# Two examples of degree-driven algorithm design

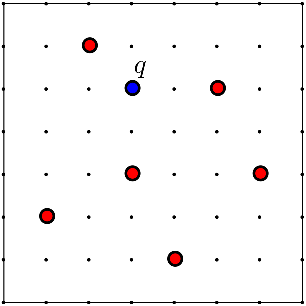
David L. Millman

December 22, 2009

Examples:

1. Reduced Precision Voronoi w/ Jack Snoeyink
2. Discrete Voronoi w/ Timothy M. Chan and Jack Snoeyink

# Goal



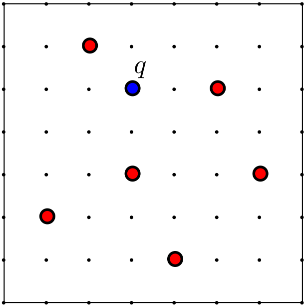

**Given**

sites $S = \{s_1, \ldots, s_n\}$ and
query points
on a $U \times U$ grid $\mathbb{U}$.

**Compute**

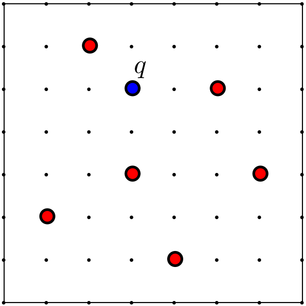
a data structure
supporting post office queries
in $O(\log n)$ time
and double precision

# Goal




**Given**
sites $S = \{s_1, \ldots, s_n\}$ and
query points
on a $U \times U$ grid $\mathbb{U}$.

**Compute**
using **less than** quadruple precision,
a data structure
supporting post office queries
in $O(\log n)$ time
and double precision




**Given**
sites $S = \{s_1, \ldots, s_n\}$ and
query points
on a $U \times U$ grid $\mathbb{U}$.

**Compute**
using **less than** quadruple precision,
a data structure
supporting post office queries
in $O(\log n)$ time
and double precision

**Results**, assuming $O(n \log n) < O(U^2)$

1. RP-Voronoi - 3x precision, $O(n \log Un)$ expected time
2. Discrete Voronoi - 2x precision, $O(U^2)$ expected time

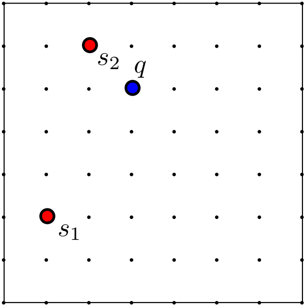
# Previous work on handling numerics in CG

Techniques for implementing geometric algorithms with finite precision computer arithmetic:

- Rely on machine precision (+epsilon)
- Exact Geometric Computation [Y97]
- Arithmetic Filters [FW93][DP99]
- Adaptive Predicates [P92][S97]
- Topological Consistency [SI92]
- Degree-driven algorithm design [LPT99]

# Analyzing Precision[LPT99]

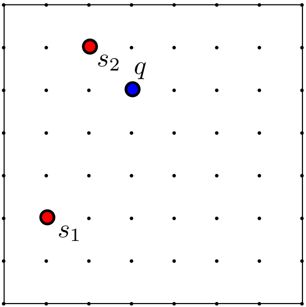
Is $q$ closer to $s_1$?




$$
\begin{aligned}
s_1, \quad s_2, \quad & q \in \mathbb{U} \\
s_1 \quad = \quad & (x_1, y_1) \\
s_2 \quad = \quad & (x_2, y_2) \\
q \quad = \quad & (x_q, y_q)
\end{aligned}
$$

$$\|q - s_1\|^2 \quad \gtrless \quad \|q - s_2\|^2$$

Is $q$ closer to $s_1$?



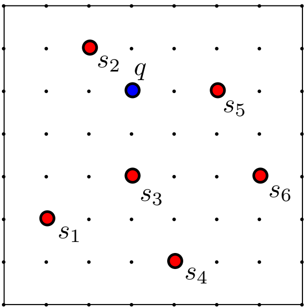

$$
\begin{aligned}
s_1, \quad s_2, \quad q &\in \mathbb{U} \\
s_1 &= (x_1, y_1) \\
s_2 &= (x_2, y_2) \\
q &= (x_q, y_q)
\end{aligned}
$$

$$\|q - s_1\|^2 \quad \gtrless \quad \|q - s_2\|^2$$

$$(x_q - x_1)^2 + (y_q - y_1)^2 \quad \gtrless \quad (x_q - x_2)^2 + (y_q - y_2)^2$$

$$\boxed{\text{degree } 2}$$

Which site is $q$ closest to?




