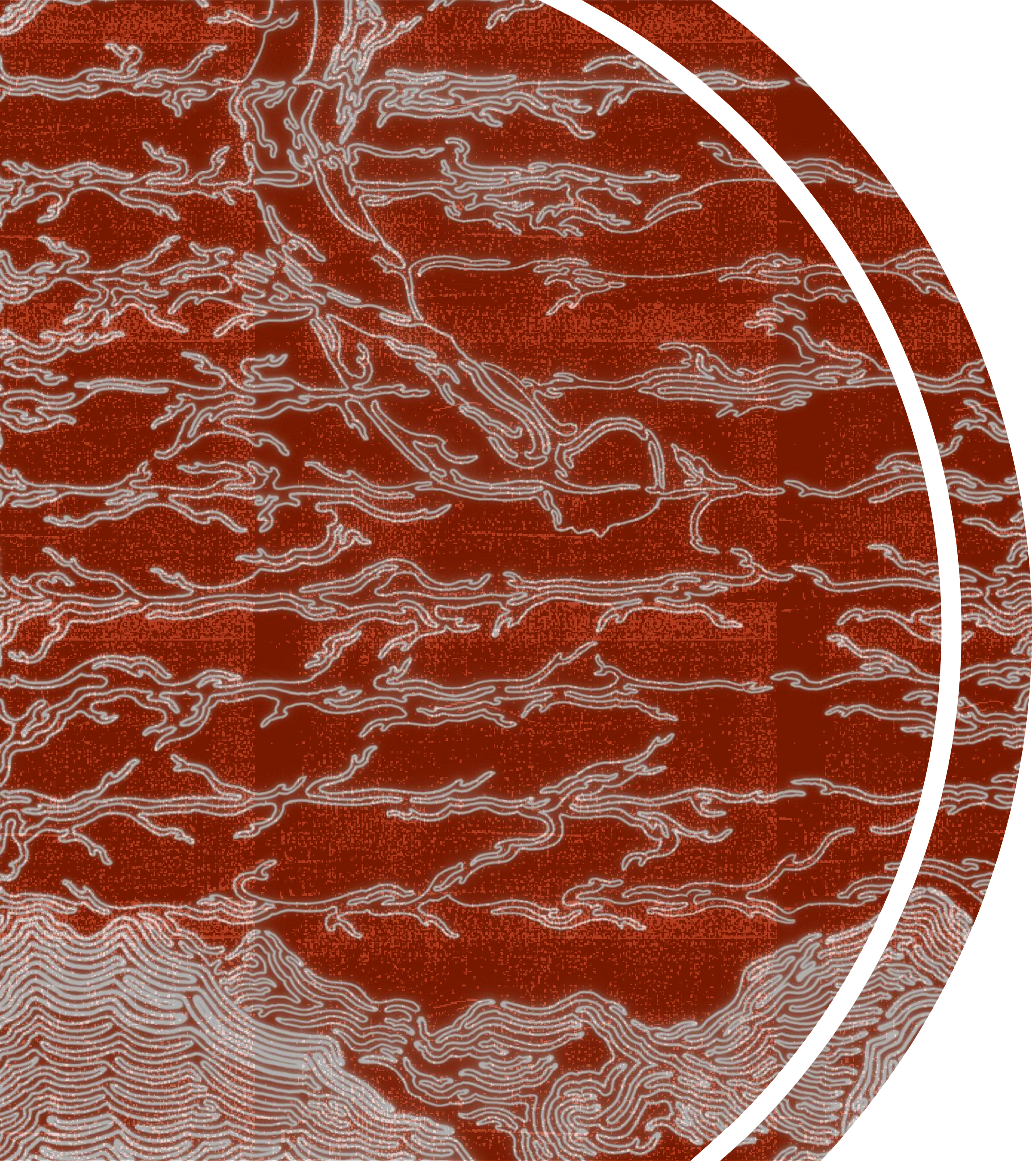


# Introduction to Computational Topology

Hsien-Chih Chang

Lecture 10, October 14, 2021

# Cycle Separator and r-Division

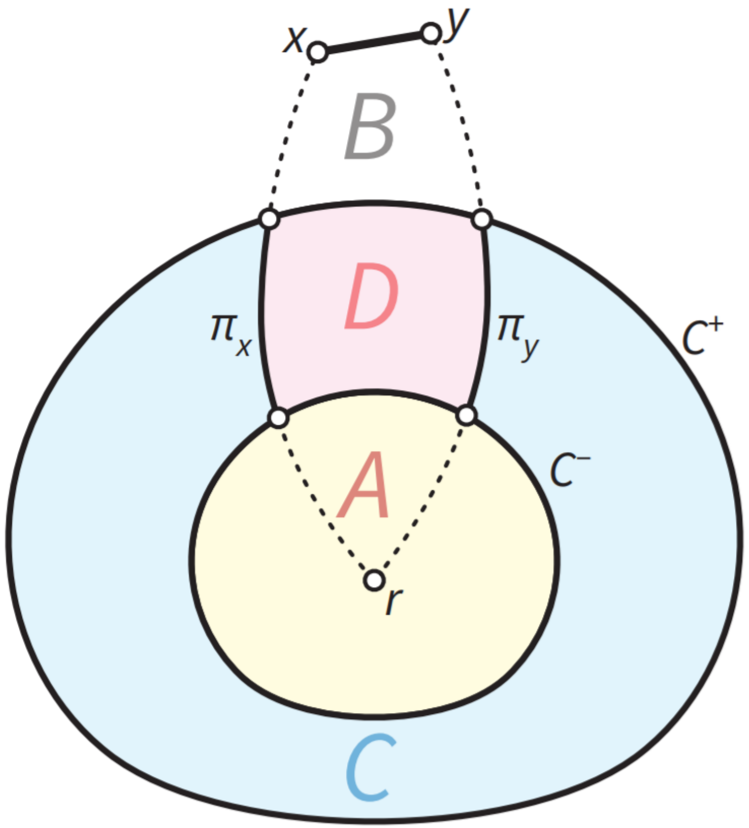
# Goal: r-Division [Frederickson 1989] [Klein-Mozes-Sommer 2012]

- Decompose the plane graph G into roughly equal size pieces
  - each piece has size $\leq$ r
  - #pieces at most $O(n/r)$
  - #boundary vertices per piece $\leq O(r^{1/2})$
  - $O(1)$ holes per piece

# Separator

- A separator is a vertex subset C such that [Lipton-Tarjan 1979]
