



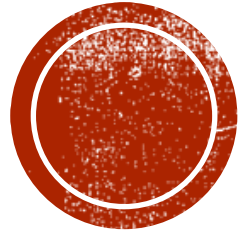
INTRODUCTION TO COMPUTATIONAL TOPOLOGY

HSIEN-CHIH CHANG
LECTURE 9, OCTOBER 12, 2021

ADMINISTRIVIA

- Homework 3 is out, due 10/25 (Mon)
- Optional Final Project:
 - Project proposal is due 10/18 (Mon)
 - Presentation during finals week (likely to be 11/23 (Tue))
 - Project report due 11/29 (Mon)





MINIMUM CUT IN PLANAR GRAPHS



MINIMUM CUT IN A GRAPH

- Given undirected graph G with positive edge-weights and two vertices s and t , find a minimum-weight edge cut separating s and t

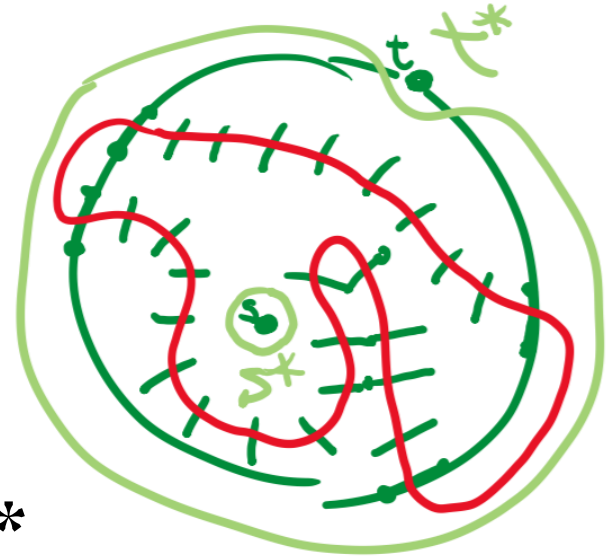


MINIMUM CUT IN PLANAR GRAPH

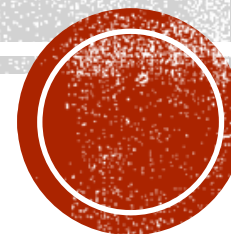
- Given undirected **planar** graph G with positive edge-weights and two vertices s and t , find a minimum-weight edge cut separating s and t

$$\{\text{edge cuts}\} \iff \{\text{circuit} = \text{union of cycles}\}$$

$$\min (s,t)\text{-cut} \iff \text{minimum cycle separating } s^* \text{ and } t^*$$

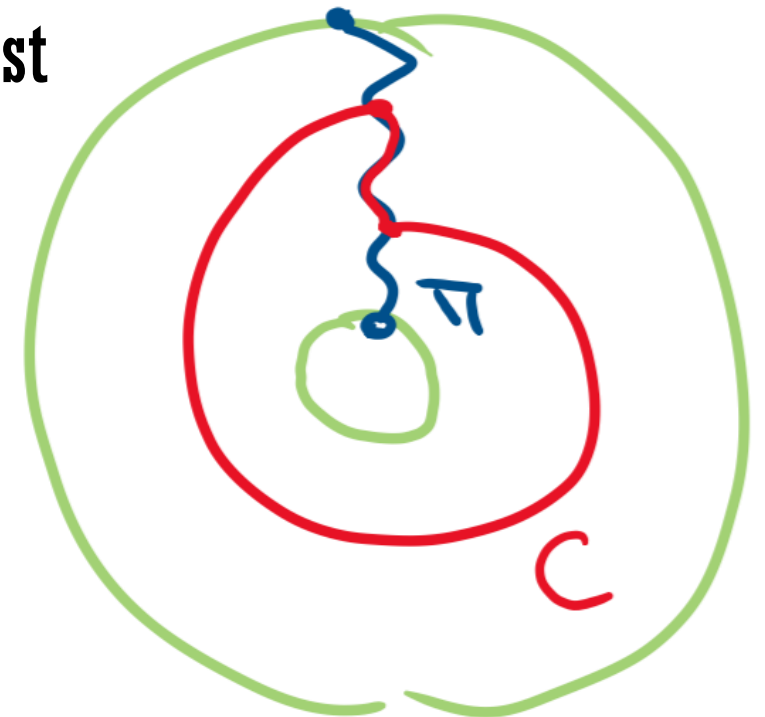


FIND A MIN HOMOTOPIC CYCLE!



