Let's try this again

Monge heap:

Given a Monge array D ($k \times k$), and "given" an unknown vector c (pretend initially ∞)

Define $M[i,j] = D[i,j] + c[j]$ ← Monge (No matter what's in c)

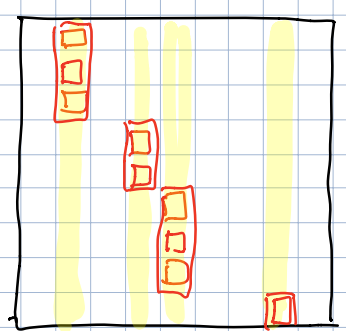
Three operations:

- Reveal(j, x) — Define $c[j] \leftarrow x$, "revealing" 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 × unhidden rows is Monge
⇒ row mins are monotone



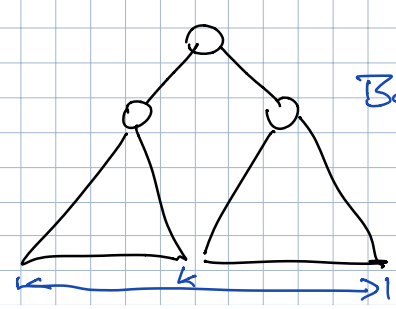
For each live column j
Maintain row interval(s)
where min element is in col. j.

Keep triples (j, i_{min}, i_{max}) in a min-heap
keyed by minimum element.
$O(\log k)$ insert delete extractmin
← $\le k$ triples in heap

- Min element in any subcolumn depends only on D!

Prep each column of D for range minimum queries

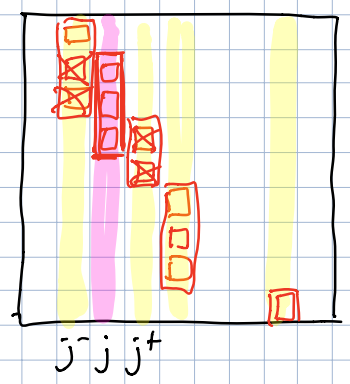
Given i_{min}, i_{max}, find $i_{min} \le i \le i_{max}$
minimizing $D[i,j]$



Balanced (static) binary tree
with k leaves

i^{th} leaf stores $D[i,j]$

internal node v 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

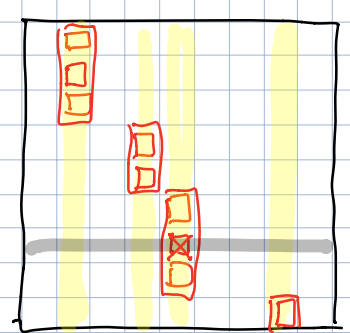
Remove triples $(j^\pm, -, -)$ that overlap (j, i_{min}, i_{max})

Add triples $(j^-, --), (j, --), (j^+, --)$

Charge time to prior insertion

Overall amortized time $O(\log k)$

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



Hide(i)

maintain i_{min}, i_{max} intervals in BST

Row i intersects at most one triple (j, i_{min}, i_{max}) in global heap

Remove that triple,
Insert $(j, i_{min}, i-1)$ and $(j, i+1, i_{max})$

We don't have to worry about other cols:

- Visible — Monge
- Invisible — User guarantee ☆

$O(\log k)$ time

$O(k)$ Reveal + Hide ops $\Rightarrow$ $O(k \log k)$ time

(Recall SMAWK only needs $O(k)$ time, but completely offline.)

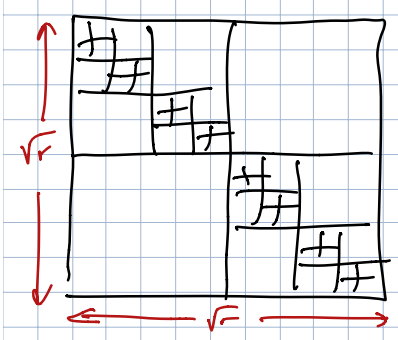
FR-Dijkstra

Given plane graph G:

Preprocessing:

- Build nice r-division — $O(n)$ time (to be determined)
- Build dense distance graph via MSSP — $O(n \log r)$ time
- Recall bdry-to-bdry distance arrays are not Monge but do decompose into Monge arrays

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



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

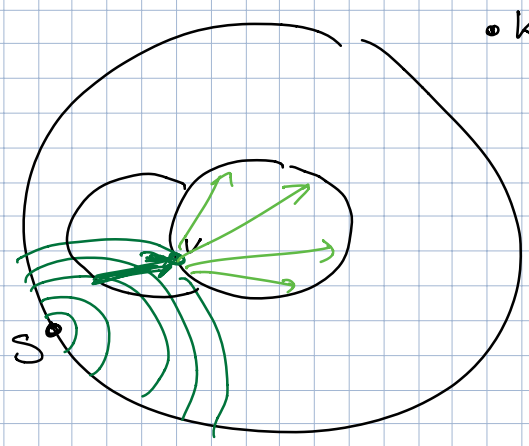
Query time: Given boundary vertex s (vertex of DDG), compute distance to every other bdry vertex

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

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

We're trying to avoid this

Run Dijkstra as usual, but



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

$\Rightarrow$ Reveal$(v, \text{dist}(v))$ and Hide(v) in every Monge heap containing v

If G has bounded degree: v is on bdry of $O(1)$ pieces

HW3!

