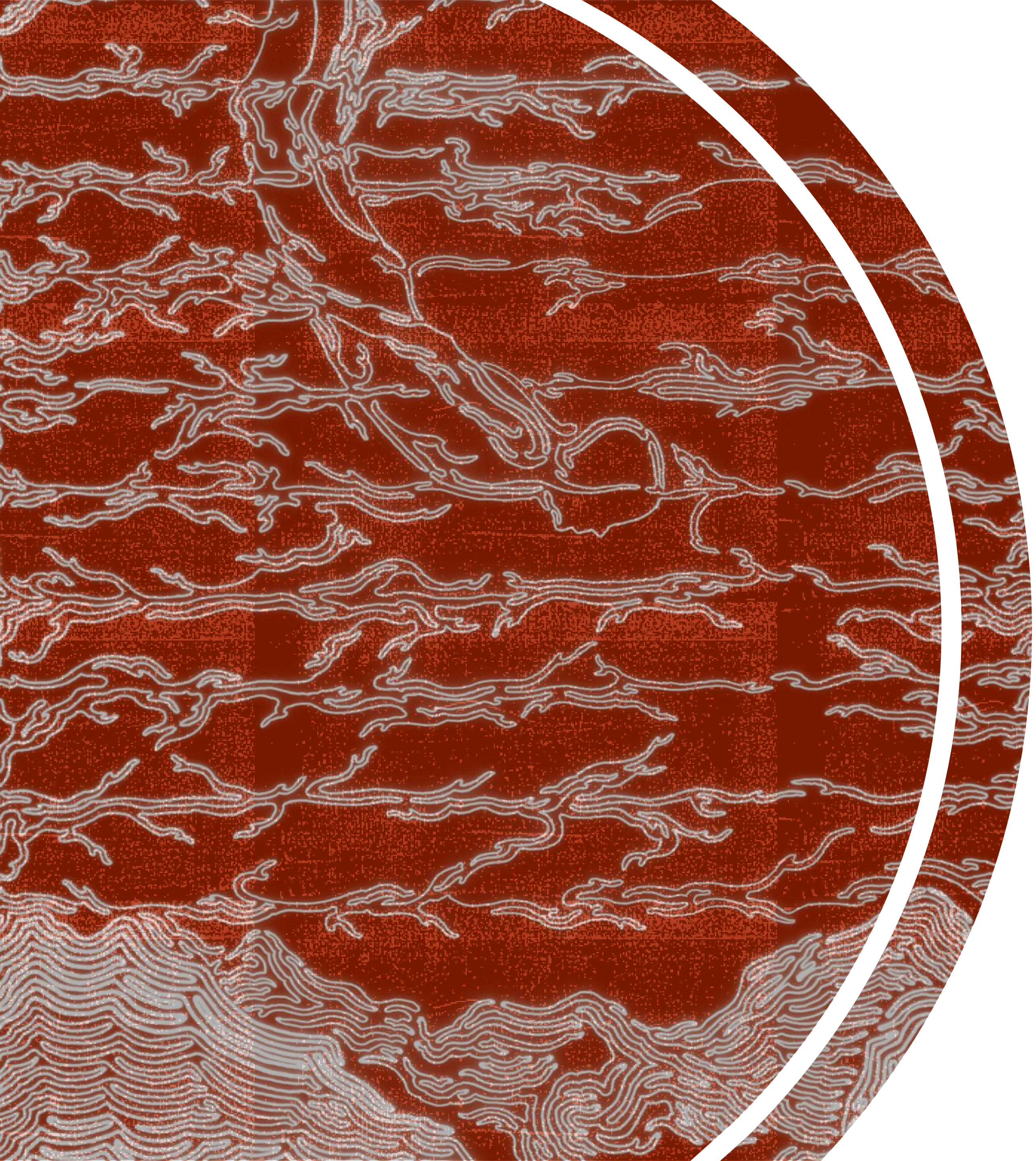


Introduction to Computational Topology

Hsien-Chih Chang

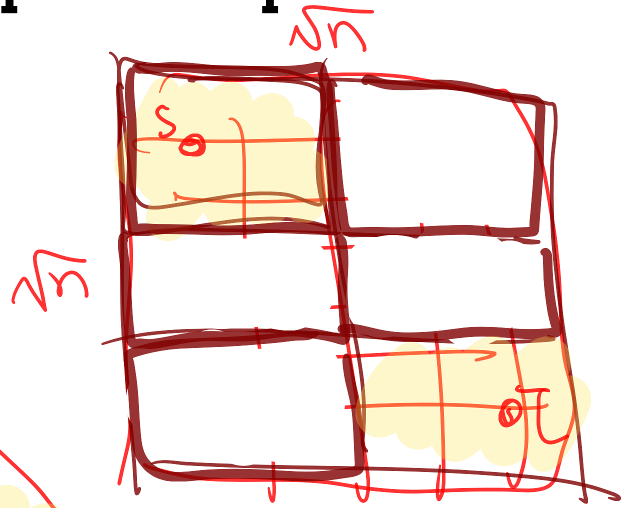
Lecture 10, October 14, 2021

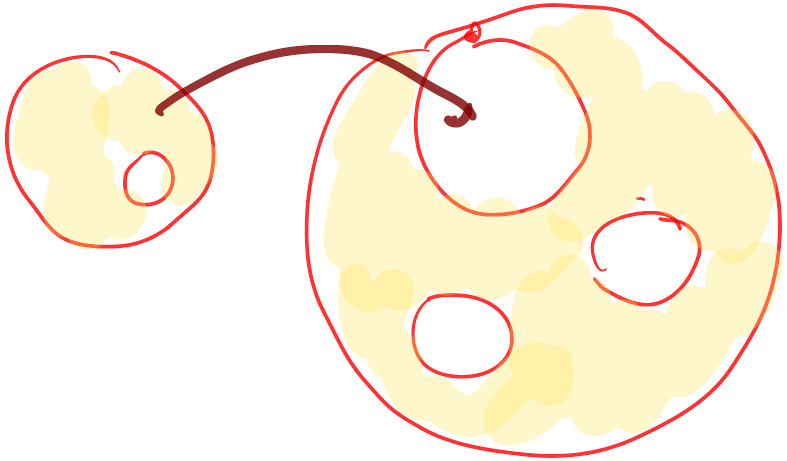
Cycle Separator and r-Division

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





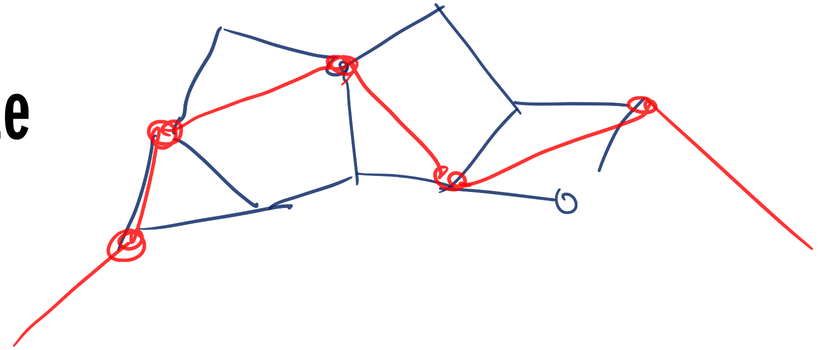
Separator

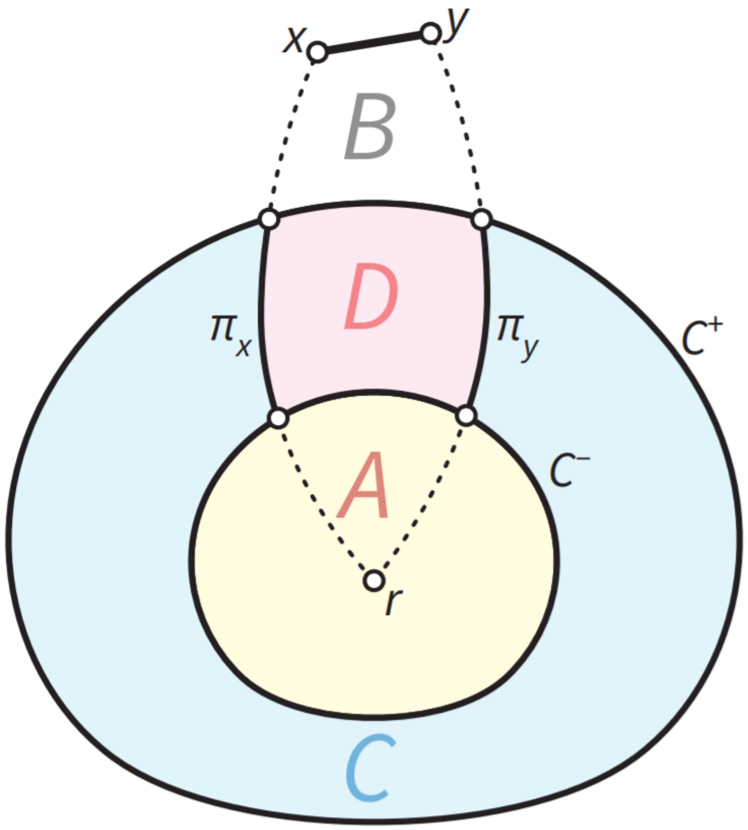