OBSERVATIONS

- Shortest cycle C must pass through any path π from s^* to t^*
- Cycle C intersects π at one segment if π is shortest

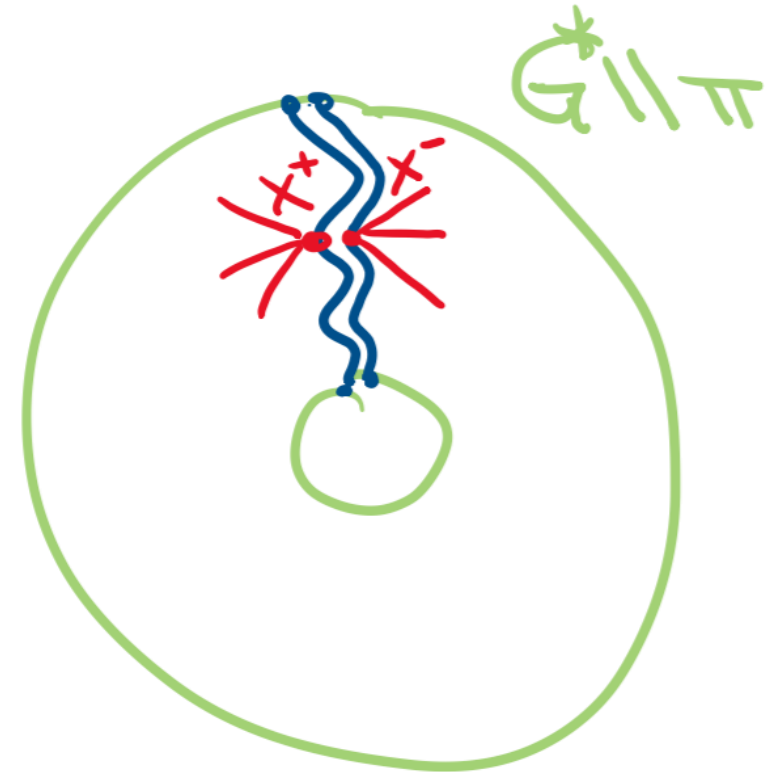


NAÏVE ALGORITHM

MinCut (G, s, t) :

Find shortest path π from $s^* \rightsquigarrow t^*$
Cut open G^* along π .
for each vertex x on π :
 find shortest path $x^+ \rightsquigarrow x^-$

Return length of $\min \{x^+ \rightsquigarrow x^-\}$



REIF'S ALGORITHM

[Reif 1983]

MinCut (G, s, t):

Find shortest path π from $s^* \rightsquigarrow t^*$
Cut open G^* along π .

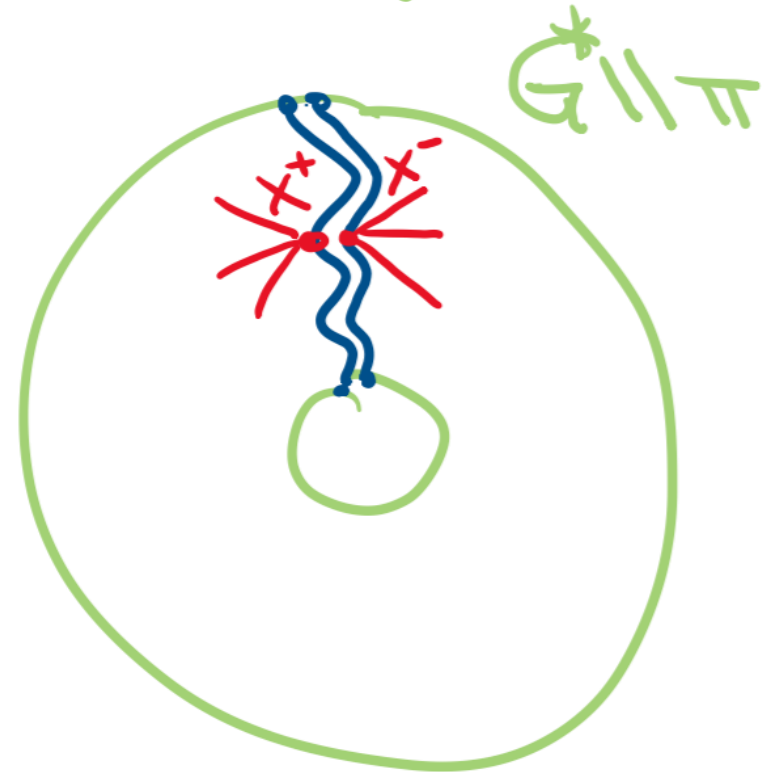
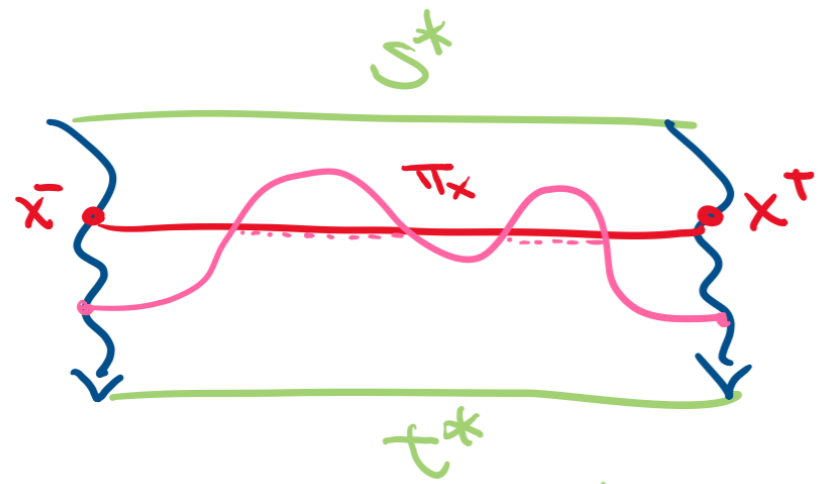
for ~~each~~ ^{middle} vertex x on π :

