Algorithmic tools on planar graphs

11/2/20 9:26 AM

Administrivia.

- Project proposals are out. I am proud of the set of diverse topics we have!
- Project presentations: 11/30 – 12/4
- Project writeup: no later than 12/7.

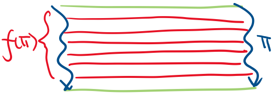
☺

Motivation: divide-&-conquer approach for problems on planar graphs. ¡algorithms!

Recall speeding up min-cut:

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

$O(n)$ $\frac{\pi}{\log \pi}$ parallel shortest paths.



Tools to get there: [INSW'11]

Today

- [Frederickson'89] [KMS'12] (Cycle) separator decomposition / r-division. $O(n)$
- [Klein'05] [CCE'07] Multiple-source shortest paths $O(r \log r)$ per piece.
- [Fakcharoenphol-Rao'01] Monge heap / dense distance graph. $O(\frac{n}{r} \cdot r \log r)$
- [AKMSW'87] FR-Dijkstra $O(\log n \cdot \frac{n}{\sqrt{r}} \log n \log r)$
- [Reif'83] Reif. $O(\frac{n}{r} \cdot r \log n)$

Wed.

Multi-source shortest paths.

Input: A directed planar graph G w/ × edge weights embedding (a piece)

Set of boundary vertices those incident to exterior face.

Output : Implicit representation of shortest paths between

o.w. $\Omega(n^2)$ every boundary vertex s & every vertex t

shortest path trees at every bdry vertex.

[K'05]

Thm. MSSP problem can be solved in $O(n \log n)$ time, s.t. each shortest path distance can be reported in $O(\log n)$ time.

Cor. If the piece has $O(\sqrt{n})$ boundary vertices, all boundary-to-boundary shortest path dists. can be computed in $O(n \log n)$ time.

Main Idea : Analyze how shortest path tree changes as one moves the root around ∂G.

Disk-tree Lemma.

Take any spanning tree T of G, an any edge $e \in T$.

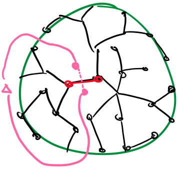
Let A, B be the two trees of $T - e$.

Then, boundary vertices in A (resp. B) are consecutive.

Pf. $C := (E \setminus T)^*$ cotree. $C + e^*$ has a cycle.

By JCT, $T - e \,/\, \partial G$ has exactly 2 parts. ▨

Notice : T might not be a shortest tree!



Cor. Let $S_0 \ldots S_{k-1}$ be bdry vertices of G.

Shortest path trees $T_0 \ldots T_{k-1}$, each rooted at S_i.

edges pointing out from s_i

Then, each directed edge $x \to y$ belongs to a consecutive interval of shortest path trees: $T_i, T_{i+1}, \ldots, T_{i+j}$

Pf. Let T be the unique shortest path tree at y, where all edges point towards y.

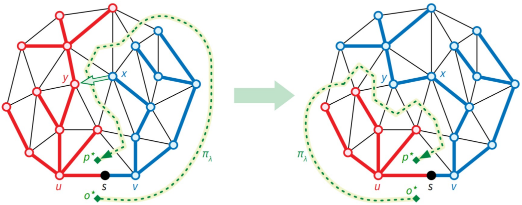
Apply disk-tree lemma on $T - xy$. ▨

Intuition : each edge only shows up / disappear from T once as we move the root around ∂G.

Remark. We made some assumptions on the uniqueness of shortest path trees.

Isolation lemma! cotree perturbation.

Parametric shortest paths.



Consider edge uv on ∂G. As one moves the root from u to v, we imagine the shortest path tree rooted as s changes at discrete steps. "pivot".

