CS 341: Algorithms

Lecture 17: Max flow = Min cut

Éric Schost

based on lecture notes by many other CS341 instructors

David R. Cheriton School of Computer Science, University of Waterloo

Fall 2024

Cuts

Cuts

Definition

- a **cut** is a partition of the vertices into sets $\boldsymbol{A}$ and $\boldsymbol{B} = \boldsymbol{V} - \boldsymbol{A}$, with $s \in A$ and $t \in B$.
- the **capacity** of the cut is

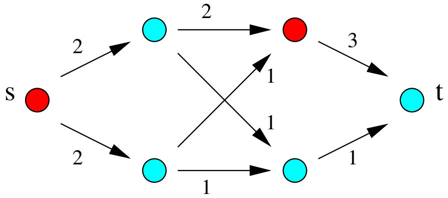
$$c(A) = \sum_{e:A \to B} c(e)$$

(does not depend on any flow, only on the graph and its capacities)

- if f is a flow, the **out-going** and **in-going** flows of the cut are

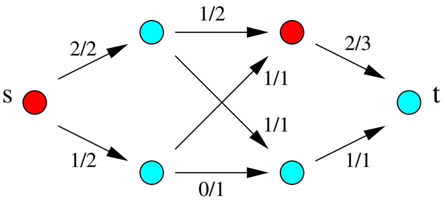
$$v_{\text{out}}(f, A) = \sum_{e:A \to B} f(e), \quad v_{\text{in}}(f, A) = \sum_{e:B \to A} f(e)$$

Examples



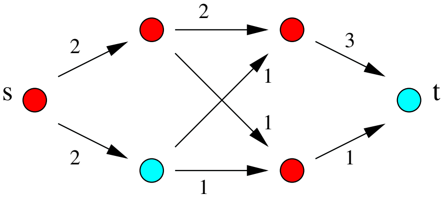
- A is in red and B in light blue,
- capacity is $2 + 2 + 3 = 7$,

Examples



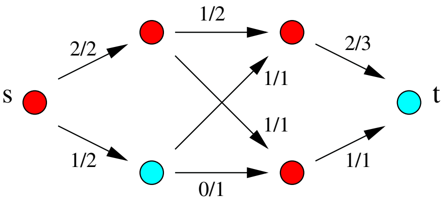
- A is in red and B in light blue,
- capacity is $2 + 2 + 3 = 7$,
- out-going flow is $2 + 1 + 2 = 5$,
- in-going flow is $1 + 1 = 2$,
- value is 3

Examples



- A is in red and B in light blue,
- capacity is $2 + 3 + 1 = 6$,

Examples



- A is in red and B in light blue,
- capacity is $2 + 3 + 1 = 6$,
- out-going flow is $1 + 2 + 1 = 4$,
- in-going flow is 1,
- value is 3

Flows and cuts

Claim

For any flow f and any cut A, we have

$$\mathsf{Val}(f) = v_{\text{out}}(f, A) - v_{\text{in}}(f, A)$$

Remark: this shows that what comes out of s equals what comes into t.

Flows and cuts

Claim

For any flow f and any cut A, we have

$$\mathsf{Val}(f) = v_{\text{out}}(f, A) - v_{\text{in}}(f, A)$$

Remark: this shows that what comes out of s equals what comes into t.

Proof: induction on A.

- true when $A = \{s\}$, by definition.
- suppose this is true for a cut A, $B = V - A$, we show this is true for the cut $A' = A \cup \{v\}$, $B' = B - \{v\}$, for any vertex $v \in B$ (with $v \neq t$).

What we need to do:

- relate $v_{\text{out}}(f, A)$ to $v_{\text{out}}(f, A')$,
- relate $v_{\text{in}}(f, A)$ to $v_{\text{in}}(f, A')$.

Step 1

$$\begin{aligned} v_{\text{out}}(f, A) &= \sum_{e:A\to B} f(e) \\ &= \sum_{e:A\to v} f(e) + \sum_{e:A\to B'} f(e) \end{aligned}$$

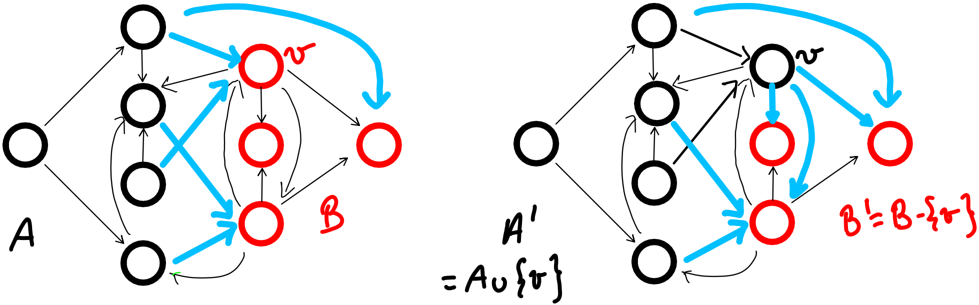
and

$$\begin{aligned} v_{\text{out}}(f, A') &= \sum_{e:A'\to B'} f(e) \\ &= \sum_{e:A\to B'} f(e) + \sum_{e:v\to B'} f(e). \end{aligned}$$

so

$$v_{\text{out}}(f, A') = v_{\text{out}}(f, A) - \sum_{e:A\to v} f(e) + \sum_{e:v\to B'} f(e)$$