find shortest path $\pi_x: x^+ \rightsquigarrow x^-$

$\text{MinCut}(G^* \setminus \pi_x, s^*, \pi_x^+)$

$\text{MinCut}(G^* \setminus \pi_x, \pi_x^-, t^*)$

Return length of min $\{x^+ \rightsquigarrow x^-\}$



Improved Algorithms for Min Cut and Max Flow in Undirected Planar Graphs

Giuseppe F. Italiano^{*}
Dipartimento di Informatica,
Sistemi e Produzione
University Rome "Tor Vergata"
Rome, Italy
italiano@disp.uniroma2.it

Yahav Nussbaum[†]
The Blavatnik School of
Computer Science
Tel Aviv University
Tel Aviv, Israel
yahav.nussbaum@cs.tau.ac.il

Piotr Sankowski[‡]
Institute of Informatics
University of Warsaw
Warsaw, Poland
sank@mimuw.edu.pl

Christian Wulff-Nilsen[§]
School of Computer Science
Carleton University
Ottawa, Canada
koolooz@diku.dk

ABSTRACT

We study the min st -cut and max st -flow problems in planar graphs, both in static and in dynamic settings. First, we present an algorithm that given an undirected planar graph and two vertices s and t computes a min st -cut in $O(n \log \log n)$ time. Second, we show how to achieve the same bound for the problem of computing a max st -flow.

Categories and Subject Descriptors

G.2.2 [Graph Theory]: Graph algorithms

General Terms

Algorithms, Theory

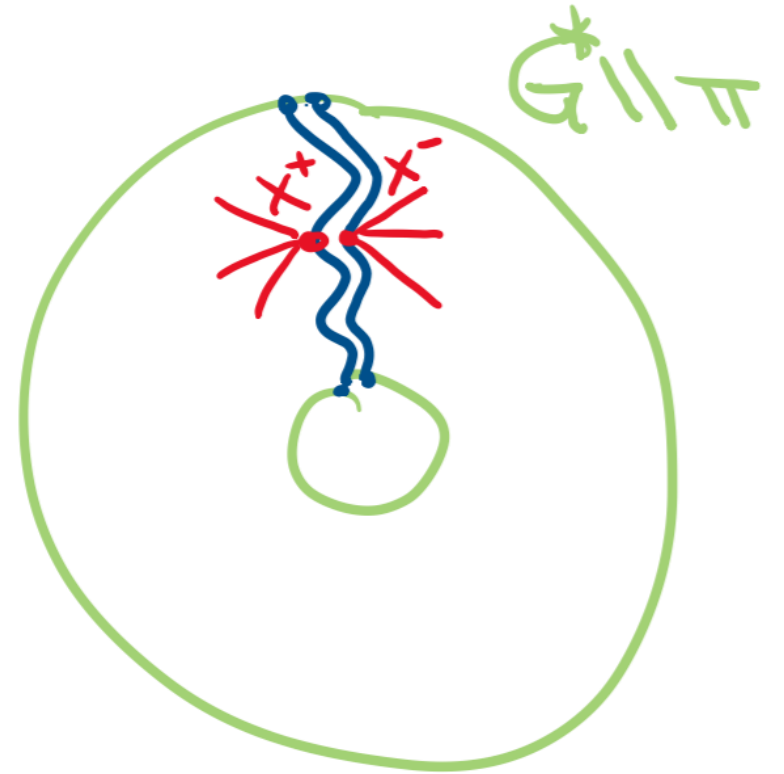
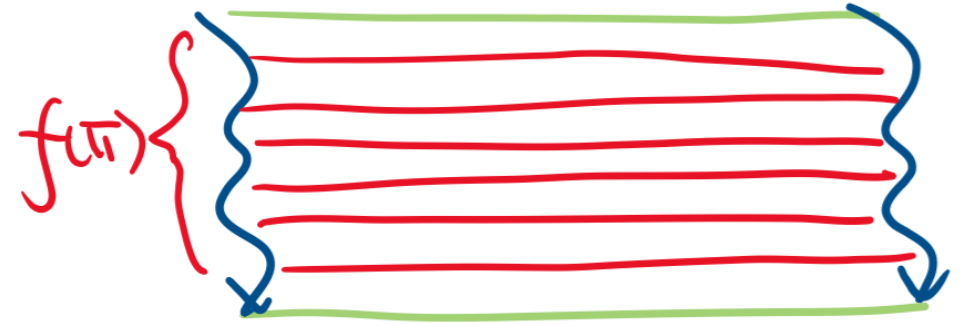
FASTER PLANAR MIN-CUT