  - $|C| \leq O(n^{1/2})$
  - $G - S = A \cup B$ and $|A|,|B| \leq 3n/4$
  - G can be vertex-weighted!

- Cycle separator: vertices of C forms a simple cycle




# Cycle Separator Theorem [Miller 1986] [Har-Peled Nayyeri 2018]

Cycle separator can be found in O(n) time

# Finding Cycles

- Compute BFS tree $T_{BFS}$
- Level of a triangle face: max among levels of three vertices
- $R_{\leq i}$: region with face levels at most i

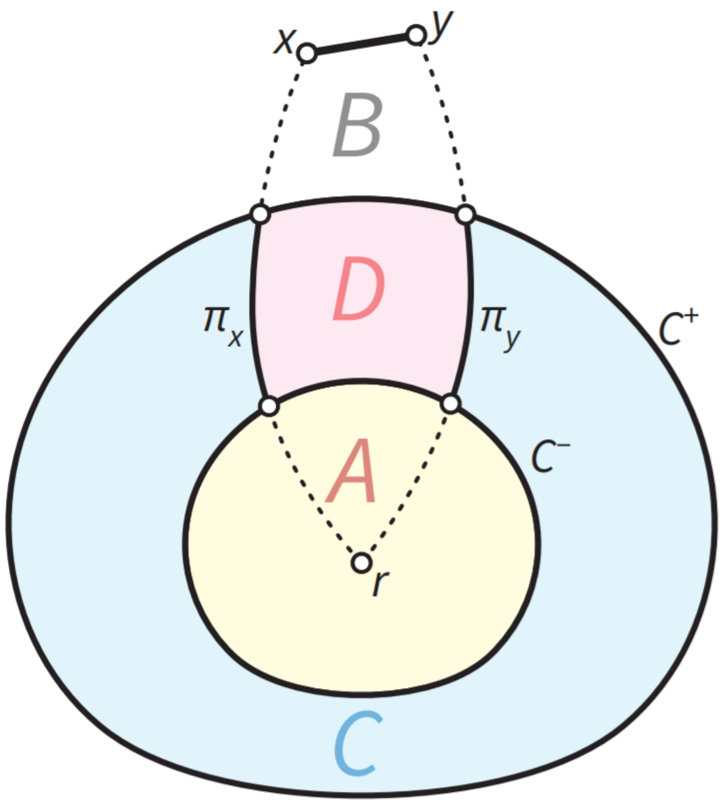
- **Lemma.** Boundaries of $R_{\leq i}$ are pairwise disjoint simple cycles $C_i$