For parameter $\lambda \in [0,1]$. T_λ shortest path tree:

$$w_\lambda(us) = \lambda \cdot w(uv), \qquad w_\lambda(sv) = (1-\lambda) \cdot w(uv)$$

Define $d_\lambda(x)$ to be dist from s to x wrt. w_λ.

$$\text{slack}_\lambda(x \to y) := d_\lambda(x) + w(x \to y) - d_\lambda(y)$$

[Bellman-Ford '53]. [Lester]

Observation. Any shortest path tree has ≥ 0 slack on all darts.
$= 0$ " tree darts.
> 0 " non-tree darts.

Pf. Otherwise one can "relax" $x \to y$ if $\text{slack}_\lambda(x \to y) < 0$:

$$d_\lambda(y) \leftarrow d_\lambda(x) + w(x \to y). \quad \text{pred}(y) \leftarrow x. \quad ∎$$

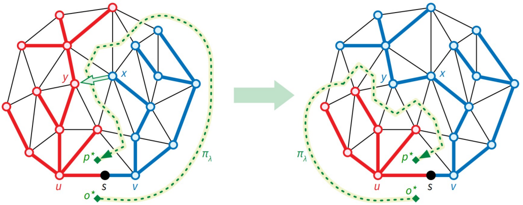
A vertex is red / blue if $d_\lambda(x) \uparrow$ / $\downarrow$ as $\lambda \uparrow$

Dart $x \to y$ active if $\text{slack}_\lambda(x \to y) \downarrow$ as $\lambda \uparrow$.

Red-Blue Lemma. $\forall \lambda \in [0,1]$, we have the invariant

- If $s \to v \notin T_\lambda$, all vertices are red, only $s \to v$ active.
- If $s \to u \notin T_\lambda$, " blue, no active darts.
- Else, all vertices "behind" under u are red
  " "in front" " v are blue.

$x \to y$ active if x blue & y red.



Look at the shortest $\overset{\pi_\lambda}{\text{path}}$ from o^* to p^* in $\overset{\text{CoTree}}{C_\lambda^*} := (E \setminus T_\lambda)^*$.

Lemma. The dual of active darts is the path π_λ.

Cor. The min-slack active dart on π_λ is the next pivot.

Intuition. Now we can execute pivoting using dynamic tree DS.

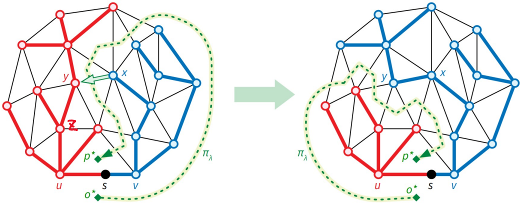
Cor. MSSP performs $O(n)$ pivots.

Pf. By disk-tree lemma. ▨

MSSP algorithm. Assume scenario that both $u \to s$, $v \to s$ in T_λ.

```
NextPivot(G, T_λ):
    $x \to y \leftarrow$ MinPathSlack($o^*$, $p^*$)
    $\Delta \leftarrow \text{slack}_\lambda(x \to y)/2$

    if $\lambda + \Delta / w(u \to v) < 1$:
        Pivot($x \to y$, $\Delta$)
        return $\lambda + \Delta / w(u \to v)$
    else
        return 1.
```

```
Pivot(x→y, Δ):
    AddSubtreeDist($\Delta$, $u$)
    AddSubtreeDist($-\Delta$, $v$)

    AddPathSlack($-2\Delta$, $o^*$, $p^*$)

    $z \leftarrow$ pred(y), pred(y) $\leftarrow x$

    Cut(yz), Link(xy)
    Cut($(xy)^*$), Link($(z \to y)^*$).
```



Details:

Implement primal & dual spanning trees using dynamic tree DS like link-cut tree, supporting

AddSubtreeDist, Link, Cut, MinPathSlack, AddPathSlack.

all in $O(\log n)$ amortized time i data structures!

⇒ splay trees [Sleator-Tarjan '85]

Summary. $O(n)$ pivots, each in $O(\log n)$ amortized time.

⇒ $O(n \log n)$ time in total.

