March 7, 2023

# 1 Definitions

**Definition 1.** *A Voronoi diagram* is a subdivision of a $n \times m$ table equipped with a greedy braid into regions such that

1) Left boundary of each region is composed of a single braid strand and possibly a piece of the boundary;

2) Right boundary of each region is composed of several pieces $e_1, e_2, \ldots, e_k$ of braid strands, such that

   (a) the pieces $e_1, e_2, \ldots, e_k$ cross no strand from left to right,

   (b) the strand containing $e_i$ crosses the strand containing $e_{i+1}$ from left to right at the point where $e_i$ meets $e_{i+1}$, $i = 1, \ldots, k-1$.

**Lemma 1.** *The following are equivalent:*

1) *Point $p$ belongs to the Voronoi cell $f_i$ of a Voronoi site $s_i$;*

2) *Site $s_i$ is the leftmost site such that there is a path from $s_i$ to $p$ that crosses no braid strand from left to right.*

# 2 Notation

1) $\mathcal{B}$ for the greedy braid of the $\frac{n}{2} \times m$ table;

2) $\mathsf{VD}$ for the Voronoi diagram of $\mathcal{B}$

3) $\mathcal{B}^*$ for the upward $\frac{n}{2} \times m$ greedy braid;

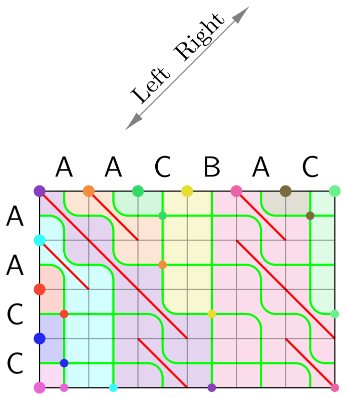
4) $\mathcal{B}^{-h}$ for the $\left(\frac{n}{2} + h\right) \times m$ greedy braid that starts $h$ rows above the middle line;

5) $\mathsf{VD}^{-h}$ for the Voronoi diagram corresponding to $\mathcal{B}^{-h}$;

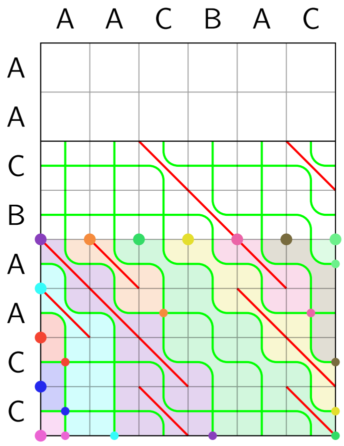
6) $s_0, \ldots, s_{m+n}$ for the sites of the Voronoi diagram;

7) $f_0, \ldots, f_{m+n}$; $f_0^{-h}, \ldots, f_{m+n}^{-h}$ for the Voronoi cells of $\mathsf{VD}$ and $\mathsf{VD}^{-h}$ correspondingly;

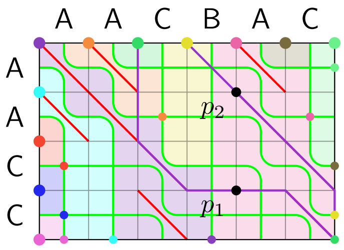
8) $c_0, \ldots, c_{m+n}$; $c_0^{-h}, \ldots, c_{m+n}^{-h}$ for the lower right corners of the Voronoi cells of $\mathsf{VD}$ and $\mathsf{VD}^{-h}$ correspondingly.




(a)




(b)




(c)

Figure 1: (a) Voronoi diagram for a given greedy braid, (b) $\mathsf{VD}^{-2}$, (c) paths from $s_6$ to $c_6^{-2}$ through $p_1$ and from $s_7$ to $c_7^{-2}$ through $p_2$

## 3 Query

**Problem 1.** Given braid $\mathcal{B}$, the lower right corners $c_0^{-h}, \ldots, c_{m+n}^{-h}$ of the Voronoi cells of $\mathsf{VD}^{-h}$, point $p$ and a number $i$, check whether $p \in f_i^{-h}$, $p$ is to the left or to the right from $f_i^{-h}$.

**Lemma 2.** *The following are equivalent:*

1) *Point $p$ belongs to the Voronoi cell $f_i^{-h}$ of $\mathsf{VD}^{-h}$;*

2) *There is a path from $s_i$ to $c_i^{-h}$ passing through $p$ that crosses no strand of $\mathcal{B}$ twice.*

The path can utilize diagonal edges, see Figure 1, (c): $p_1 \in f_6^{-2}$ (green), $p_2 \in f_7^{-2}$ (yellow).