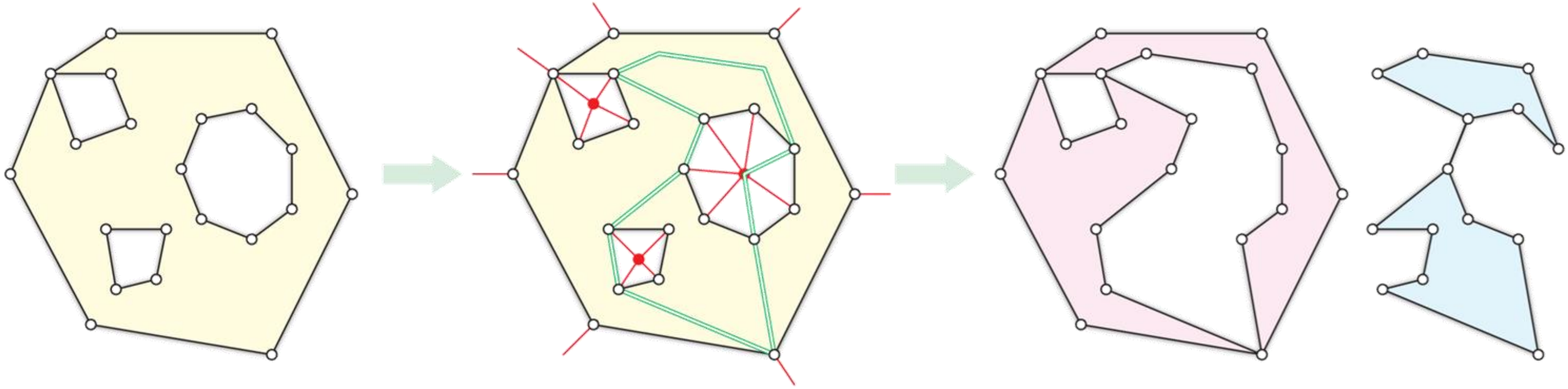
# Finding Separator

- Find fundamental cycle separator cycle($T_{BFS}$, uv); reroot to lca(u,v)

- **Lemma.** cycle($T_{BFS}$, uv) intersects each $C_i$ at most twice

# Finding Cycle Separator





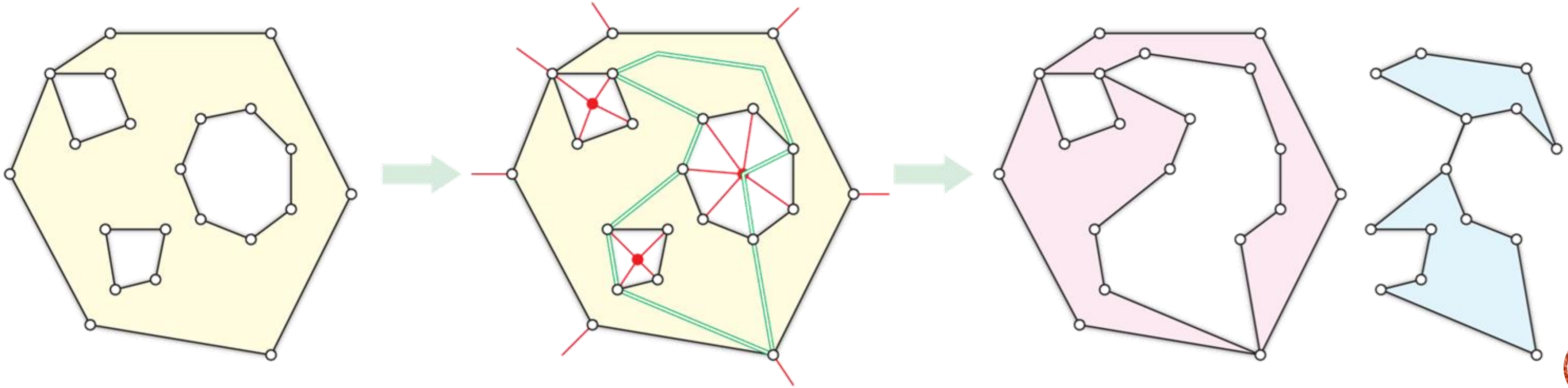
# Efficient r-Division

[Frederickson 1989] [Goodrich 1995]
[Klein-Mozes-Sommer 2012]

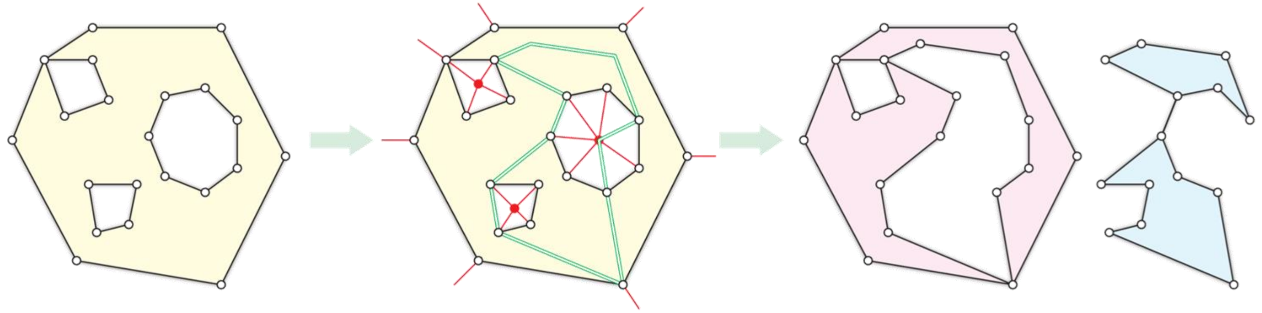
r-division can be computed in O(n) time

# To Get r-Division

- Iteratively find cycle separators. At level i:
  - If i mod 3 = 0: Separate vertices evenly
  - If i mod 3 = 1: Separate boundary vertices evenly
  - If i mod 3 = 2: Separate holes evenly