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

Aggregate Monge heaps in each piece
piece heaps in global heap

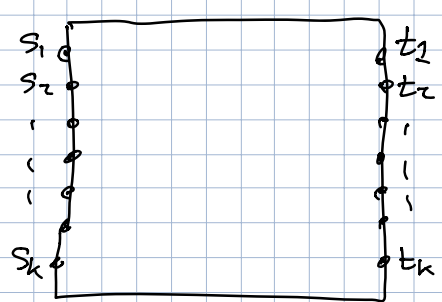
Each Hide: $O(\log r)$ piece heap ops $\longrightarrow O(\log^2 r)$ time
$+ O(1)$ global heap ops $\longrightarrow O(\log n)$ time

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

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

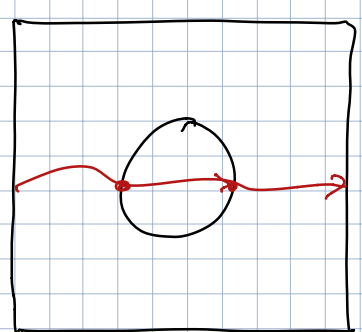
Okay, Finally: Minimum cut

We've already reduced to the following problem



Given plane graph G with vertices $s_1 s_2 \ldots s_k t_k \ldots t_2 t_1$ in cyclic order on outer face ← maybe more nodes on outer face

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



① Build r-division and DD G prep for FR-Dijkstra $O(n \log r)$

② Follow Reif's algorithm to build $k/\log n$ paths $s_i \to t_i$

But use FR-Dijkstra: $O\left(\frac{n}{\sqrt{r}} \log^2 n\right)$

③ within each stripe of width $O(\log n)$ just run Reif's algo $O(n \log\log n)$

Intuition: Phase 2 running time is

$$T(n,k) = O\left(\frac{n}{\sqrt{r}}\log^2 n\right) + T(n_1, k/2) + T(n_2, k/2)$$

where $n_1 + n_2 = n$

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

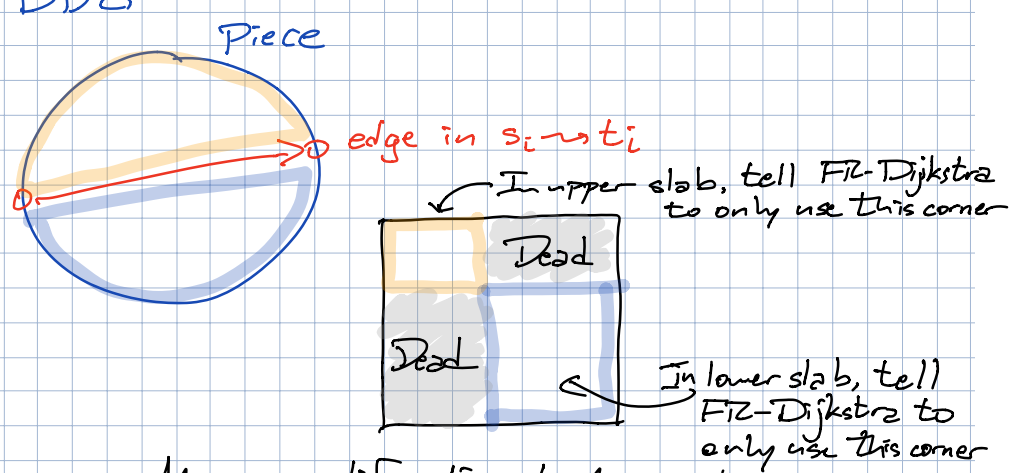
So total time is

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

Set $r = \log^6 n \implies O(n\log\log n)$

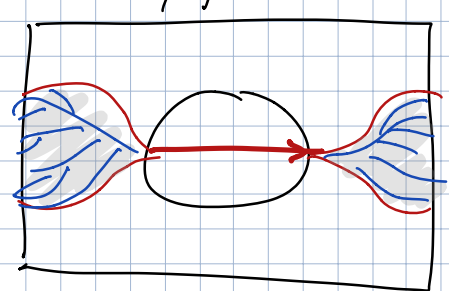
Technical problem: Time for FIZ-Dijkstra depends on total size of pieces intersecting current slab

"Split" DDG



Minor modification to Monge Heaps

If any piece reduced to a single edge,



we can finish off entire slab in $O(n)$ time!