### 3.1 Entanglement of triples oracle

**Definition 2.** We call two triples of numbers $(a_1, a_2, a_3)$, $(b_1, b_2, b_3)$ *entangled* if

$$a_1 < b_1,\ a_2 > b_2,\ a_3 < b_3 \quad \text{or} \quad a_1 > b_1,\ a_2 < b_2,\ a_3 > b_3$$

**Theorem 3.** *There exists a data structure that can store a set of $n$ triples*

$$S = \left\{ \left(a_1^1, a_2^1, a_3^1\right), \ldots, \left(a_1^n, a_2^n, a_3^n\right) \right\}$$

*and, given a query triple $q = (q_1, q_2, q_3)$, output in time $\tilde{O}(1)$ the number of triples in $S$ that are entangled with $q$.*

*Proof.* Store the set $S$ in a 3-dimensional range tree [cite!]. A $d$-dimensional range tree is a data structure that stores several points in $\mathbb{R}^d$ and effectively reports all the stored points that are inside a $d$-dimensional rectangular parallelepiped defined by its lower and higher coordinates on each axis.

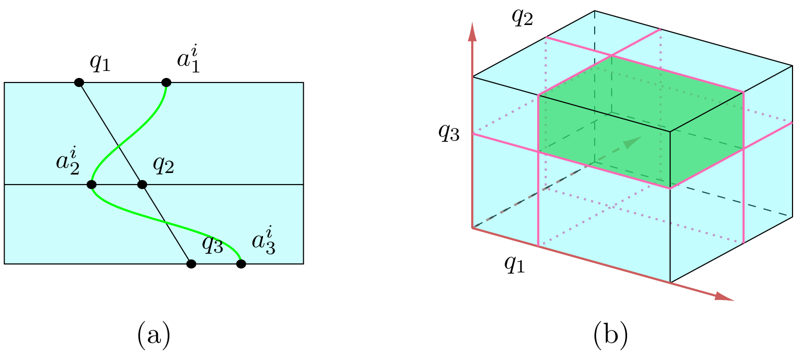



Figure 2: (a) A triple $\left(a_1^i, a_2^i, a_3^i\right)$ entangled with $(q_1, q_2, q_3)$, (b) a range query that finds this triple

An entanglement query can then be interpreted as two 3-dimensional range queries, as shown in Figure 2:

1) $a_1 < q_1,\ a_2 > q_2,\ a_3 < q_3$ is a 3-dimensional parallelepiped to the left, further and lower than the point $(q_1, q_2, q_3)$ (call this *left entanglement query*);

2) $a_1 > q_1,\ a_2 < q_2,\ a_3 > q_3$ is a 3-dimensional parallelepiped to the right, closer and higher than the point $(q_1, q_2, q_3)$ (call this *right entanglement query*).

□

**Remark 4.** The construction of a 3-dimensional range tree takes $O\left(n \log^2 n\right)$ time, and a range query takes $O(\log n)$ time.

### 3.2 Entanglement of strands oracle

**Theorem 5.** *There exists a data structure that can store a reduced embedded braid $\mathcal{B}$ and, given an arbitrary point triple $g = (0, r)$, $h = (k, s)$, $f = (\ell, t)$, decide in time $\tilde{O}(1)$ if there exists a path from $g$ to $f$ through $h$ that is not double-crossed by any strand.*

*Proof.* To build the data structure, first partition the grid of height $n$ hierarchically into $O(n)$ canonical strips located in a binary tree: each canonical strip contains two canonical strips of half its height.

For each canonical strip $(a, b)$ build an entanglement of triples oracle (Theorem 3): for each strand $s$ store the triple $(s_0, s_a, s_b)$ of its positions at the ground of the grid, the top of the canonical strip, and the bottom of the canonical strip respectively.

In total there are $O(n)$ entanglement of triples oracles, the construction of them takes $O\big(n^2 \log^2 n\big)$ time.

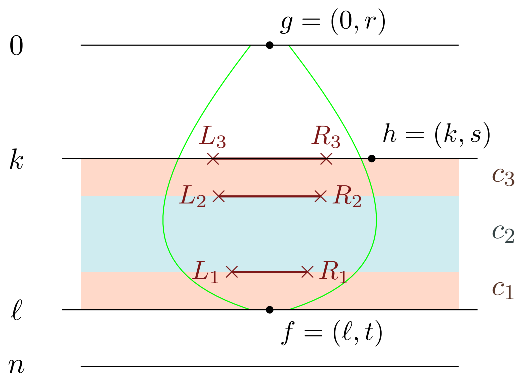



Figure 3: The decomposition of the strip $(k, \ell)$ into canonical strips; boundaries $(L_i, R_i)$ of entanglement-free regions

To answer the query, decompose the strip $(k, \ell)$ into canonical strips $c_1, c_2, \ldots, c_v$, the indexation is from the bottom of the strip upwards (see Figure 3). At each boundary of neighboring canonical strips $c_i$, $c_{i+1}$ we maintain an *entanglement-free region* $[L_i, R_i]$ which is the region where the not-double-crossed path can pass through.