# To Get r-Division



- Iteratively find cycle separators. At level i:
  - If i mod 3 = 0: Separate vertices evenly
  - If i mod 3 = 1: Separate boundary vertices evenly
  - If i mod 3 = 2: Separate holes evenly

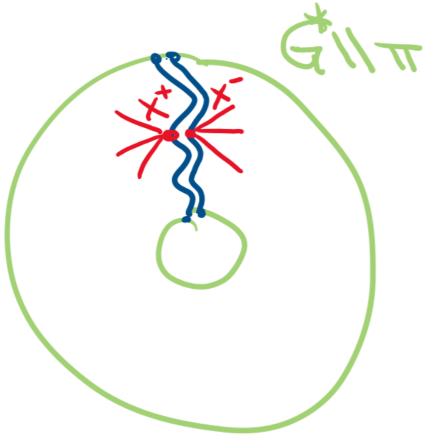
- #vertices, #bdry vertices, #holes all decrease by O(1) factor after 3 levels
- O(n log (n/r)) time naïvely; dynamic tree to the rescue

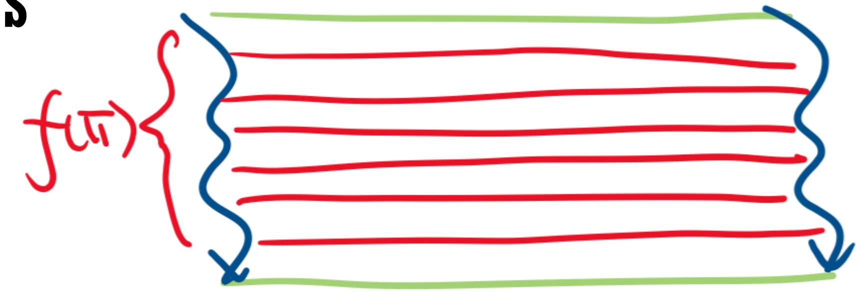
# Toolbox to be Built

- 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]

# Master Plan for Min-Cut Algorithm

- Compute r-division for plane graph G
- Compute APSP between bdry vertices per piece using MSSP
- Replace each piece with a complete graph on bdry vertices
- Compute n/log n parallel shortest paths for Reif's



# Intermission

Just enjoy the break.

# Wanted: a Sublinear-Size Representation of a Piece

# APSP Distance Around a Piece

- Distance matrix D: k-by-k array where each entry

$$D[i, j] = d_P(s_i, s_j)$$

- Four vertices $s_1, \ldots, s_4$ around P satisfies cyclic **Monge Property** [Monge 1781]

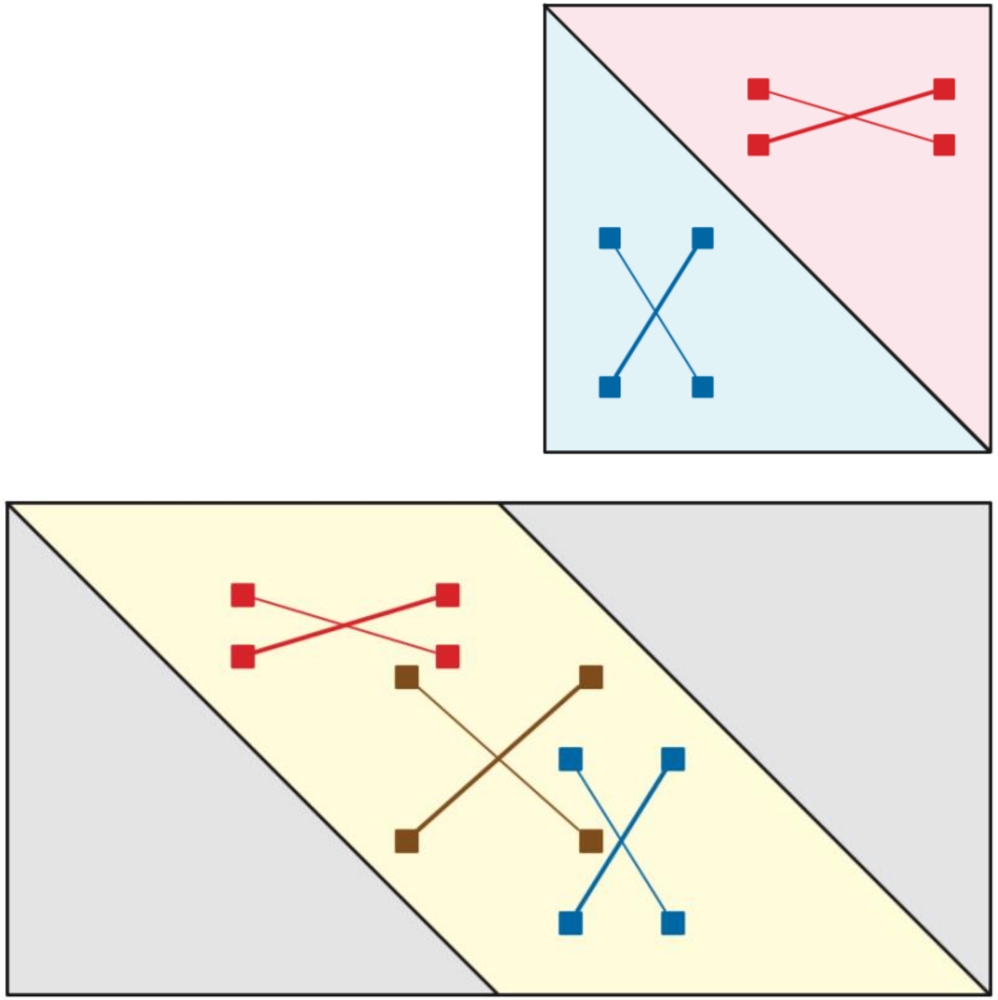
$$d_P(s_1, s_2) + d_P(s_3, s_4) \le d_P(s_1, s_3) + d_P(s_2, s_4)$$