**Given**

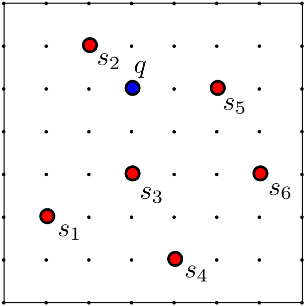
Sites $S = \{s_1, \dots, s_n\}$ and query point $q$ with $s_i, q \in \mathbb{U}$

**Determine**

The site of $S$ closest to $q$

# Post Office Query

Which site is $q$ closest to?




**Given**

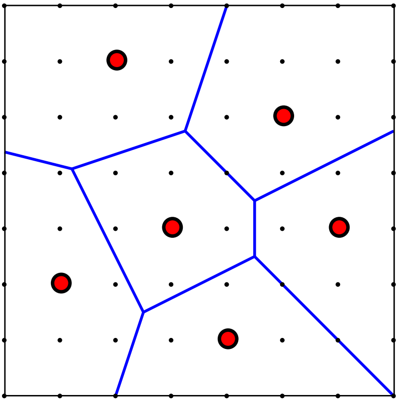
Sites $S = \{s_1, \ldots, s_n\}$ and query point $q$ with $s_i, q \in \mathbb{U}$

**Determine**

The site of $S$ closest to $q$

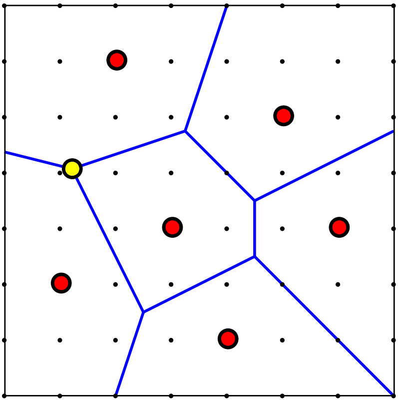
| | Preprocess | | Query | |
|---|---|---|---|---|
| | Alg | Time | Alg | Time |
| Brute force | - | - | deg 2 | $O(n)$ |
| Voronoi diagram | deg 4 | $O(n \log n)$ | deg 6 | $O(\log n)$ |
| Imp Voronoi [LPT99] | deg 5 | $O(n \log n)$ | deg 2 | $O(\log n)$ |

# Voronoi Review



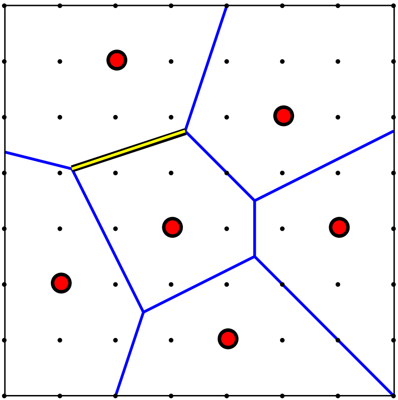
- Voronoi vertices
- Voronoi edges
- Voronoi cell
- Voronoi diagram
- Trapezoidation

# Voronoi Review



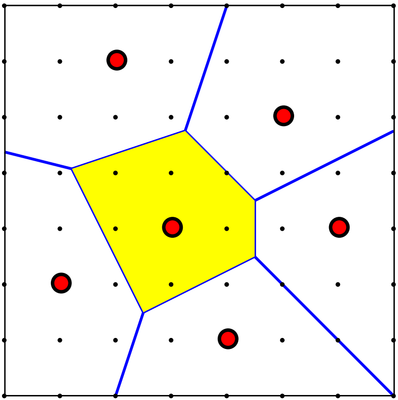
- **Voronoi vertices**
- Voronoi edges
- Voronoi cell
- Voronoi diagram
- Trapezoidation

# Voronoi Review

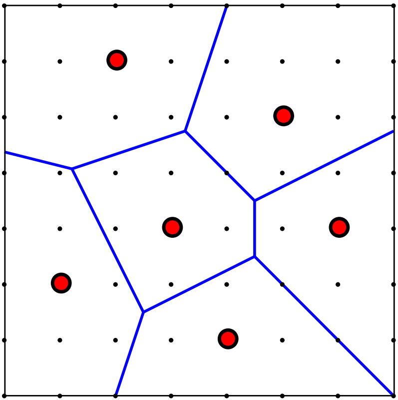


- Voronoi vertices
- **Voronoi edges**
- Voronoi cell
- Voronoi diagram
- Trapezoidation