- A separator is a vertex subset C such that [Lipton-Tarjan 1979]
  - $|C| \leq O(n^{1/2})$
  - $G - C = A \cup B$ and $|A|,|B| \leq 3\overset{W}{\cancel{n}}/4$
  - G can be vertex-weighted!

$w : V(G) \to \mathbb{R}_+$

$\sum_{v \in G} w(v) =: W$

- Cycle separator: vertices of C forms a simple cycle



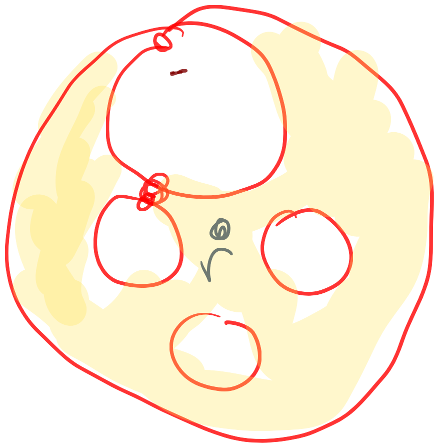


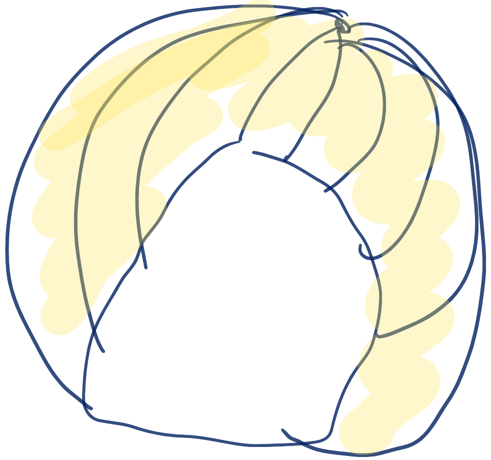
Cycle Separator Theorem

[Miller 1986] [Har-Peled Nayyeri 2018]

