Ulpana - Tirgul 2

Thorup-Zwick Distance Oracles

Lecture: rand. construction that given graph G and param k outputs a data structure (Oracle)

- for query $\langle u,v\rangle \in V \times V$ get dist. estimate $\hat{\delta}(u,v)$ s.t

$$d_G(u,v) \leq \hat{\delta}(u,v) \leq (2k-1)\cdot d_G(u,v)$$

- query take $O(k)$ time
- Oracle takes up $\tilde{O}(k\, n^{1+\frac{1}{k}})$ space

This lesson:
- report also a path of length $\hat{\delta}(u,v)$
- find the "hidden spanner" in the oracle
- learn about "tree covers"

Reminders

centers $\quad V = A_0 \supseteq A_1 \supseteq \cdots A_{k-1} \supseteq A_k = \emptyset$

$\quad A_i = \text{Sample}(A_{i-1}, n^{-1/k})$ for $i = 1, \dots, k-1$

pivots $\quad p_i(v)$ - closest i-center to v

bunches $\quad B(v) = \bigcup_{i=0}^{k-1} \left\{ w \in A_i \setminus A_{i+1} \mid d(w,v) < d(A_{i+1}, v) \right\}$

high level of $query(u,v)$

- finds some "good" w and i s.t.
  - (i) $p_i(u) = w$
  - (ii) $v \in B(v)$
- returns $\hat{\delta}(u,v) = d(u,w) + d(w,v)$
- the specifics of the query alg. that chooses v, i ensures $(2i+1)$ approx.

Choice of pivots

What happens if there are many vertices from A_i that are tied for being closest to v?

In the lecture, we just chose arbitrarily one of them to be $p_i(v)$.

Now, we will choose them consistently along the levels:

$$d(A_i, v) = d(A_{i+1}, v) \Rightarrow p_i(v) = p_{i+1}(v)$$

Claim: under this choice, $p_i(v) \in B(v)$ for all $0 \le i \le k-1$.

Proof: backwards induction. Base: $p_{k-1}(v) \in A_{k-1} \subseteq B(v)$.

Step: if $d(A_i, v) = d(A_{i+1}, v) \Rightarrow p_i(v) = p_{i+1}(v) \in B(v)$

else $d(p_i(v), v) = d(A_i, v) < d(A_{i+1}, v)$ and $p_i(v) \in A_i \setminus A_{i+1}$ $\Rightarrow p_i(v) \in B(v)$ ∎

Clusters "inverses of bunches"

if $w \in A_i \setminus A_{i+1}$ then $C(w) = \{v \in V \mid d(w,v) < d(A_{i+1}, v)\}$
$= \{v \in V \mid w \in B(v)\}$

Lemma: suppose $v \in C(w)$ for $w \in A_i \setminus A_{i+1}$
suppose x is on shortest path between v, w
then $x \in C(w)$

Proof:

$$d(w,x) + d(x,v) \overset{\text{x on shortest path bet. } w,v}{=} d(w,v) \overset{v \in C(w)}{<} d(A_{i+1}, v) \overset{\text{triangle ineq.}}{\leq} d(A_{i+1}, x) + d(x,v)$$

subtract $d(x,v)$ from both sides:

$$d(w,x) < d(A_{i+1}, x) \qquad \blacksquare$$

Tree Cover

By the previous lemma, for every w we can build a tree $T(w)$ on the vertices of $C(w)$, such that
- w is the root
- the tree path from the root w to $v \in C(w)$ is a shortest path in G between w, v

Thm: The collection $\{T(w)\}_{w \in V}$ is a "tree cover" with the following properties (whp):
1. Each v belongs to $O(k n^{1/k} \log n)$ trees
2. For every $u, v \in V$ there is some $T(w)$ s.t. $d_{T(w)}(u,v) \leq (2k-1)\, d_G(u,v)$