[Italiano-Nussbaum-Sankowski-Wulff-Nilsen 2011]

Planar min-cut can be computed in $O(n \log \log n)$ time



HIGH-LEVEL IDEAS

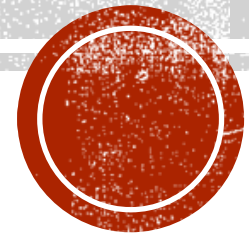


TOOLBOX TO BE BUILT

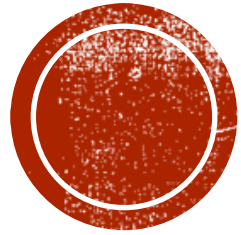
- **Multiple-source shortest paths** [Klein 2005] [Cabello-Chambers-Erickson 2013]
- **Cycle separator decomposition/r-division** [Frederickson 1989] [Klein-Mozes-Sommer 2012]
- **Monge heap/dense distance graph** [Aggarwal-Klawe-Moran-Shor-Wilber 1987]
- **FR-Dijkstra** [Fakcharoenphol-Rao 2001]
- **Monge emulator** [Chang-Ophelders 2020] [Chang-Krauthgamer-Tan 2022]



INTERMISSION



PHILOSOPHICAL QUESTION:
Why do we care about shaving logs?



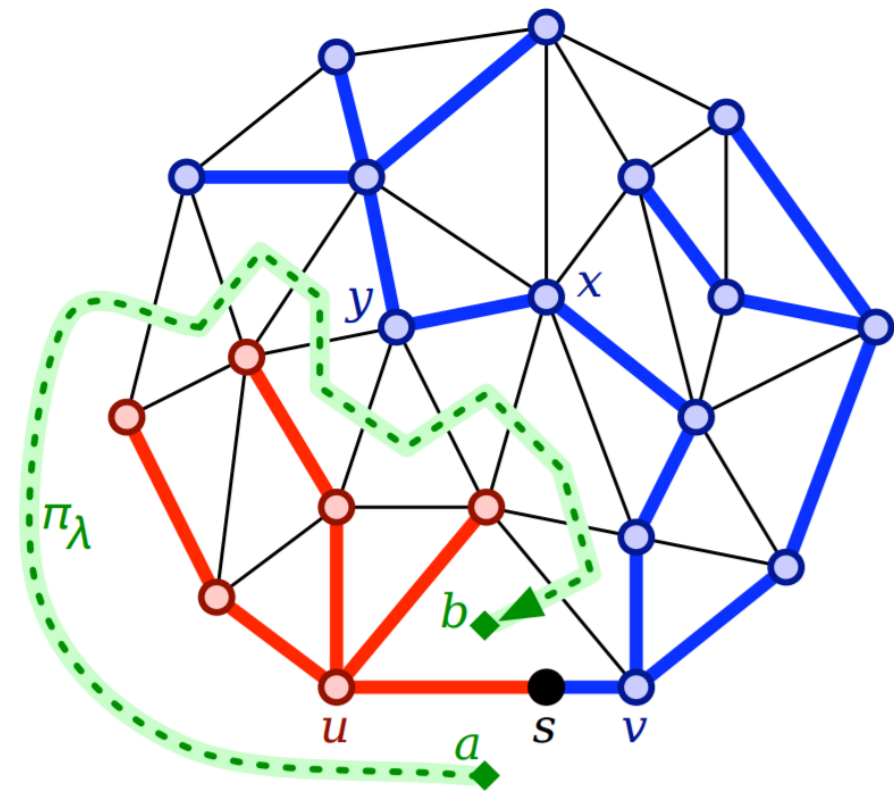