Cycle separator can be found in O(n) time

Finding Cycles





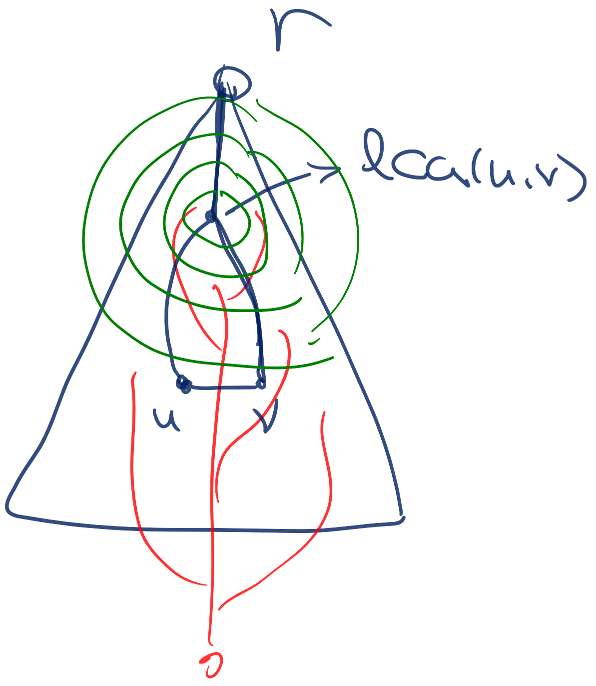
- Compute BFS tree T_{BFS}
- Level of a triangle face: max among levels of three vertices
- $R_{\leq i}$: region with face levels at most i

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

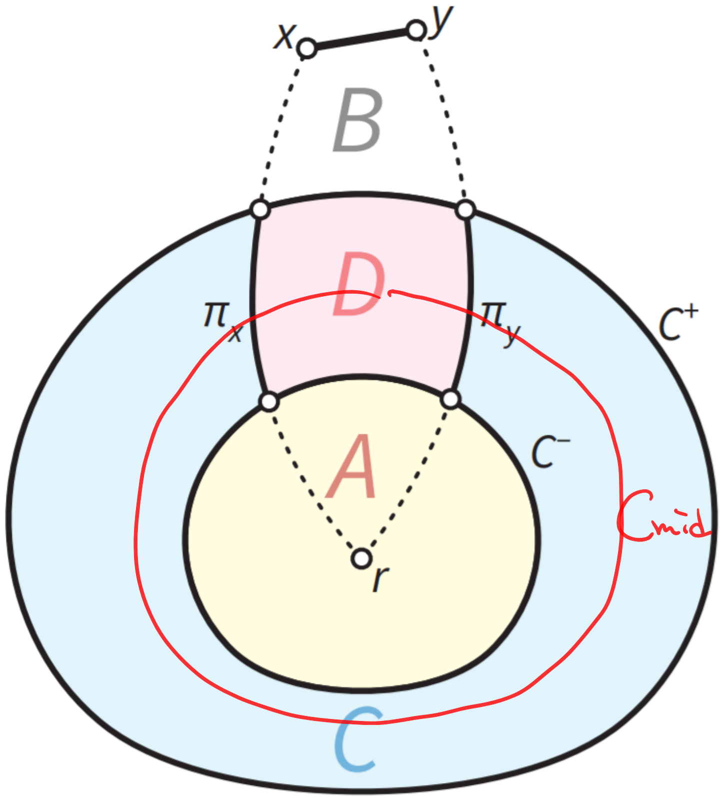
Finding Separator

- 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



C_{mid} : last cycle containing $\leq \#F/2$

$$\overbrace{C^- \leq}^{\sqrt{n}} \overbrace{C_{mid} \leq}^{\sqrt{n}} C^+$$

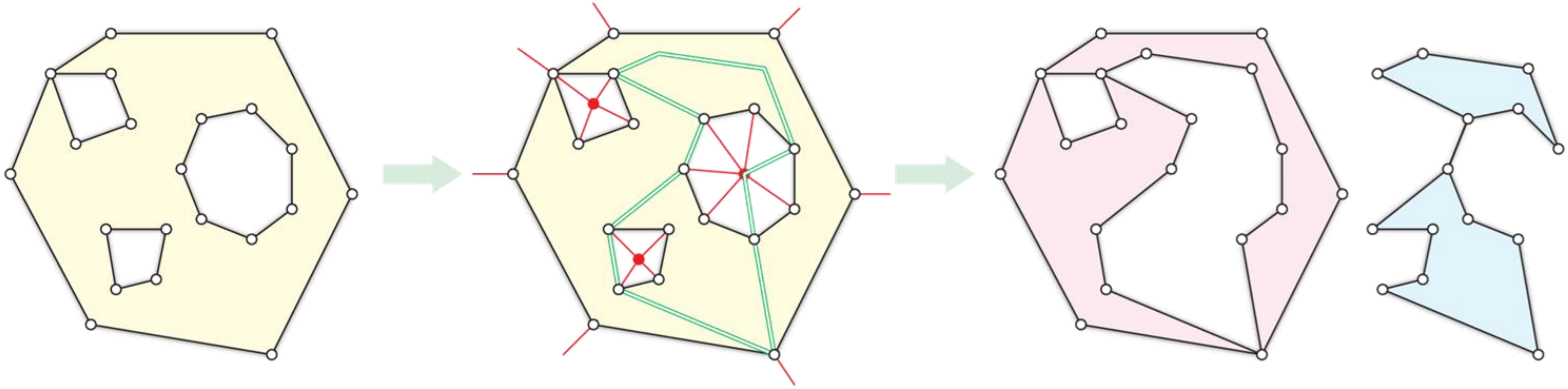
$|C^-|, |C^+| \leq O(\sqrt{n})$

otherwise if all $\sqrt{n}$ levels of cycles near C_{mid} has size $> \sqrt{n}$, we exhaust all vertices as C_i's are disjoint.

