Dense-distance graphs and FR-Dijkstra

11/4/20 10:58 AM

Where were we? Ah right, chopping plane graph into pieces.

Dense-distance graphs.

Recall a good r-division: n-vertex plane graph G.

- Chop G into $O(n/r)$ pieces
- each piece has size $\le r$.
- # bdry vertices per piece is $O(\sqrt{r})$
- # holes per piece is $O(1)$

Imagine we use MSSP to compute APSP between bdry vertices for each piece.

Now, replace each piece w/ a complete graph on bdry vertices.

- Obtaining dists: $\left[O(r\log r) + O(\sqrt{r})^2 \cdot O(\log n) \right] \cdot O(\frac{n}{r}) = O(n\log n)$
- # vertices in the new graph (dense-dist. graph):
  edges
  $O(\frac{n}{r}) \cdot O(\sqrt{r})^2 = O(n/\sqrt{r})$ $O(n)$

example. Shortest path on planar graphs w/ ≥ 0 edge-weights

1. Declare s & t as "bdry vertices".
2. Build r-division. (Choose r later) $O(n)$
3. Build DDG. $O(n\log r)$
4. Dijkstra from s. $O(V\log V + E) = O(\frac{n}{\sqrt{r}}\log n + n)$

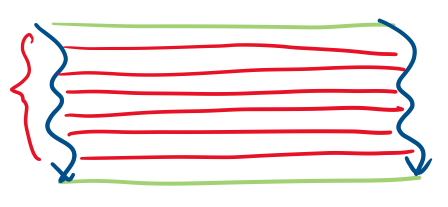
If we set $r = \log^2 n$, $O(n\log\log n)$ in total!

Okay, but what do we need?

$O(n\log\log n)$ preprocessing.

$O(n)$ $\approx\sqrt{\pi}$ parallel shortest paths.

say $\pi/\text{polylog}\,\pi$



$$T(n,\pi) \le O(n\log\log n) + \sum_{i=1}^{\pi/\text{polylog}\,\pi} T(n_i, \text{polylog}\,\pi)$$

$\Rightarrow O(n\log\log n)$.

Can't afford $O(\frac{n}{\sqrt{r}}\log n + n) \cdot O(\log\frac{\pi}{\text{polylog}\,\pi})$.

Main Question: How do we perform Dijkstra w/o checking edges?

FR-Dijkstra. [Fakcharoenphol–Rao '01]

What do we need for Dijkstra: ¡algorithms!

- FindMin: Return min-element from a heap.

What are the elements?

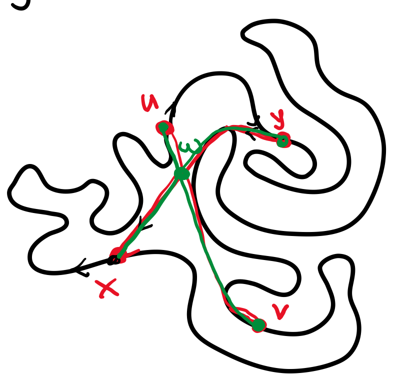
The edges in DDG of a piece.

We can't build the whole heap!

$\Rightarrow$ Explore properties of the distance matrix.

Monge property. [Monge 1781] (Another close friend of JCT).

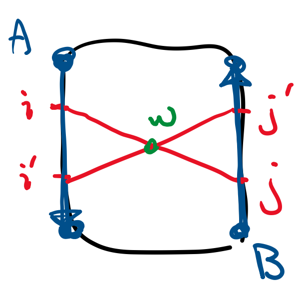
$$d(u,v) + d(x,y) \ge d(u,x) + d(v,y)$$



Monge matrix.

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

Row labels: A (row 1), i (row 3), i' (row 4); column labels: j' (column 2), j (column 3); B.



$$M(i,j) + M(i',j') \ge M(i',j) + M(i,j') \quad \forall i \le i', j \le j'$$

$$M(i',j') - M(i,j') \ge M(i',j) - M(i,j)$$

- Row-differences are growing monotonically.

SMAWK algorithm. [AKMSW'87] [KK'90]

Row-wise minimums can be found in $O(n)$ ($\alpha(n)$) time for n×n (partial) Monge matrix.

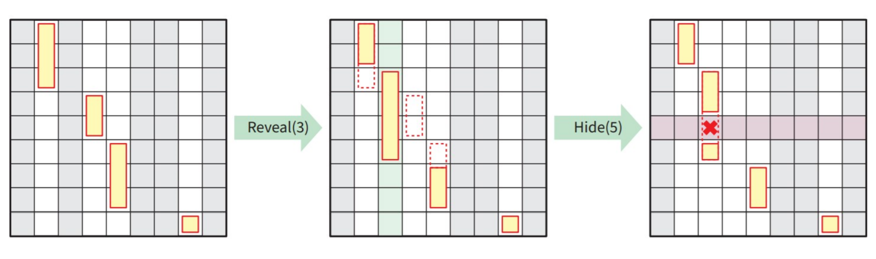
Monge Heap.

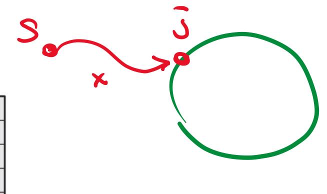
Let's say we have a (k×k) Monge matrix M.

Some entries are hidden, and some visable.

Monge heap supports: ¡data structure!