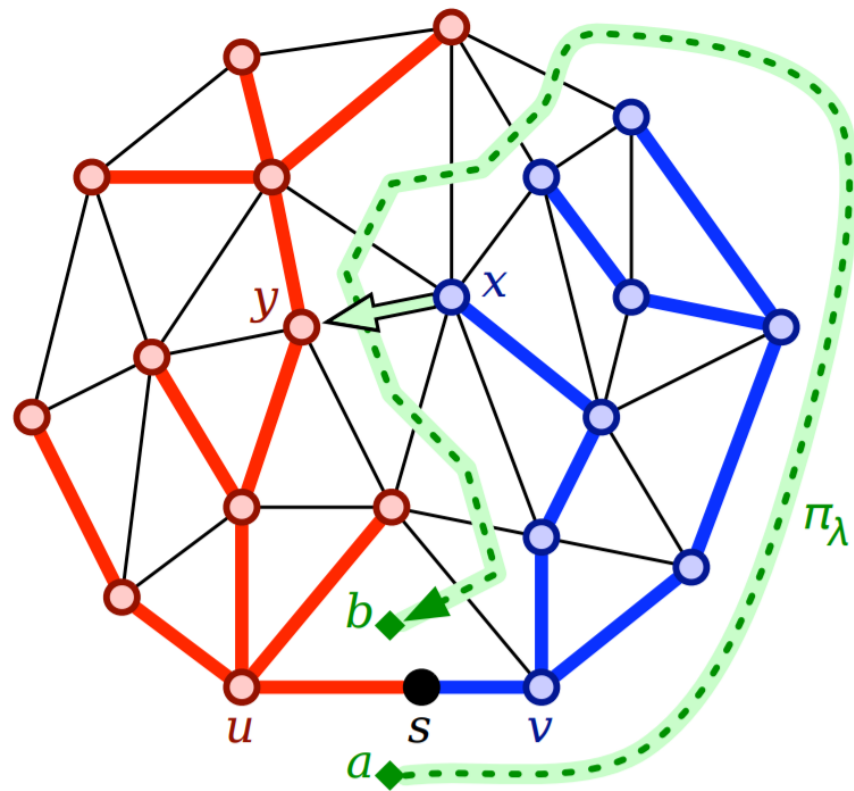
MULTIPLE-SOURCE SHORTEST PATHS



MSSP PROBLEM DEFINITION

- Given a planar directed graph G with and **sources** all on the outer-face, and **edges weights** $w: E(G) \rightarrow \mathbb{R}_+$
- Compute shortest paths between every source s and every vertex x
 - Represented implicitly



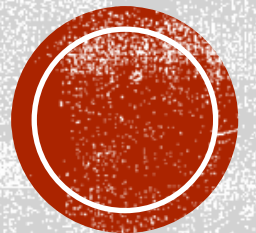


MULTIPLE-SOURCE SHORTEST PATHS

[Klein 2005]

[Cabello-Chambers-Erickson 2013]

MSSP problem can be solved in $O(n \log n)$ time,
such that each distance can be queried in $O(\log n)$ time



DISK-TREE LEMMA. For any spanning tree T and tree-edge e , the boundary vertices in components of $T-e$ are consecutive.