Claim $|A|, |B| \leq \#F/2$

Claim. $|C|, |D| \leq 3/4 \cdot \#F$

Claim. One of A, B, C, D has size $\geq \#F/4$



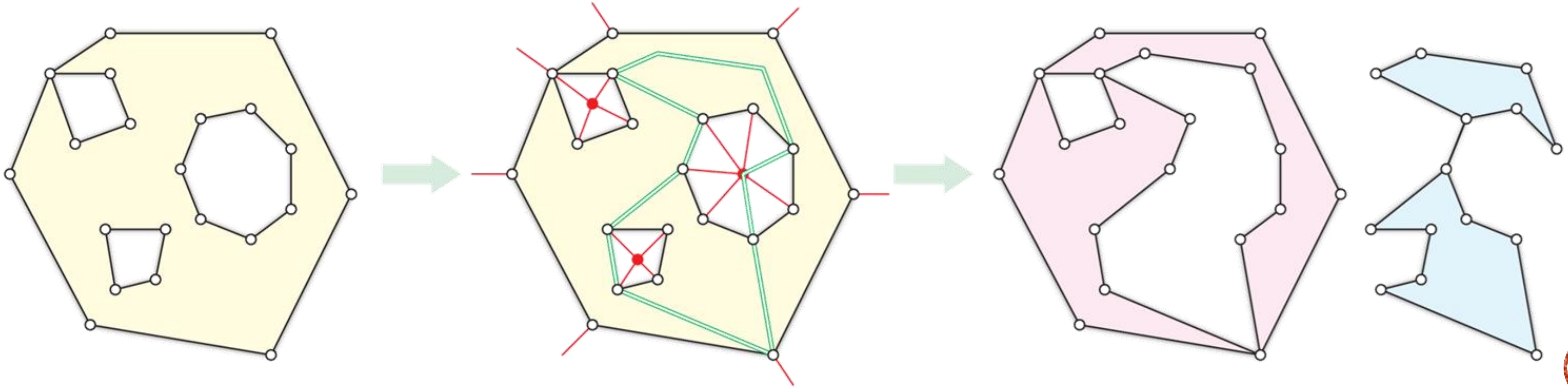
Efficient r-Division

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

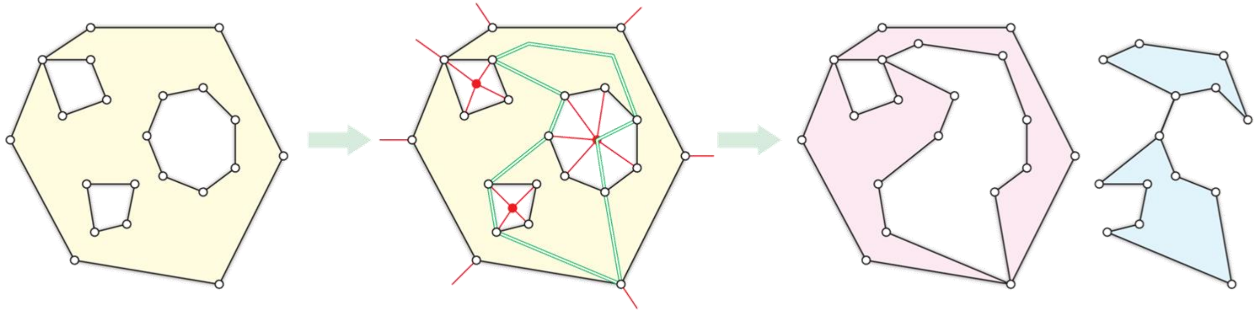
r-division can be computed in O(n) time

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

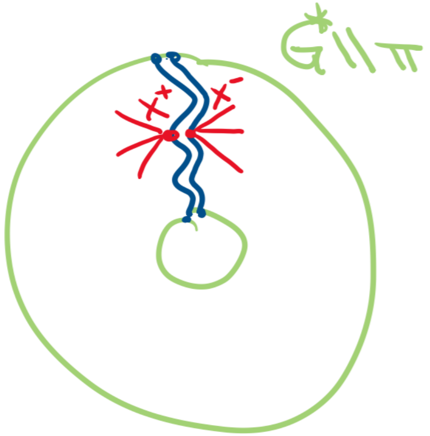
Toolbox to be Built

- 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

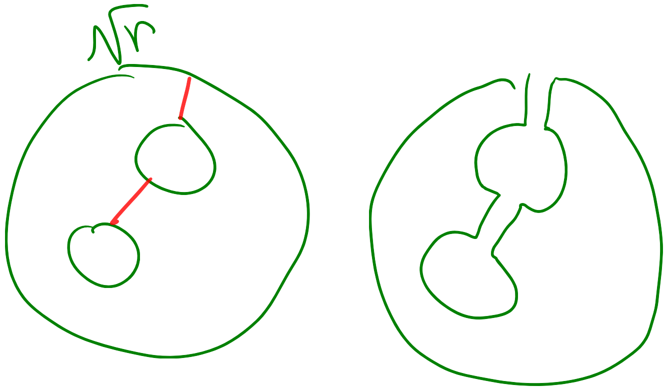


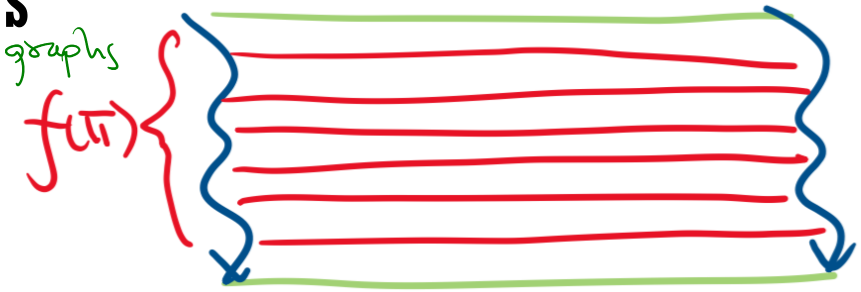
time for MSSP & complete graphs

$\sqrt{r} \cdot \sqrt{r} \cdot \log r$

$= r \log r$ per piece

$\frac{n}{r} \cdot r \log r = \boxed{n \log r}$





#edges: $\frac{n}{r} \cdot r = n$

Intermission

Just enjoy the break.