It is obvious that at least one such path from $g$ to $f$ exists, therefore each entanglement-free region is nonempty. Naturally, $L_0 = R_0 = f$. Points $L_{i+1}, R_{i+1}$ are constructed from $L_i, R_i$ recursively:

1) $L_{i+1}$ is the leftmost point such that any strand that is to the left of $g$ and $L_i$ is also to the left of $L_{i+1}$,

2) $R_{i+1}$ is the rightmost point such that any strand that is to the right of $g$ and $R_i$ is also to the right of $R_{i+1}$.

Given $L_i$, we find $L_{i+1}$ using binary search in the boundary of $c_{i+1}, c_{i+2}$: if for a point $p$ the left entanglement query for $(g, p, L_i)$ returns some strands, descend to the right, otherwise descend to the left. Given $R_i$, we find $R_{i+1}$ in a symmetric way.

**Lemma 6.** *The recursive construction of $L_i, R_i$ is correct; the following are equivalent:*

*a) a point $p$ is to the left of $L_i$,*

*b) there exists a strand that is to the left of $g$ and $f$, but to the right of $p$.*

*Proof.*

- b) $\Rightarrow$ a): Assume that $p$ is to the right of (or equal to) $L_i$, see Figure 4a. The strand $\sigma$, which is to the left of $g$ and $f$, but to the right of $p$, is also to the right of $L_i$; this contradicts the construction of $L_i$.

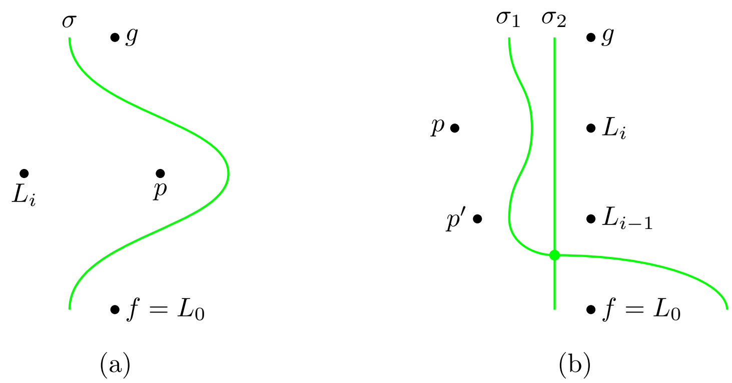



Figure 4: Proof of Lemma 6

- a) $\Rightarrow$ b): Proof by induction in $i$; assume there are no strands that are to the left of $g$ and $f$, but to the right of $p$. There is however a strand $\sigma_1$ between $L_i$ and $p$ that made us descend to the right when performing the binary search for $L_i$; $\sigma_1$ is to the left of $L_{i-1}$, see Figure 4b. The strand $\sigma_1$ must be to the right of $f$, since it is to the left of $g$ and to the right of $p$.

  Take a point $p'$ immediately to the left of $\sigma_1$ at the level of $L_{i-1}$. By the induction hypothesis, there exists a strand that is to the left of $g$ and $f$, but to the right of $p'$, call it $\sigma_2$. The strand $\sigma_2$ must cross $\sigma_1$ below $p'$, since $\sigma_2$ is to the left of $f$, and $p'$ is the immediate left neighbor of $\sigma_1$.

  Since the braid is reduced, $\sigma_2$ is to the right of $p$; this contradicts the assumption for $p$.

$\square$

Lemma 6 implies that there exists a path from $g$ to $f$ through $h$ that is not double-crossed by any strand if and only if $h$ is between $L_v$ and $R_v$. Moreover, if $h$ is outside the entanglement-free region, we can conclude whether there is a right or left entanglement of the $g$-to-$h$-to-$f$ path with a strand of $\mathcal{B}$.

**Remark 7.** The strip $(k, \ell)$ is decomposed into $O(\log n)$ canonical strips, therefore there is as much entanglement-free regions to be calculated. For each region $(L_i, R_i)$ two binary searches with $O(\log n)$ probes are done. Each probe calls a range query that takes $O(\log n)$ time. In total, the entanglement of strands query takes $O\big(\log^3 n\big)$ time.

$\square$