* * *

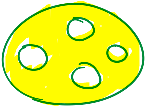
r-division / separator decomposition.

Goal. Decompose plane graph G into roughly equal-size pieces.

- each piece has size r.
- #pieces is $O(\frac{n}{r})$.
- #boundary vertices per piece is $O(\sqrt{r})$.

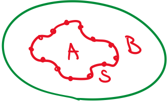
We also want:

- $O(1)$ "#holes"



Separator. Vertex subset S w/ $G[V-S] = A \cup B$. s.t.

- $|S| \le O(\sqrt{n})$.
- $|A|, |B| \le \overset{2/3}{C} \cdot n \quad C < 1$ can be weighted!

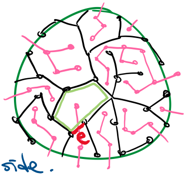


Cycle separator. Vertices in S form a simple cycle. Assume G triangulation.

[Miller'86, KMS'12, HP'18]

Thm. Cycle separator exists and can be found in $O(n)$ time.

Fundamental-cycle separator.



cycle(T, e) := unique cycle in $T+e$

dual Tree C is a binary tree.

Walk down from root of C towards larger side.

find first edge vw w/ $\le \frac{2n}{3}$ vertices in subtree of w.

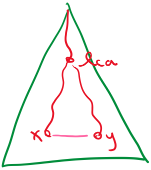
$\Rightarrow \ge \frac{n}{3}$ ..

cycle$(T, (vw)^*)$ is a balanced cycle separator.

Downside: $S :=$ cycle$(T, (vw)^*)$ can be big!

Idea. Balance faces instead. Euler formula.

$\Rightarrow$ nested cycles.



Let cycle(T_0, xy) be balanced separator

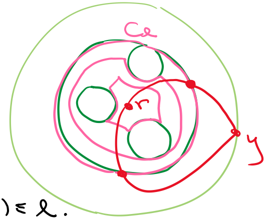
Reroot the BFS tree to lca$(x, y) \Rightarrow T$.

cycle(T, xy) = cycle(T_0, xy);

assume cycle(T, xy) be exterior.

Define $\ell(v)$ based on BFS.

assume $\ell(x) \le \ell(y)$.



Define $\ell(f) := \max_{v \in f} \ell(v)$. #levels $\ge \sqrt{n}$.

Consider the region containing $\boxed{\text{faces}}$ of $\ell(f) \le \ell$.

$R_{\le \ell}$

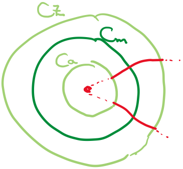
Lemma.

- the "boundary" of $R_{\leq \ell}$ is a simple cycle, C_ℓ
- all C_ℓ's are pairwise vertex-disjoint.
- cycle(T.xy) intersects each C_ℓ at most twice.

Now, let C_m be outermost cycle w/ $\leq \#F/2$ faces

Find C_a, C_z s.t.

- $m-\sqrt{n} \leq a < m < z \leq m+\sqrt{n}$, &
- $|C_a|, |C_z| \leq O(\sqrt{n})$.



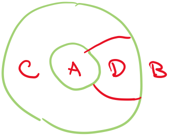
Claim. $|A|, |B| \leq \#F/2$, $|C|, |D| \leq 2\#F/3$

Pf. If $|A| > \#F/2$, C_m too big $-\!\!\times\!\!-$

If $|B| > \#F/2$, C_m too small $-\!\!\times\!\!-$.

If $|C| > 2\#F/3$, cycle(T.xy) not balanced $-\!\!\times\!\!-$
or
$|D|$

∎



At least one of A,B,C,D has size $\geq \#F/4$.

$\Rightarrow$ Boundary cycle of one of ABCD gives balanced cycle separator. ∎

◎

To get r-division.

- Iterate between separating vertices / bdry vertices / holes.
- Efficient implementation to reuse tree-cotree across.