Wanted: a Sublinear-Size Representation of a Piece

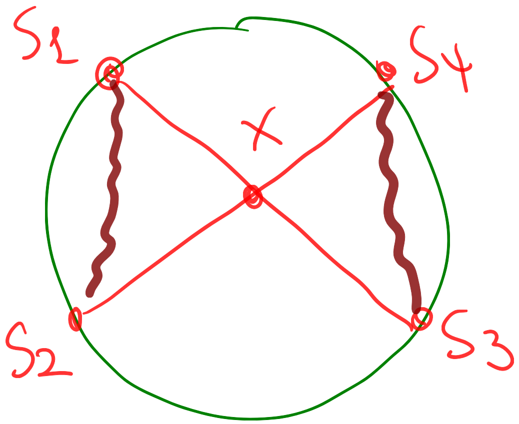
APSP Distance Around a Piece

- 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]

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



Monge Property

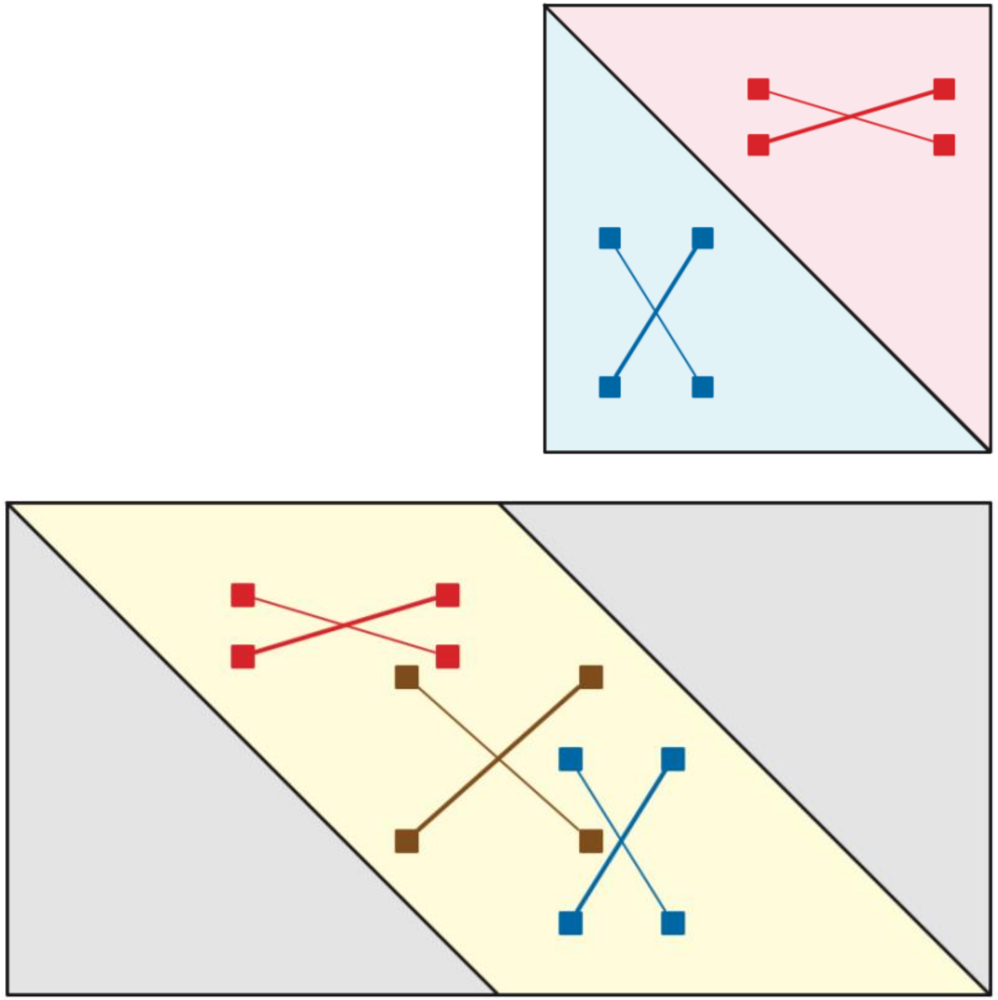
[Monge 1781]

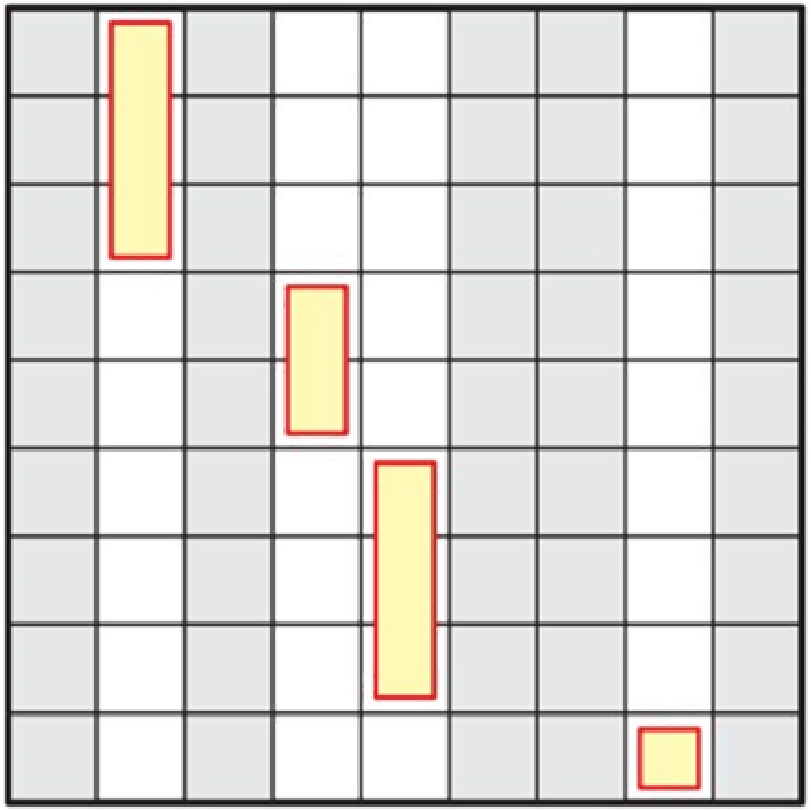
$$d(i_2, j_2) - d(i_2, j_1) \leq d(i_1, j_2) - d(i_1, j_2)$$