Step 1



so

$$v_{\text{out}}(f, A') \quad = \quad v_{\text{out}}(f, A) - \sum_{e:A\to v} f(e) + \sum_{e:v\to B'} f(e)$$

Step 2

$$\begin{aligned} v_{\text{in}}(f, A) &= \sum_{e:B\to A} f(e) \\ &= \sum_{e:v\to A} f(e) \;+\; \sum_{e:B'\to A} f(e) \end{aligned}$$

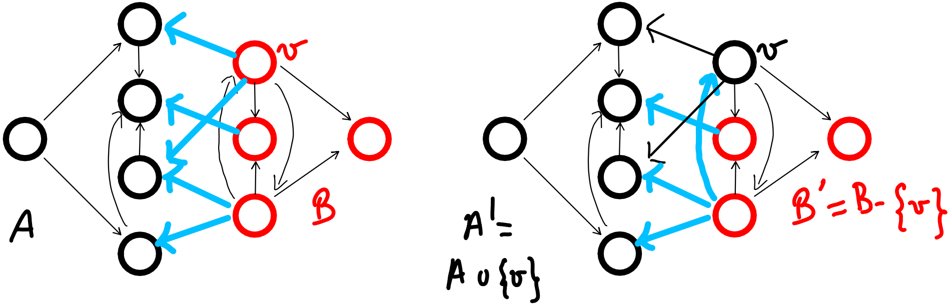
and

$$\begin{aligned} v_{\text{in}}(f, A') &= \sum_{e:B'\to A'} f(e) \\ &= \sum_{e:B'\to A} f(e) \;+\; \sum_{e:B'\to v} f(e). \end{aligned}$$

so

$$v_{\text{in}}(f, A') = v_{\text{in}}(f, A) - \sum_{e:v\to A} f(e) + \sum_{e:B'\to v} f(e)$$

Step 2



so

$$v_{\text{in}}(f, A') = v_{\text{in}}(f, A) - \sum_{e:v \to A} f(e) + \sum_{e:B' \to v} f(e)$$

Step 3

Because f is a flow, we have

$$\sum_{e:v\to A} f(e) + \sum_{e:v\to B'} f(e) = \sum_{e:B'\to v} f(e) + \sum_{e:A\to v} f(e)$$

so

$$\begin{aligned} v_{\text{out}}(f,A') &= v_{\text{out}}(f,A) - \sum_{e:A\to v} f(e) + \sum_{e:v\to B'} f(e) \\ &= v_{\text{out}}(f,A) - \sum_{e:v\to A} f(e) + \sum_{e:B'\to v} f(e) \end{aligned}$$

and still

$$v_{\text{in}}(f,A') = v_{\text{in}}(f,A) - \sum_{e:v\to A} f(e) + \sum_{e:B'\to v} f(e)$$

This gives

$$\begin{aligned} v_{\text{out}}(f,A') - v_{\text{in}}(f,A') &= v_{\text{out}}(f,A) - v_{\text{in}}(f,A) \\ &= \mathsf{Val}(f). \end{aligned}$$

Maximum flow and minimal cut

Consequences

- for **any flow** f and **any cut** A, we have

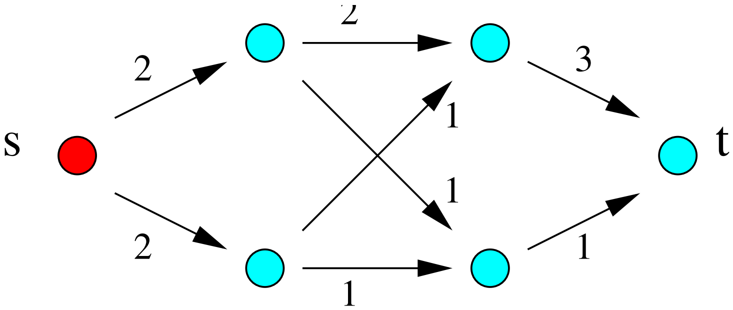
$$\mathsf{Val}(f) \leq c(A).$$

proof:

$$\begin{aligned} \mathsf{Val}(f) &= v_{\text{out}}(f, A) - v_{\text{in}}(f, A) \\ &\leq v_{\text{out}}(f, A) \\ &\leq c(A) \end{aligned}$$

- so the **maximal value** of a flow $\leq$ **minimal capacity** of a cut
- and if we find **any** flow and cut with equality, they are optimal

Example 1

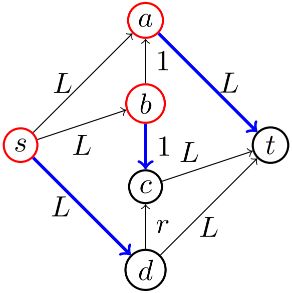


Max flow?

