Minimum Cuts in Plane Graphs

Whitney duality: minimal cut \leftrightarrow simple cycle
separating s, t \leftrightarrow separating s* and t*
min. cut \leftrightarrow min. wt.

So minimum (s,t)-cut in G is dual to
minimum generating cycle in annulus G* \setminus G(s,t*)

\Rightarrow homotopic to boundary
\Rightarrow winding \neq 1
\Rightarrow reduced crossing seq. length \neq 1

Let \pi = shortest path from s* to t*

Shortest gen. cycle A crosses \pi exactly once
\Rightarrow A \cap \pi is an interval

Consider crossing sequence

\begin{align*}
\text{crossing} & \Rightarrow + \Rightarrow - \\
\text{not crossing} & \Rightarrow - \Rightarrow +
\end{align*}

Boundary has crossing sequence +
So any cycle hom. to body
has reduced x-seq +

So |X| \geq 1, and
If |X| > 1, must contain + - or - +, so we can shortcut
Cut open along $\tau$: Duplicate vertex edges, split incident edges

Now looking for shortest path between matching vertices on $\tau'$ and $\tau''$

Suppose $\tau$ has $k$ edges.

Naive: $k \times$ Dijkstra $\Rightarrow O(k \log n)$ time

Fast SP $\Rightarrow O(kn)$

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

[Theif] Divide and conquer $\Rightarrow O(n \log k)$

Split along median path from $x_i$ to $x_i'$

Recurse above and below

$T(n, k) = O(n \log n) + T(n_1, k/2) + T(n_2, k/2)$

$n_1 + n_2 = n$

$= O(n \log n \log k)$

FTR-Dijkstra $\Rightarrow$ Shortest paths in dense distance graphs

Recall def. DDGs:

- Build nice r-division: $\Theta(r)$ pieces, each with $O(r)$ verts
  - $O(r)$ bdry verts
  - dish with $O(1)$ holes

- Replace each piece with clique over bdry verts
  - Distances via MSSP in $O(r \log r)$ per piece
  - $= O(n \log r)$ total

- $O(n/r^2)$ vertices
  - $O(n)$ edges

$\Rightarrow$ Dijkstra runs in $O(n \log^2 r)$ time

We want to lose the $n_+ \log n$ term — Don’t look at every edge

FTR exploit Merge structure!
For now, assume pieces are disks.

Let $P$ be any piece. When Dijkstra's wave front hits $P$, vertex $x$, relax all edges $x \rightarrow z$ in $P$.

Then later when wave hits $y$, relax all edges $y \rightarrow z$.

Define $M[\ell, n] = \text{dist}(\ell) + w(\ell \rightarrow n)$.

We discover $\text{dist}(\ell)$ in increasing order.

$\rightarrow$ Columns of $M$ "revealed" one at a time.

Ultimately we need $\min_{\ell} (\text{dist}(\ell) + w(\ell \rightarrow n))$ for all $n$.

$\rightarrow$ Need minima of every row of $M$.

Merge heap: Maintain row minima of an $n \times n$ merge array as columns revealed.

Trick: Row minima are monotone.

Suppose columns $j_1 < j_2 < \ldots < j_e$ revealed so far.

Each column contains minima in an interval of rows $L[j_i] \ldots U[j_i]$.

If unseen column $j_i \leq j < j_{i+1}$

$L[j_i] \leq L[j] \leq L[j_{i+1}]$

$U[j_i] \leq U[j] \leq U[j_{i+1}]$

So when new column is revealed, binary search for $L[j]$ and $U[j]$ in $O(\log n)$ time.

Distance arrays decompose into $O(r)$ merge arrays.

Each vertex appears in $O(\log n)$ merge arrays.
Back to FR-Dijkstra:
Keep baby verts in global priority queue
Pull min vertex, relax outgoing edges
< reveal columns in $O(\log r)$ merge arrays for each incident piece

Keep a global heap of all $O(\sqrt{r})$ vertices
Actually a heap of $O(\sqrt{r})$ merge arrays, each maintaining its own minimum row
Each vertex in $O(\log r)$ arrays, so ExtractMin takes $O(\log n \log r)$ time.

Within each piece:

Total time for first columns: $T_F(h) = O(h) + 2T_F(\sqrt{r})$

$\Rightarrow O(\sqrt{r} \log r)$

Total time for all other columns: $T_D(h) = O(h \log h) + 2T_D(\sqrt{r})$

$\Rightarrow O(\sqrt{r} \log^2 r)$

Each relaxation could change $O(\sqrt{r})$ distances.
Find \( \log n \) paths \( T(n) \).

\[
T(n) = O(\frac{c}{\sqrt{n}} \log^2 n) + 2T(n)
\]

\[
= O(\frac{c}{\sqrt{n}} \log^2 n) + O(\log n)
\]

The curve in both sides:

\[
T(n) = O(n) + O(\log n) - T(\log\log n) = O(\log\log n)
\]