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

$$
\begin{array}{c c}
 & \begin{array}{c c c c c c} & j_1 & & j_2 & & \end{array} \\
\begin{array}{c} \\ i_1 \\ i_2 \\ \\ \\ \\ \\ \end{array} &
\begin{bmatrix}
\boxed{10} & 17 & -4 & 13 & 28 & 23 \\
17 & 22 & -6 & 16 & 29 & 23 \\
24 & 28 & -6 & 22 & 34 & 24 \\
11 & 13 & -7 & 6 & 17 & 7 \\
45 & 44 & -8 & 32 & 37 & 23 \\
36 & 33 & -14 & 19 & 21 & 6 \\
75 & 66 & -15 & 51 & 53 & 34
\end{bmatrix}
\end{array}
$$

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

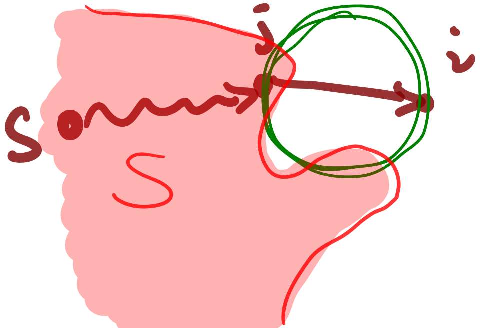
- Distance matrix D: k-by-k array where each entry

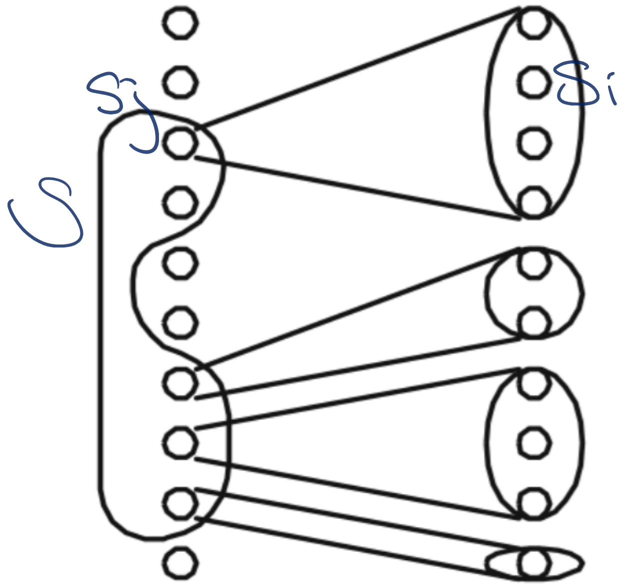
$$D[i, j] = d_P(s_i, s_j)$$

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

$d(s, s_j)$

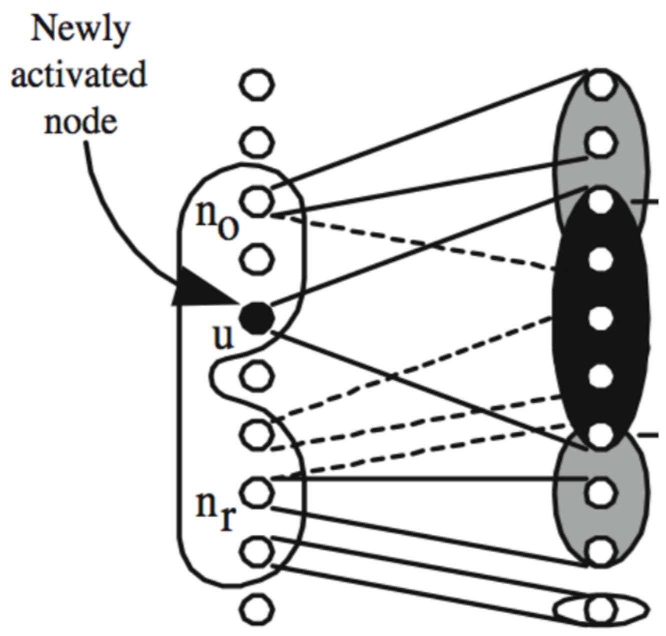
- Search matrix $M[i,j]=D[i,j]+c(j)$ for row-minimums

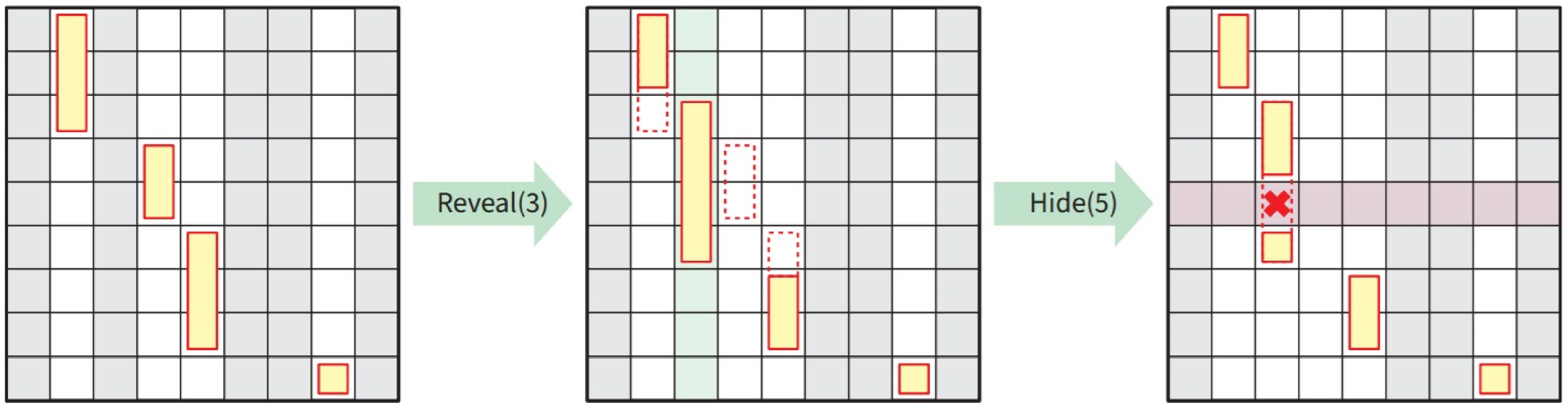




Monge Heap

- 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:
- $v \leftarrow$ FindMin()
- In all Monge heaps relevant to v:
  - Reveal(v, $d(s,v)$)
  - Hide(v)