- we found 4 in the previous lecture
- with $A = \{s\}$, $c(A) = 4$
- so max flow = min cut = 4

Example 2

last lecture: $r = (\sqrt{5} - 1)/2 \simeq 0.618$, L large enough



Max flow?

- easy to get $2L + 1$
- with $A = \{s, a, b\}$, $c(A) = 2L + 1$
- so max flow = min cut = $2L + 1$

Max flow = min cut

Claim

no improving path in G_f $\implies$ can find a cut A such that $\mathsf{Val}(f) = c(A)$ $\implies$ f is a max flow

(first $\implies$ to do, second $\implies$ already done)

Consequences:

- **maximal value** of a flow = **minimal capacity** of a cut
- if Ford and Fulkerson's algorithm terminates, we have a max flow and also a min cut.

(we know that for integer capacities, Ford-Fulkerson's algorithm always terminates)

Max flow = min cut

Claim

no improving path in G_f $\iff$ can find a cut A such that $\mathsf{Val}(f) = c(A)$ $\iff$ f is a max flow

(first $\implies$ to do, second $\implies$ already done)

Consequences:

- **maximal value** of a flow = **minimal capacity** of a cut
- if Ford and Fulkerson's algorithm terminates, we have a max flow and also a min cut.

(we know that for integer capacities, Ford-Fulkerson's algorithm always terminates)

Proof

How to build A

- take a flow f with no augmenting path in G_f
- let A be of vertices **reachable from s** in G_f

This is a cut:

- s is in A,
- no path $s \to t$ in G_f so t is in $B = V - A$

Left to prove: $\mathsf{Val}(f) = c(A)$

Computing the value of f

Observation: there is **no edge** $A \to B$ in G_f

out-going flow:

- all **outgoing edges** $(A \to B)$ are saturated in G: $\boldsymbol{f(e) = c(e)}$
- gives $\boldsymbol{v_{\text{out}}(f, A) = c(A)}$

in-going flow

- all **incoming edges** $(B \to A)$ have no flow in G: $\boldsymbol{f(e) = 0}$
- gives $\boldsymbol{v_{\text{in}}(f, A) = 0}$

finally: $c(A) = v_{\text{out}}(f, A) - v_{\text{in}}(f, A) = \mathsf{Val}(f)$

Remark 1: Edmonds-Karp (bonus)

A strategy that refines Ford-Fulkerson: choose a **shortest** path (BFS)

Key ideas

- f : old flow, f' : new flow
- distances from s in the **residual graphs** cannot decrease: for $e = (u, v)$ in $G_{f'}$,
  - if e **was not in** G_f, $\delta_f(s, v) \leq \delta_{f'}(s, v) + 2$
  - else, $\delta_f(s, v) \leq \delta_{f'}(s, v)$

  (takes some work)
- $\boldsymbol{\delta_f(s, v) \leq n}$ so e can **appear** in the residual graph at most $n/2$ times
- but then e also can **disappear** at most $n/2$ times
- each iteration, at least one edge disappears from G_f
- at most $2m$ edges so at most $\boldsymbol{mn}$ iterations
- runtime $\boldsymbol{O(m^2 n)}$

Remark 2: thick paths (bonus)

A slightly weaker strategy to refine Ford-Fulkerson: choose a path that **maximizes the bottleneck** capacity x.

Key ideas

- finding the thickest path: similar to Dijkstra
  - Dijkstra minimizes $\sum_{e\in\gamma} w(e)$
  - here we maximize $\min_{e\in\gamma} c(e)$
- in G_f, there is a path with $x \geq (M - \mathsf{Val}(f))/2m$, $M =$ max flow so

$$\mathsf{Val}(f') \geq \mathsf{Val}(f) + (M - \mathsf{Val}(f))/2m$$

  (takes some work)
- if capacities are integers, implies we do $O(m \log(M))$ iterations
- total $\boldsymbol{O(m^2 \log(n) \log(M))}$

Remark 3: maximal flow from linear programming (bonus)

Equations for the max flow problem:

1. create a variable $f_{u,v}$ for each edge (u, v) and the linear constraints

$$f_{u,v} \geq 0, \quad f_{u,v} \leq c(u, v), \quad \sum_{(u,v) \text{ edge}} f_{u,v} = \sum_{(v,w) \text{ edge}} f_{v,w}$$

2. maximize

$$\sum_{(s,v) \text{ edge}} f_{s,v}.$$

- this is an instance of a **linear programming** problem
- max flow / min cut special case of **linear programming duality** (max something = min something else)

  (takes work)