- FindMin: smallest (w.r.t. row) element visable (candidates) in M.
- Reveal(j, x) reveal j-th column, w/ additive x
- Hide(i): hide i-th row (reached).





At most k reveals & k hides. $O(k\log k)$

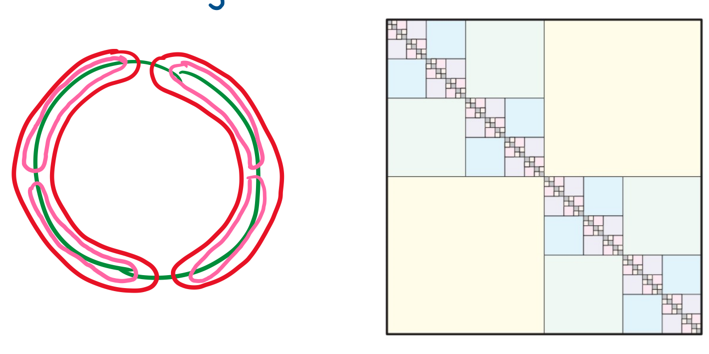
(Priority queue + binary tree + range-min queries.)

Decompose a piece into Monge matrices.

Let D be the distance matrix of a piece. (w/ $\sqrt{r}$ bdry vertices)

D can be decomposed into $O(\log r)$ levels of Monge matrices:

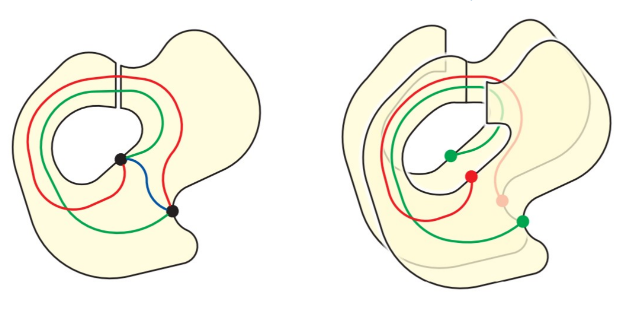
1. For a single hole, divide-&-conquer.



Monge matrices:
$\#M(k) = 2 + 2\cdot\#M(\frac{k}{2}) \le 3k-2$

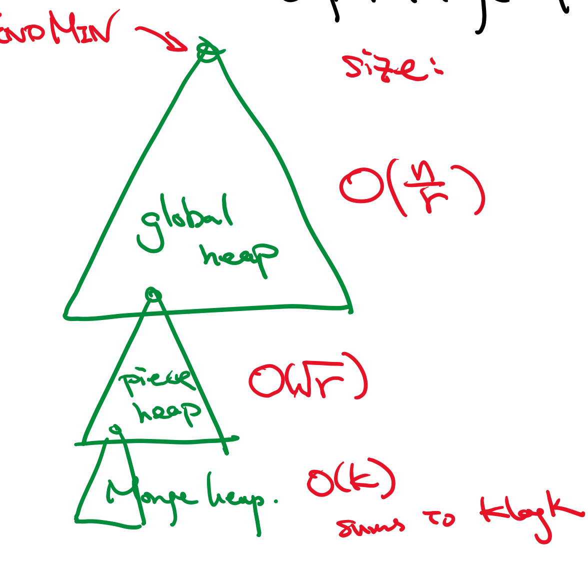
Total Perimeters:
$TP(k) = 2\cdot k/2 + 2\cdot TP(\frac{k}{2}) \le O(k\log k)$

2. For multiple hole, cyclic covering.



$O(1)^2$ pairs of holes:
$\Rightarrow$ min of $O(1)$ Monge matrices.

Maintaining a piece heap of min-elements in all Monge heaps, one per Monge matrix.



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, dist(v))
  |   | Hide(v)

Return dist(t)
```

Analysis of FR-Dijkstra:

- Each Monge heap takes $O(k\log k)$
- All Monge heaps take $O(\frac{n}{r}) \cdot O(\underbrace{\sqrt{r}\log r}_{\text{total perimeter}} \cdot \underbrace{\log k}_{\text{update time}}) = O(\frac{n}{\sqrt{r}}\log^2 r)$
- All piece heaps take $\underbrace{O(\frac{n}{\sqrt{r}})}_{\text{\# bdry vertices}} \cdot \underbrace{O(\log r)}_{\text{\# ops per bdry vertex}} \cdot \underbrace{O(\log r)}_{\text{update time}}$ 〃
- Global heap takes $\underbrace{O(\frac{n}{\sqrt{r}})}_{\text{\# ops (1 per bdry vertex)}} \cdot \underbrace{O(\log n)}_{\text{update time}}$ (assuming $O(1)$-deg)

$O(\frac{n}{\sqrt{r}}\log^2 n)$ in total.

$\underbrace{O(n)}_{\text{r-division}} + \underbrace{O(n\log r)}_{\text{DDG + MSSP.}} + \underbrace{O(\frac{n}{\sqrt{r}}\log^2 n \cdot \log\pi)}_{\text{FR-Dijkstra + d\&c}} + \underbrace{O(\frac{n}{r}\cdot r\log r)}_{\text{Ref.}}$

Set $r = \log^6 n$. $\Rightarrow O(n\log\log n)$ min cut.

Technical Details...

- r-division needs to respect strips.
- degenerate strips.
- Actually requires shortest paths to cut into strips.
- $O(1)$-deg assumption.