Return dist(s,t)

every column is revealed once
every row is hidden once

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

Time for FR-Dijkstra:

$$O(\sqrt{r}\log r)\cdot O\left(\frac{n}{r}\right) = O\left(\frac{n}{\sqrt{r}}\log r\right)$$

Time for Monge heap:

$$O(r\log r)\cdot O\left(\frac{n}{r}\right) = O(n\log r)$$

Analysis for Fast Min-Cut Algorithm

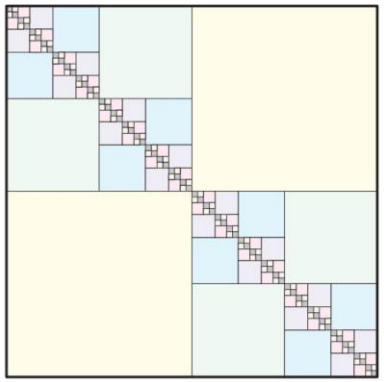
- Compute r-division for plane graph G $O(n)$
- Compute APSP between bdry vertices per piece using MSSP $O(n \log r)$
- Replace each piece with Monge heaps on bdry vertices $O(n \log r)$ 1
- Compute n/log n parallel shortest paths using FR-Dijkstra $O(\frac{n}{\sqrt{r}} \log n)$
- Recursion as in Reif $\cdot\ O(\log \frac{n}{\log n})$ 2

$\frac{n}{r} \cdot \boxed{r \log r}$ MSSP

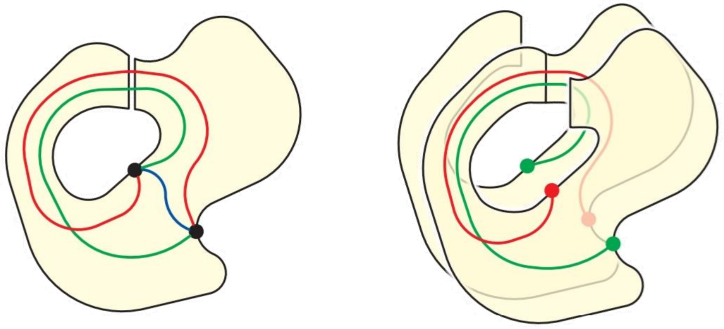
Set $r = \log^4 n$

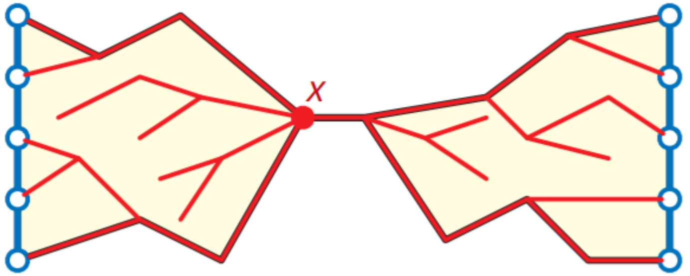
$$\underbrace{O(n \log r)}_{1} + \underbrace{O\left(\frac{n}{\sqrt{r}} \log^2 n\right)}_{2} = O(n \log \log n)$$