Proof: 1 $v \in T(w) \iff v \in C(w) \iff w \in B(v)$

saw in lecture that (whp) $|B(v)| = O(kn^{1/k} \log n)$
(hitting set argument)

2 Let w and i be those found by $query(u,v)$

by the claim on choice of pivots

(i) $w = p_i(u) \in B(u) \Rightarrow u \in C(w)$

(ii) $w \in B(v) \Rightarrow v \in C(w)$

Thus, $d_{T(w)}(u,v) \le d_{T(w)}(u,w) + d_{T(w)}(w,v)$

by the lemma on $T(w)$ $= d_G(u,w) + d_G(w,v)$

by query alg. $= \hat{\delta}(u,v)$

by analysis of query alg from lecture $\le (2k-1)\, d_G(u,v)$

Corollary: $\bigcup_{w \in V} T(w)$ is a $(2k-1)$-spanner of size $O(kn^{1+1/k} \log n)$

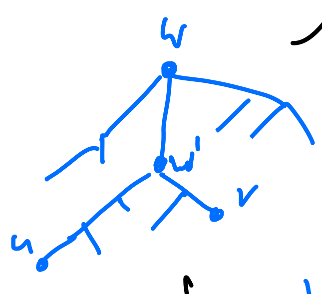
Proof: The spanner property is immediate from 2.
The size follows from 1: each edge in the union connects some v to its parent in some tree that contains v, and there are $O(kn^{1/k} \log n)$ such trees

Reporting Paths

- Store the trees $T(w)$ - $\tilde{O}(kn^{1+\frac{1}{k}})$ space
- recall that each v stores distances to $B(v)$ in hash-table
  can augment it with parent pointers:
  given $w \in B(v)$, can find the parent of v in $T(w)$
  in constant time

$path(u,v)$: return a path p weight $\leq \hat{\delta}(u,v)$

How? - let w be the vertex found by $query(u,v)$
- we know that $u,v \in T(w)$ and $\hat{\delta}(u,v) = d_{T(w)}(u,w) + d_{T(w)}(w,v)$
- Move on the tree towards the root w
  in parallel from u and v
- Stop when one walk reaches node w' already
  seen in the other walk



  w' - the lowest common ancestor
- Output the edges of the walk from u to w',
  and then, in reverse order, the edges of the
  walk from v to w'

Routing on Trees

Think of graph (here: tree T) as a communication network.

