Let's try this again.

**Monge heap:**

Given a Monge array $D$, and "given" an unknown vector $c$

Define $M[i, j] = D[i, j] + c[j]$ — Monge

No matter what's in $c$

Three operations:

- **Reveal**($j, x$) — Define $c[j] = x$, "reveling" column $j$ of $M$
- **Find Min** — Report minimum visible element
- **Hide**($i$) — Hide row $i$ of $M$ ("Extract Min")

Ultimately, we will find the min of every row of $M$

*Find Min results must be the true row minima*

---

**Structure:** Revealed cols $x$ unhidden rows is Monge

$\Rightarrow$ row mins are monotone

For each live column $j$:

- Maintain row intervals $[l, r]$ where min element is in col $j$.
- Keep triples $(j, l_{min}, r_{max})$ in a min-heap, keyed by minimum element.

$O(\log k)$ insert, delete, extractmin, $2k$ triples in heap

- Min element in any subcolumn depends only on $D$

**Prep each column of $D$ for range minimum queries**

Given $l_{min}, r_{max}$, find $i_{min}$ is $i_{max}$ minimizing $D[i, j]$

Balanced (static) binary tree with $k$ leaves

- $i_{th}$ leaf stores $D[i, j]$
- Internal node $u$ stores min of children
**Reveal (j, x):**
- Maintain live cols in BST
- Find prev and next visible columns
- Binary search in col j- for start of col j’s interval
- Binary search in col j+ for end of col j’s interval
- Remove triples (j-, i-, -) that overlap (j, i-, i+)
- Add triples (j-, i-, -), (j-, i-, -), (j+, i-, -)

**Overall amortized time:** \(O(\log k)\)

**Find Min** — \(O(1)\) time, at top of global heap

**Hide (i):**
- Maintain inactive intervals in BST
- Now i intersects at most one triple (j-, i-, i+) in global heap
- Remove that triple.
- Insert (j-, i-, i-1) and (j+, i+, i+1)
- We don’t have to worry about other cols:
  - Visible — Monge
  - Invisible — User guarantee

\(O(\log k)\) time

\(O(k)\) Reveal + Hide ops ⇒ \(O(k \log k)\) time

(Recall SAWK only needs \(O(k)\) time, but completely offline.)
**FR-Dijkstra**  Given planar graph $G$:

**Preprocessing:** to be determined

- Build nice $r$-division — $O(n)$ time
- Build dense distance graph via M5SP — $O(n \log r)$ time
- Recall bdy-to-bdy distance arrays are not Monge but do de-compose into Monge arrays

Prep each Monge subarray for range-min queries — $O(n)$ time

Total prep time = $O(n \log r)$

**Query time:** Given boundary vertex $s$, compute distance to every other bdy vertex

Naive: Dijkstra in PDG takes $O(V \log V + E)$

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

We're trying to avoid this

Run Dijkstra as usual, but

- Keep a global heap of all Monge heap minimal $\text{dist}(v)$ for all $v$ beyond current wavefront
- $u \leftarrow \text{Extract Min}$ next closest vertex to $s$

$\Rightarrow \text{Reveal} (u, \text{dist}(u))$ and $\text{Hide} (u)$ in every Monge heap containing $u$
If \( G \) has bounded degree: \( v \) is on bdy of \( O(1) \) pieces

\[ \mathcal{H}_B \]

So \( v \) is in \( O(\log r) \) Monge heaps

Aggregate Monge heaps in each piece

Piece heaps in global heap

Each Hide: \( O(\log r) \) piece heap ops \( \rightarrow \) \( O(\log^2 r) \) time

\( + O(1) \) global heap ops \( \rightarrow \) \( O(\log r) \) time

Overall time = \( O\left(\frac{r^2}{\log^2 r + \log n}\right)\)

This term can be removed with more effort (like \( O(n) \)-time shortest paths)

Okay, Finally: Minimum cut

We've already reduced to the following problem

Given plane graph \( G \) with vertices

\[ s_1, s_2, \ldots, s_k, t_1, t_2, \ldots, t_t \]

in cyclic order on outer face

Compute \( \text{dist}(s_i, t_i) \) for all \( i \)

1. Build \( r \)-division and DD, prep for \( \text{Fedor Dijkstra} \) \( O(n \log r) \)

2. Follow \( \text{Fedor Dijkstra} \) to build \( k / \log n \) paths \( s_i \rightarrow t_i \)

But use \( \text{Fedor Dijkstra} \) : \( O\left(\frac{n^2 \log^3 n}{\log r}\right) \)

3. within each stripe of width \( O(\log n) \)

just run \( \text{Fedor Dijkstra} \) \( O(n \log \log n) \)
**Intuition:** Phase 2 running time is

\[ T(n,k) = O\left(\frac{m}{\log^2 n} + \frac{1}{k^2} \right) \]

where \( n_1 + n_2 = n \)

\[ = O\left(\frac{n}{\log^2 n} \log k \right) \]

So total time is

\[ O\left(n \log r + \frac{1}{\sqrt{r}} \log^3 n + n \log \log n \right) \]

Set \( r = \log^5 n \Rightarrow O(n \log \log n) \)

**Technical problem:** Time for FR-Dijkstra depends on total size of pieces intersecting current slab.

"Split" DDG

**Minor modification to Merge Heaps**

If any piece reduced to a single edge, we can finish off entire slab in \( O(n^3) \) time!