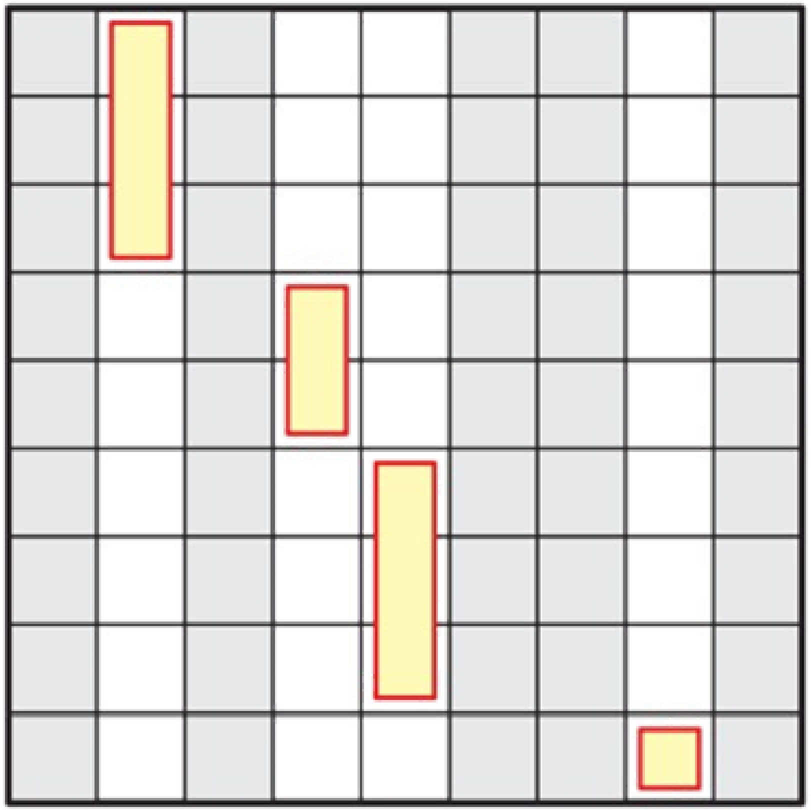
# Monge Property [Monge 1781]

$$d_P(i_1, j_1) + d_P(i_2, j_2) \leq d_P(i_1, j_2) + d_P(i_2, j_1)$$

$$\begin{bmatrix} 10 & 17 & 13 & 28 & 23 \\ 17 & 22 & 16 & 29 & 23 \\ 24 & 28 & 22 & 34 & 24 \\ 11 & 13 & 6 & 17 & 7 \\ 45 & 44 & 32 & 37 & 23 \\ 36 & 33 & 19 & 21 & 6 \\ 75 & 66 & 51 & 53 & 34 \end{bmatrix}$$

**Lemma.** Matrix D decomposes into two Monge matrices.





# SMAWK Algorithm

[Aggarwal-Klawe-Moran-Shor-Wilber 1987]
[Klawe-Kleitman 1990]