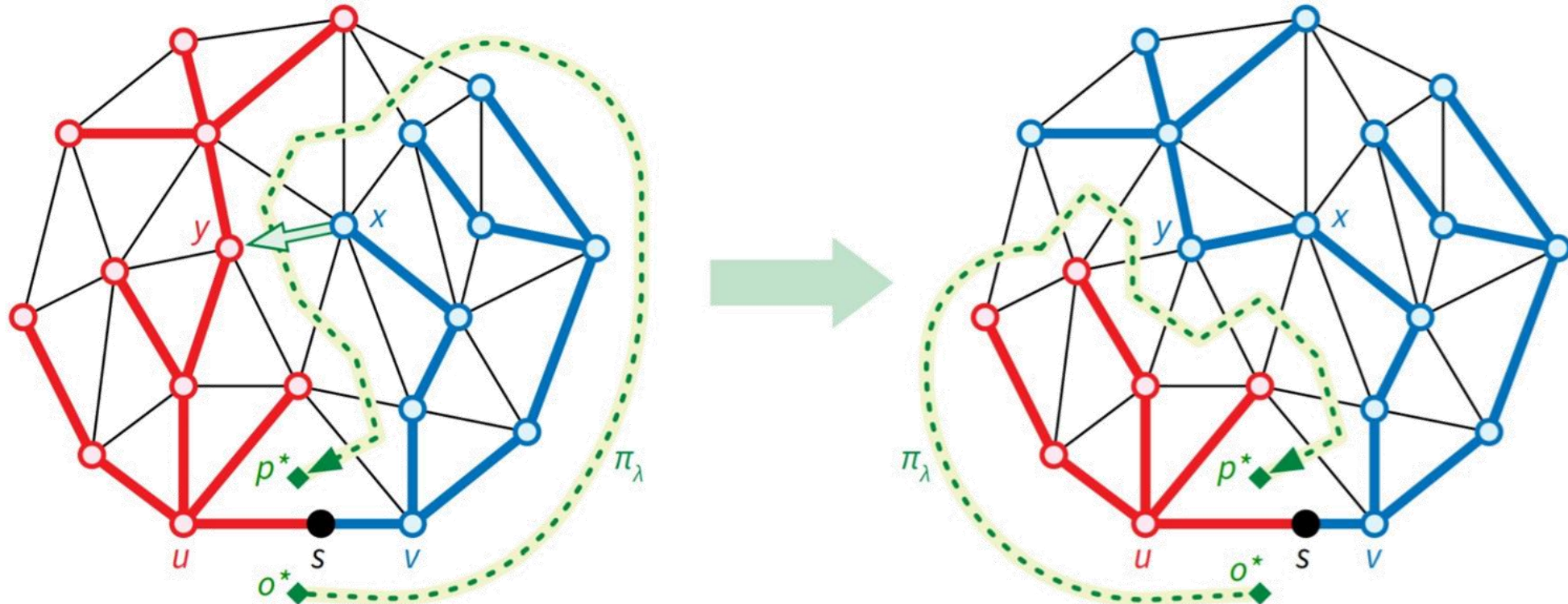


COROLLARY. Let T_0, \dots, T_{k-1} be shortest path trees in sequence. Then any edge $x \rightarrow y$ belongs to a consecutive interval of shortest-path trees: $T_i, \dots, T_{i+j \bmod k}$.



PARAMETRIC SHORTEST PATHS

- Shortest path tree **pivots** as one moves the source



- $d_\lambda(x)$: distance from s to x under w_λ
- $\text{slack}_\lambda(x \rightarrow y) = d_\lambda(x) + w(x \rightarrow y) - d_\lambda(y)$

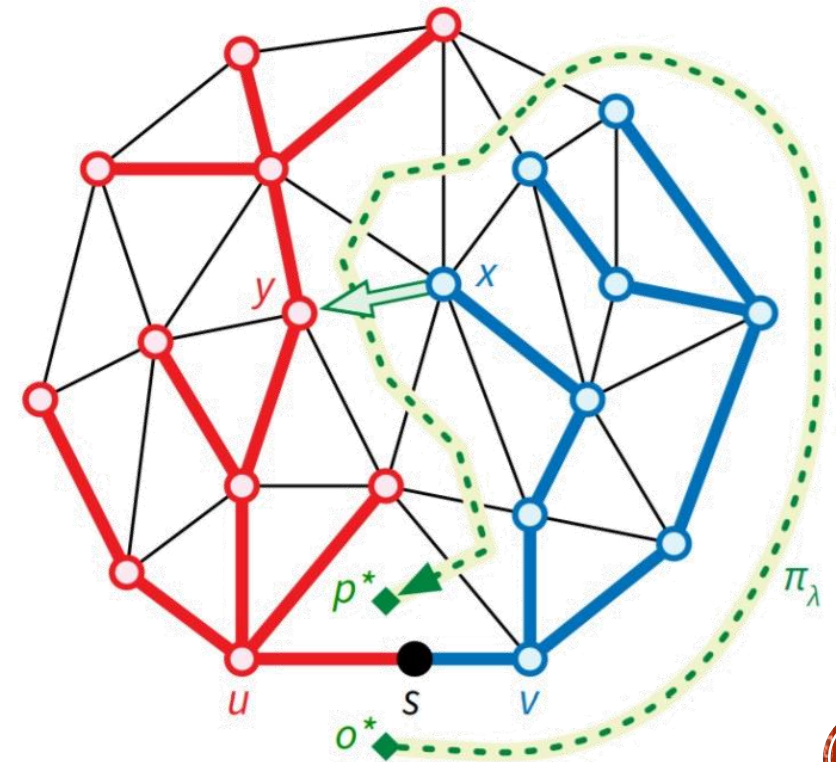
OBSERVATION. Any shortest-path tree has

- non-negative slack on all darts
- zero slack on tree darts
- positive slack on non-tree darts