Preprocessing: assign each node $v \in V(T)$ short

- routing table $R(v)$
- destination label $L(v)$

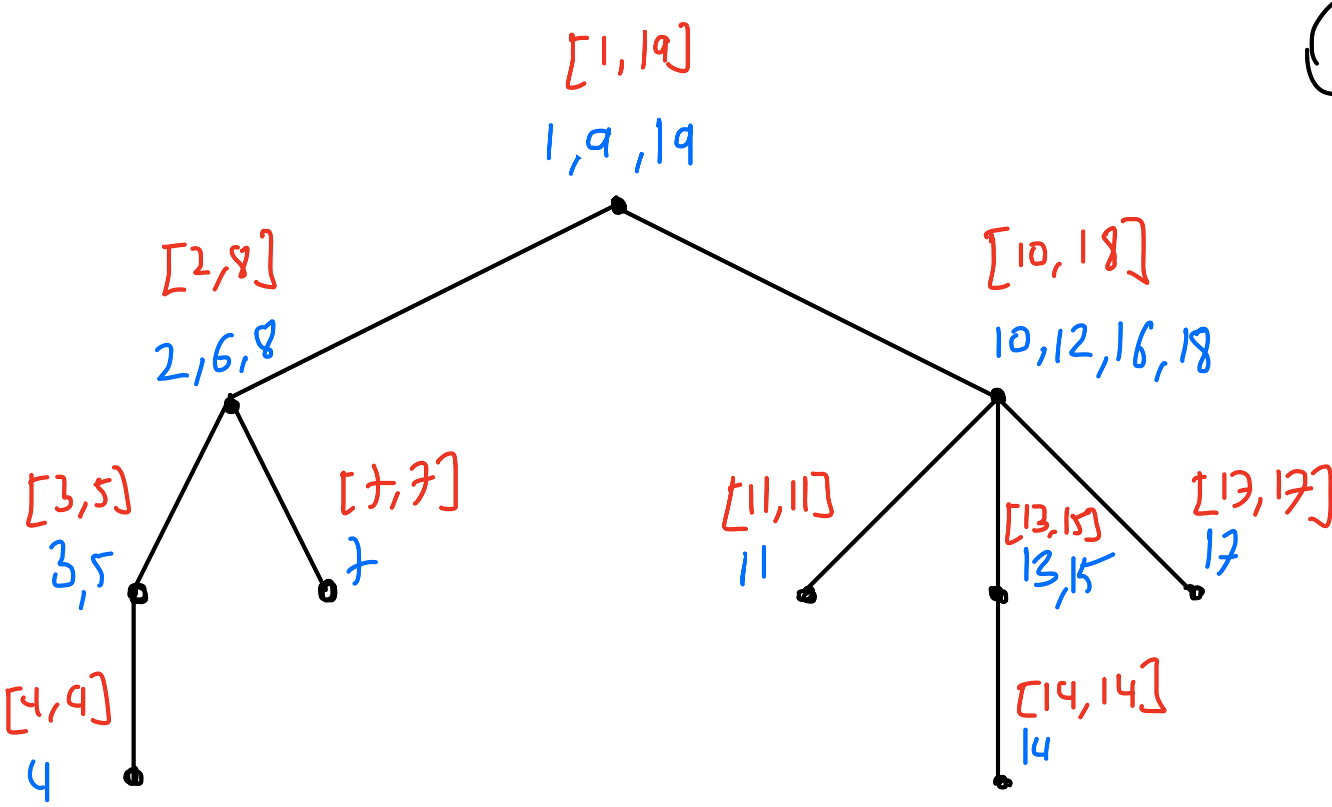
Routing phase: given $R(u)$ and $L(v)$, determine nbr of u which is the next node on the tree path to v

[if u recieves a msg with header $L(v)$, it can use its own routing table $R(u)$ to determine the next-hop for the msg to reach v]

Interval Routing

Do DFS traversal on T: each node v is associated with an interval $I(u) = [f(u), \ell(u)]$ between the first and last time the traversal visited u.

$$u \text{ ancestor of } v \iff I(u) \supseteq I(v)$$



$R(v)$: store $I(v)$ and $I(x)$ of every child x

$L(v)$: store $I(v)$

Given $R(u)$, $L(v)$:

- if there exist child x of u s.t. $I(v) \subseteq I(x)$ then next hop is x
- otherwise, next hop is parent of u

Label Size: $O(\log n)$ bits :)

Table Size: $O(\deg(v) \log n)$ bits :(

Heavy-Light Decomp.

For non-leaf v, its <u>heavy child</u> $h(v)$ is the child that has the most nodes in its subtree (break ties arbitrarily)

An edge is <u>heavy</u> if its connects between parent and heavy child, and otherwise it's <u>light</u>

<u>Obs</u>: for each node v, there are at most $\lceil \log_2 n \rceil$ light edge on the path from the root to v

Why? When you start from the root and go down to v, every time you take a light edge, you cut the #nodes in your subtree by at least $\frac{1}{2}$

Cool, simple and useful!

Heavy-light Tree Routing

$R(v)$: store $I(h(v))$ ⇝ Only $O(\log n)$ bits

$L(v)$: store $I(v)$ and all light edges on root-to-v path ⇝ $O(\log^2 n)$ bits

Given $R(u)$, $L(v)$:

- if $I(v) \subseteq I(h(v))$ next hop is $h(v)$
- if there is a light edge in $L(v)$ whose upper node is u, then next hop is the lower node of it
- else, next hop is the parent of u