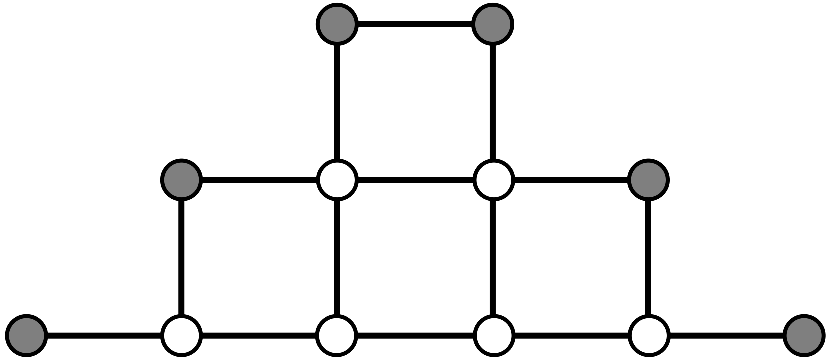
Under the Rug

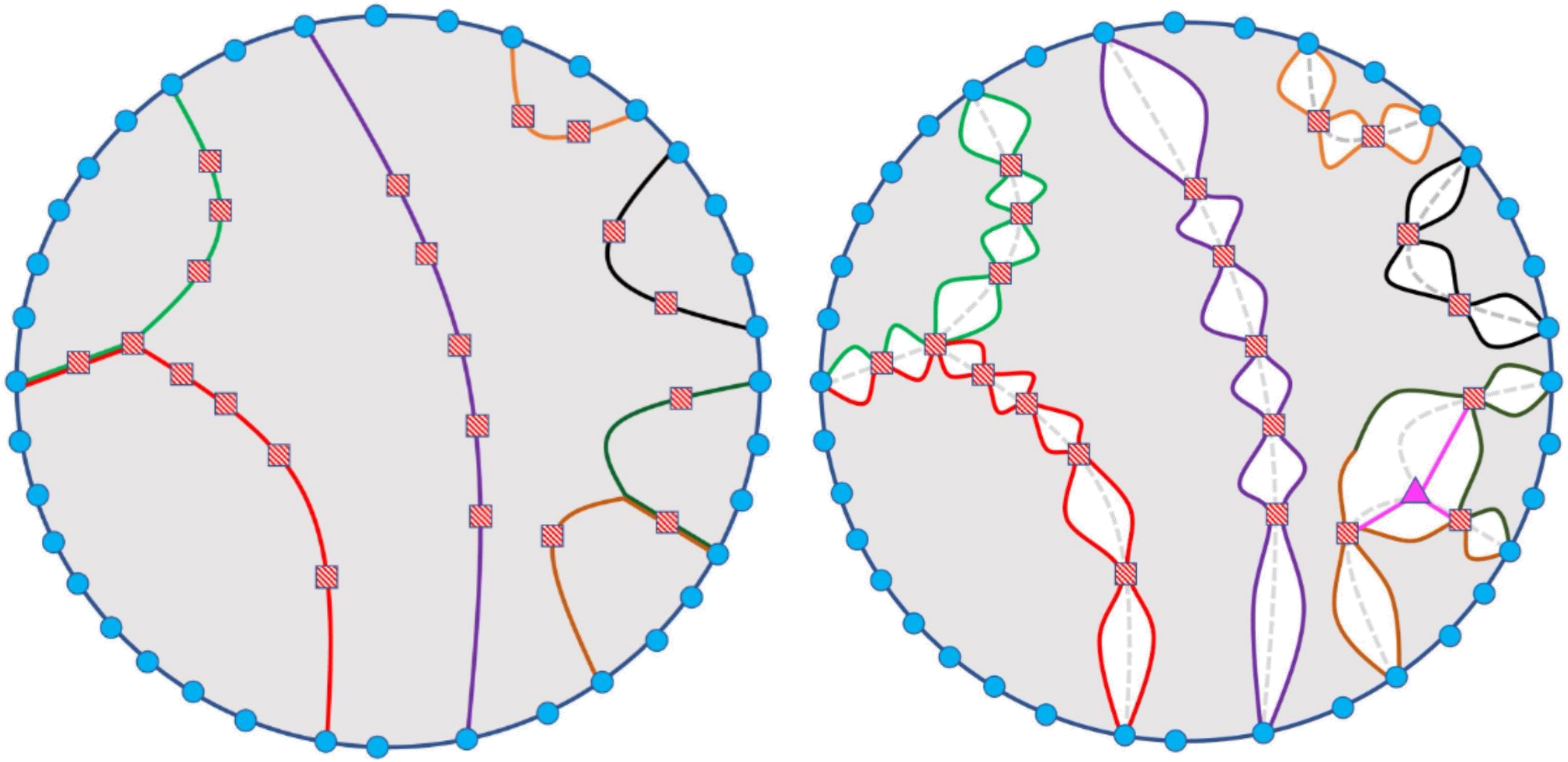


- Monge heap only works for Monge matrix
- Multiple holes
- r-division needs to respect strips
- Degenerate strips
- Actual shortest path needed from Dijkstra to cut
- O(1)-degree assumptions
- …





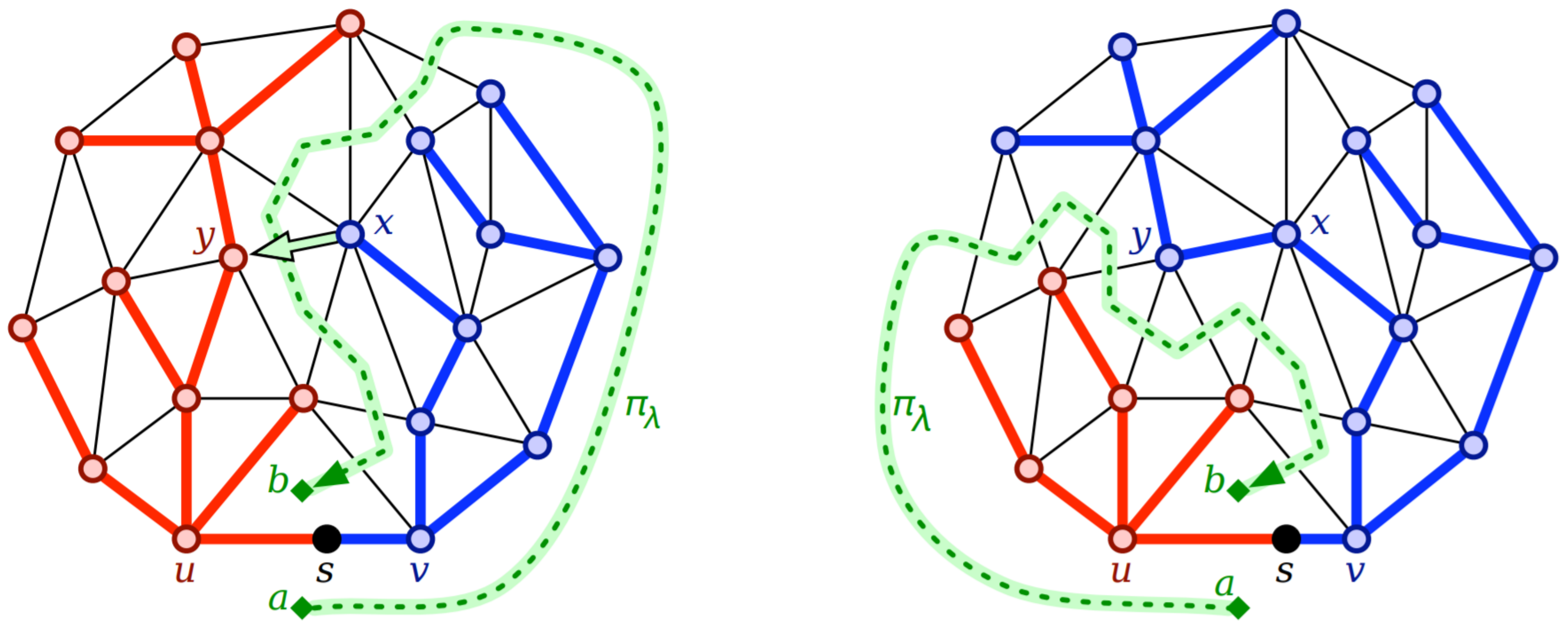




Planar Emulators

[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 ε-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 $O(\frac{n}{r} \cdot r \log^* r)$
- Replace each piece with Monge heaps on bdry vertices
- Compute n/log n parallel shortest paths using FR-Dijkstra
- Recursion as in Reif

$r = \log^4 n$

$$O(n \log^* r) + O\left(\frac{n}{\sqrt{r}} \log^2 n\right) = O(n \log^* n)$$

Algorithmic Engineering is a thing

Next Time:

Homology: a better tool to classify spaces