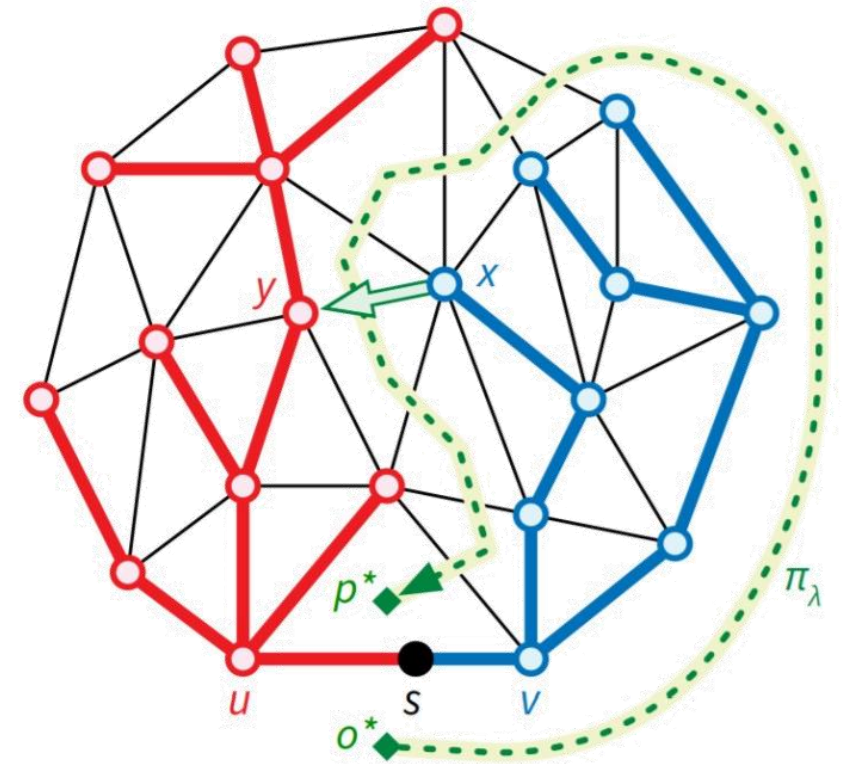
PARAMETRIC SHORTEST PATHS

- A vertex x is
 - **red** if $d_\lambda(x)$ goes up as λ goes up
 - **blue** if $d_\lambda(x)$ goes down as λ goes up
- Dart $x \rightarrow y$ is **active** if
 - $\text{slack}_\lambda(x \rightarrow y)$ goes up as λ goes up



RED-BLUE LEMMA. For any λ :

- All vertices behind u are **red**
- All vertices in front of v are **blue**
- $x \rightarrow y$ active if x **blue** and y **red**



COROLLARY.

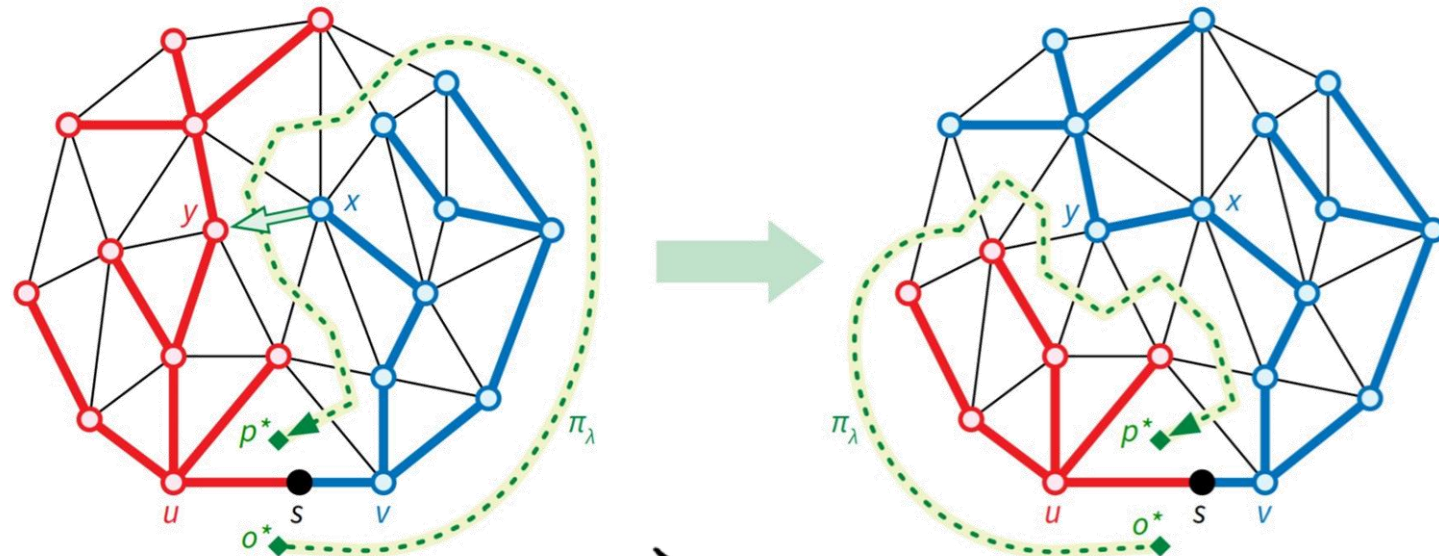
Active darts form dual path π_λ between o^* and p^* .

