so (S, T) is a cut

Proof (cont'd):

- Claim:
 - for all $u \in S$, $v \in T$, we have $f(u, v) = c(u, v)$ and $f(v, u) = 0$
- Proof of claim:
 - Suppose $f(u, v) < c(u, v)$ or $f(v, u) > 0$
 - Then $c'(u, v) = c(u, v) - f(u, v) + f(v, u) > 0$
 - (u, v) is an edge in the residual graph
 - since $u \in S$, there is a path to u in the residual graph
 - \therefore there is a path to v in the residual graph
 - $\Rightarrow \Leftarrow$

Proof (cont'd):

- $|f| = \text{net flow across } (S, T) = \text{capacity of } (S, T)$
- Net flow across (S, T) can never exceed capacity of (S, T)
- $|f|$ cannot be increased
- QED

Corollary: Max-Flow Min-Cut Theorem

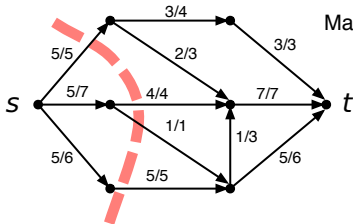
- If f is a maximum flow, then

$$|f| = m := \min\{c(S, T) : (S, T) \text{ is a cut}\}$$

Proof:

- We already know that $|f| \leq c(S, T)$ for all cuts (S, T)
- $\therefore |f| \leq m$
- In proof of Augmenting Path Theorem, we saw that $|f| = c(S, T)$ for a particular cut (S, T)
- $\therefore |f| \geq m$
- QED

Example:



Max Flow = 15 = Min Cut

Corollary: Integrality Theorem

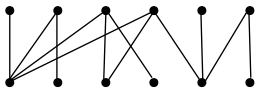
- If all capacities are integers, then there is a maximum flow f with $f(u, v) \in \mathbb{Z}$ for all $u, v \in V$.
- Moreover, the Ford-Fulkerson algorithm will produce such a flow after at most $|f|$ iterations

Proof:

- In each iteration of the algorithm, the flow is integral
- Flow must increase by at least 1 in every iteration
- QED

Application: Maximum Bipartite Matching

A bipartite graph



A maximal matching

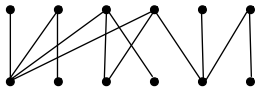


A *maximum* matching

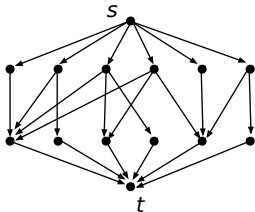


Reduction to Max Flow:

A bipartite graph

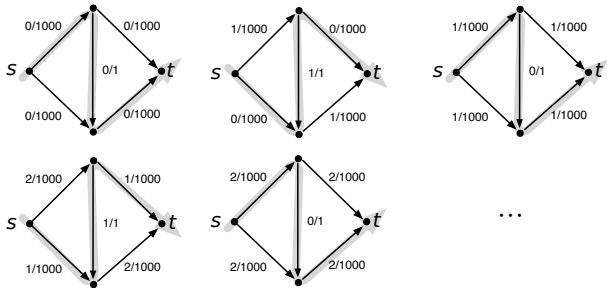


A flow network
(unit capacity on each edge)



- Matchings correspond to integral flows
- Max Flow = $O(|V|)$
- Ford-Fulkerson terminates after $O(|V|)$ iterations
- Total running time $O(|V|(|V| + |E|))$

Potentially bad behavior of Ford-Fulkerson:



Edmonds-Karp Heuristic:

- Always choose an augmenting path with the least number of edges (i.e., use BFS)

Theorem:

- Using the Edmonds-Karp heuristic, the Ford-Fulkerson algorithm terminates in $O(|E||V|)$ iterations

Proof: Next Time