All row-wise minimum elements of a k-by-k Monge matrix can be found in O(k) time

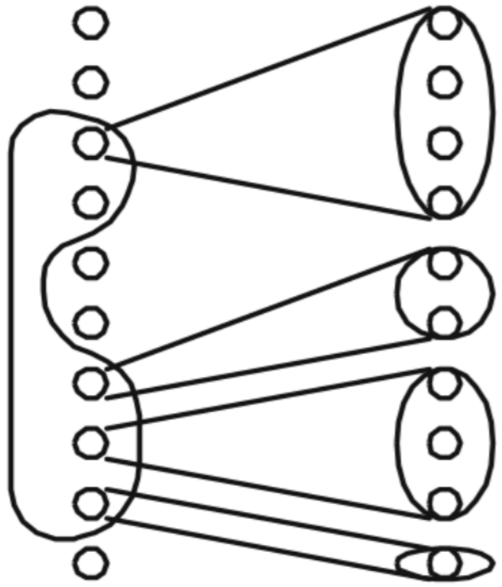
# Row-minimum in Matrix D

- Distance matrix D: k-by-k array where each entry

$$D[i, j] = d_P(s_i, s_j)$$

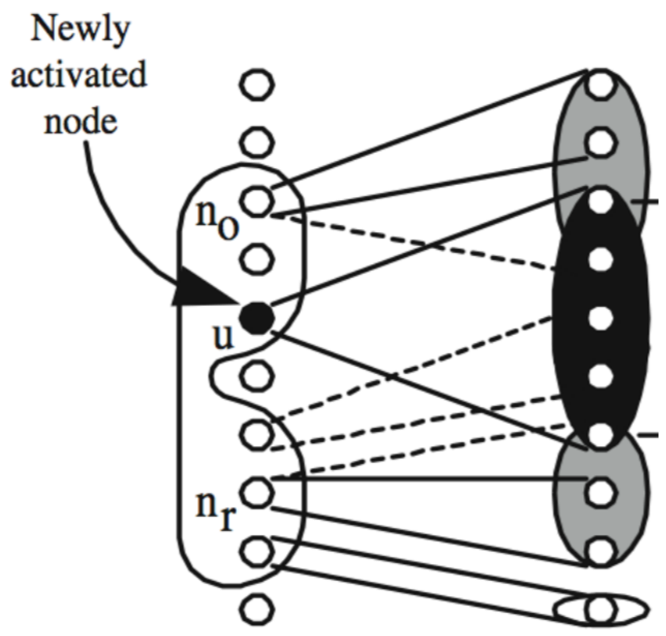
- Minimum element in row i: $\min_j d_P(s_i, s_j)$
  - Shortest "edge" going to vertex $s_i$

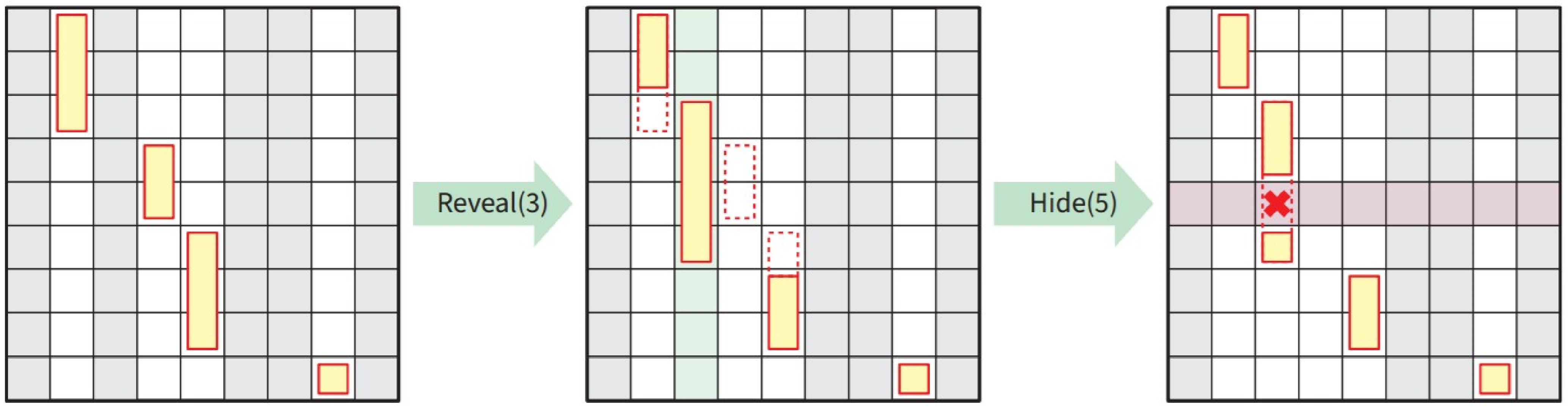
- Search matrix $M[i,j]=D[i,j]+c(j)$ for row-minimums



# Monge Heap

- Representation of matrix M, supporting
  - FindMin(): smallest visible element in M
  - Reveal(j, x): reveal column j by setting c(j) to x
  - Hide(i): hide row i



# FR-Dijkstra [Fakcharoenphol-Rao 2001]

## FR-Dijkstra

In all Monge heaps relevant to s:
- Reveal(s, 0)
- Hide(s)

Repeat until t hidden:
- v ← FindMin()
- In all Monge heaps relevant to v:
  - Reveal(v, d(s, v))
  - Hide(v)

Return dist(s, t)

# Analysis of FR-Dijkstra

```
FR-DIJKSTRA

In all Monge heaps relevant to s:
  REVEAL(s, 0)
  HIDE(s)

Repeat until t hidden:
  v ← FINDMIN()

  In all Monge heaps relevant to v:
    REVEAL(v, d(s,v))
    HIDE(v)

Return dist(s,t)
```

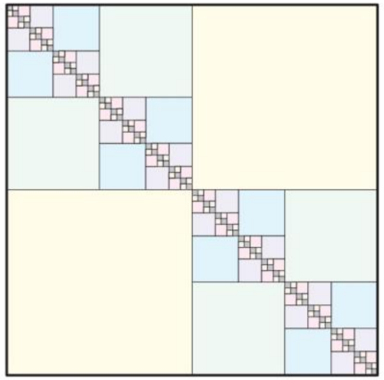
- per Monge heap: O(k log k)
- Overall:

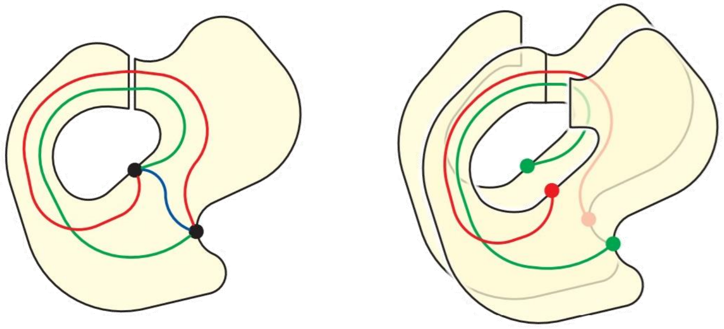
# Analysis for Fast Min-Cut Algorithm

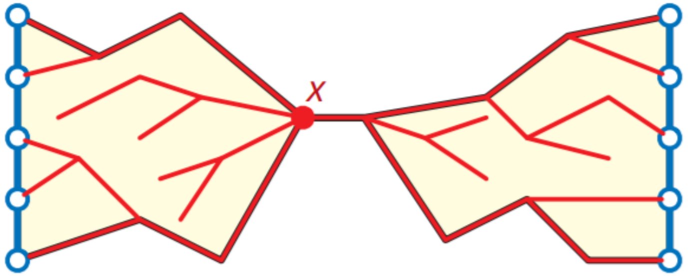
- Compute r-division for plane graph G
- Compute APSP between bdry vertices per piece using MSSP
- Replace each piece with Monge heaps on bdry vertices
- Compute n/log n parallel shortest paths using FR-Dijkstra
- Recursion as in Reif

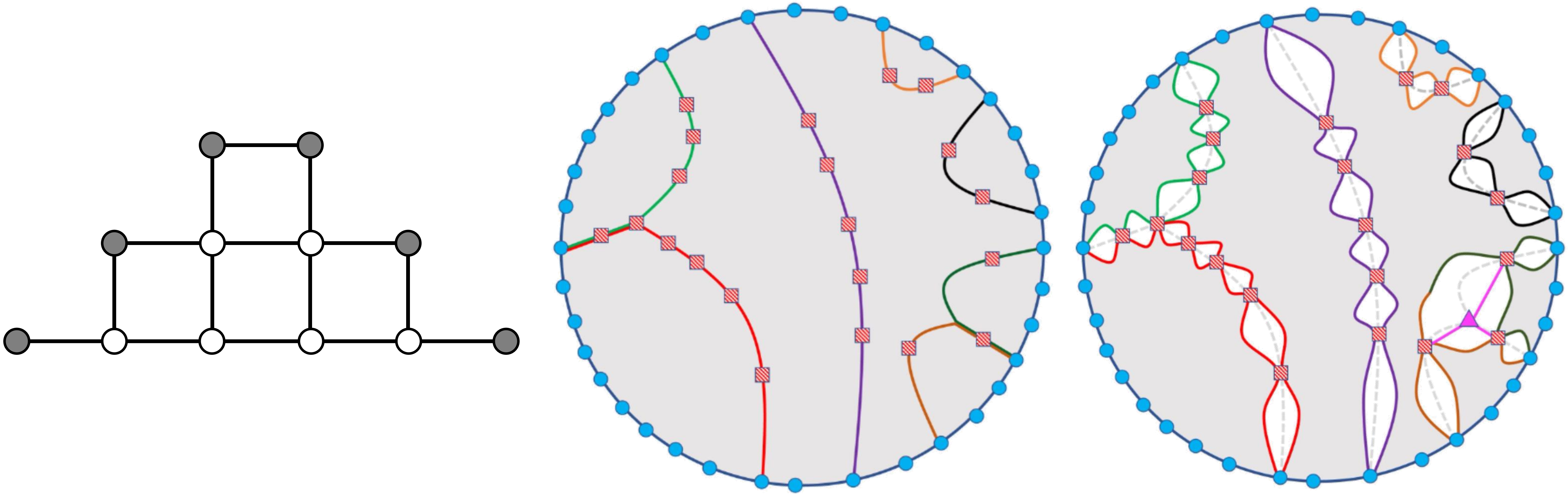
# UNDER THE RUG

- Monge heap only works for Monge matrix
- Multiple holes
- r-division needs to respect strips
- Degenerate strips
- Actual shortest path needed from Dijkstra to cut
- O(1)-degree assumptions
- …