COROLLARY.

The min-slack active dart on π_λ is the next pivot.



MSSP ALGORITHM



NEXTPIVOT (G, T_λ):

$x \rightarrow y \leftarrow \text{MINPATHSLACK}(\theta^*, p^*)$
 $\Delta \leftarrow \text{slack}_\lambda(x \rightarrow y) / 2$

if $\lambda + \Delta / w(u \rightarrow v) < 1$:
 | $\text{PIVOT}(x \rightarrow y, \Delta)$
 | return $\lambda + \Delta / w(u \rightarrow v)$

else
 | return 1.

PIVOT ($x \rightarrow y, \Delta$):

ADDSUBTREEDIST (Δ, u)
 ADDSUBTREEDIST ($-\Delta, v$)

ADDPATHSLACK ($-2\Delta, \theta^*, p^*$)

$z \leftarrow \text{pred}(y)$, $\text{pred}(y) \leftarrow x$

CUT(yz), LINK(xy)
 CUT($(xy)^*$), LINK($(z \rightarrow y)^*$).

IMPLEMENTATION AND ANALYSIS

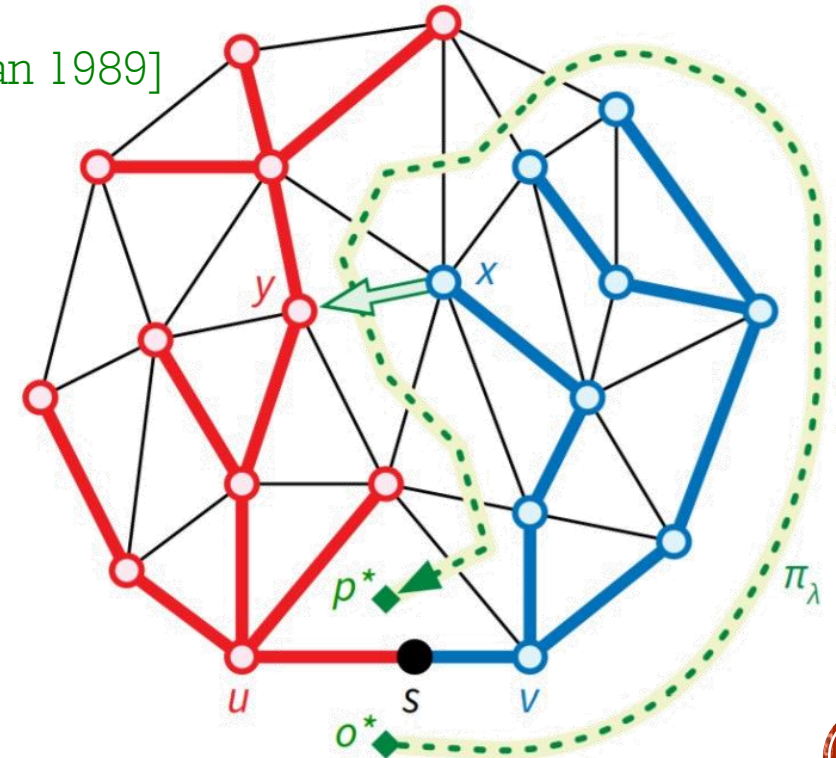
- Implement tree-cotree using dynamic tree data structure

- Splay tree into link-cut tree [Sleator-Tarjan 1982-1985]
- Persistent data structure [Driscoll-Sarnak-Sleator-Tarjan 1989]

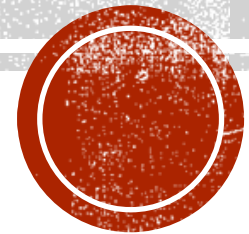
- Summary:

- $O(n)$ pivots (by disk-tree lemma)
- Correctly identify next pivot (by red-blue lemma)
- $O(\log n)$ amortized update time (by data structure magic)

- Thus $O(n \log n)$ time in total



TOPOLOGY+DATA STRUCTURE= FAST ALGORITHM



NEXT TIME:

**Two more tools from the toolbox
assemble our faster min-cut algorithm**