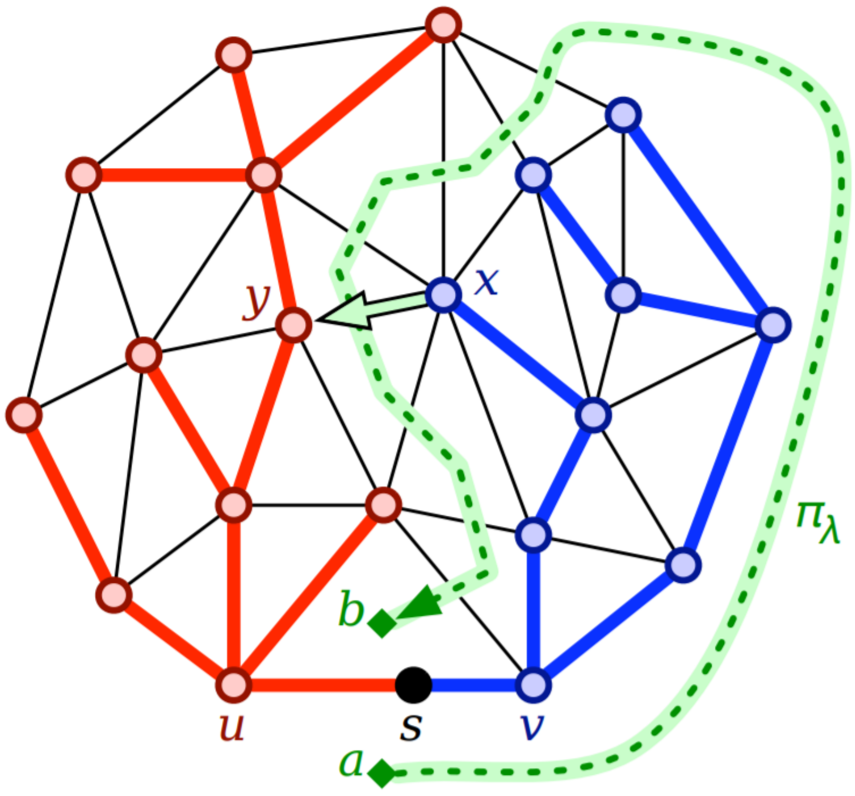


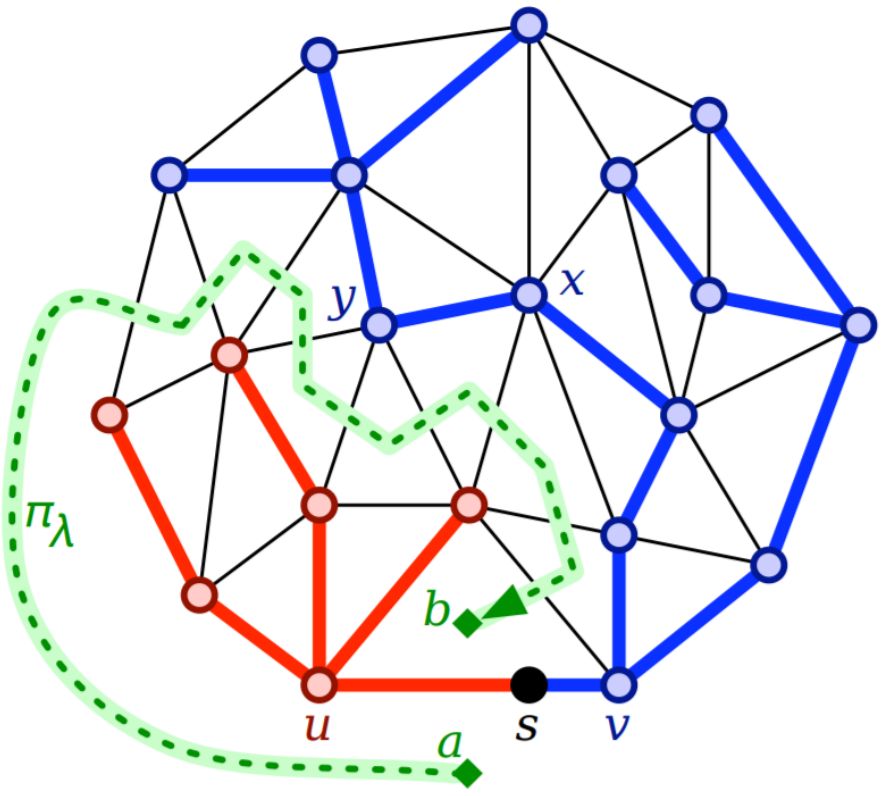


# Planar Emulators

[Chang-Ophelders 2020] [Chang-Krauthgamer-Tan 2022]

Every piece with k bdry vertices has
(1) an exact planar emulator of size $k^2$, or
(2) an planar $\varepsilon$-emulator of size $O(k \text{ polylog } k/\varepsilon)$






# Multiple-Source ε-Shortest Paths [Chang-Krauthgamer-Tan 2022]

ε-MSSP problem can be solved in $O(n \log^* n)$ time

# Analysis for Faster Min-Cut Algorithm

- Compute r-division for plane graph G
- Compute APSP between bdry vertices per piece using MSSP
- Replace each piece with Monge heaps on bdry vertices
- Compute n/log n parallel shortest paths using FR-Dijkstra
- Recursion as in Reif

# Algorithmic Engineering is a thing

## Next Time:

Homology: a better tool to classify spaces