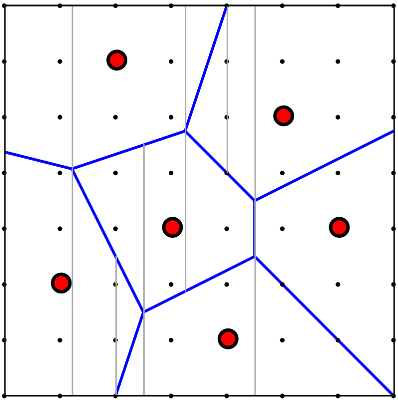
# Voronoi Review



- Voronoi vertices
- Voronoi edges
- **Voronoi cell**
- Voronoi diagram
- Trapezoidation



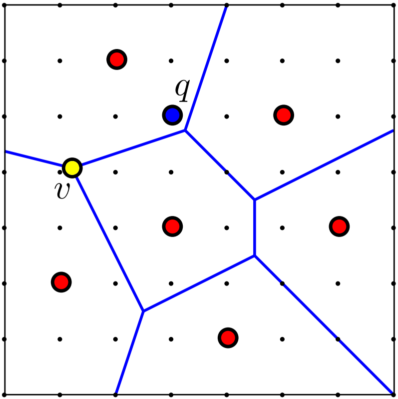
- Voronoi vertices
- Voronoi edges
- Voronoi cell
- **Voronoi diagram**
- Trapezoidation



- Voronoi vertices
- Voronoi edges
- Voronoi cell
- Voronoi diagram
- **Trapezoidation**

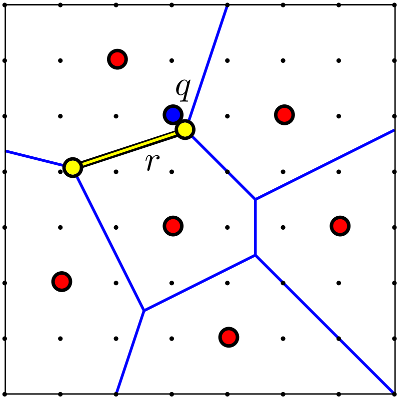
# Predicates and Their Precision

x-node:



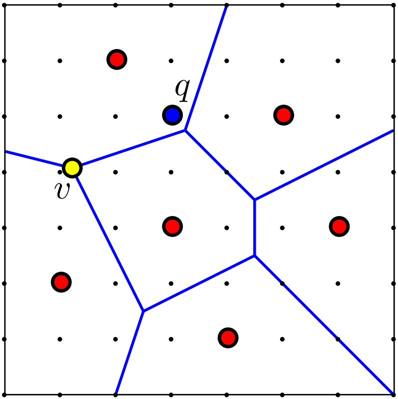

Is $x_q$ left/right of vertex $x_v$?
degree 3

y-node:



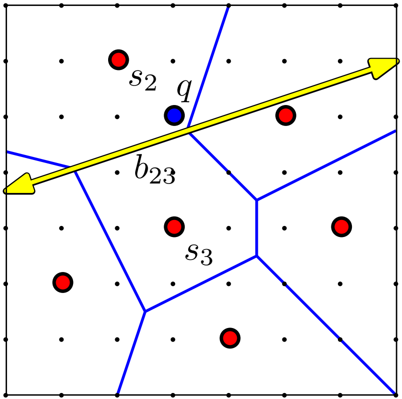

Is $q$ above/below segment $r$?
degree 6

x-node:




Is $x_q$ left/right of vertex $x_v$?
degree 3

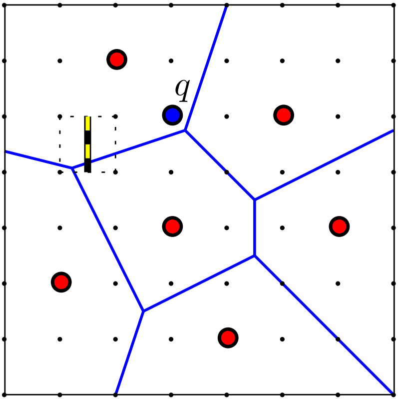
y-node:




Is $q$ closer to $s_2$ or $s_3$?
degree 2

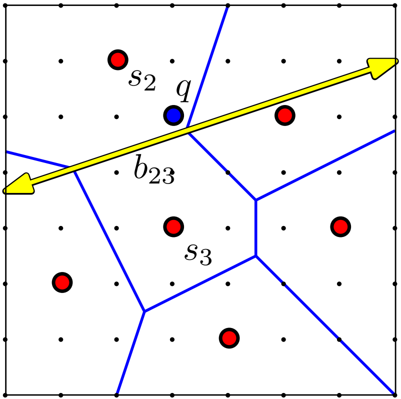
# Predicates and Their Precision

x-node:



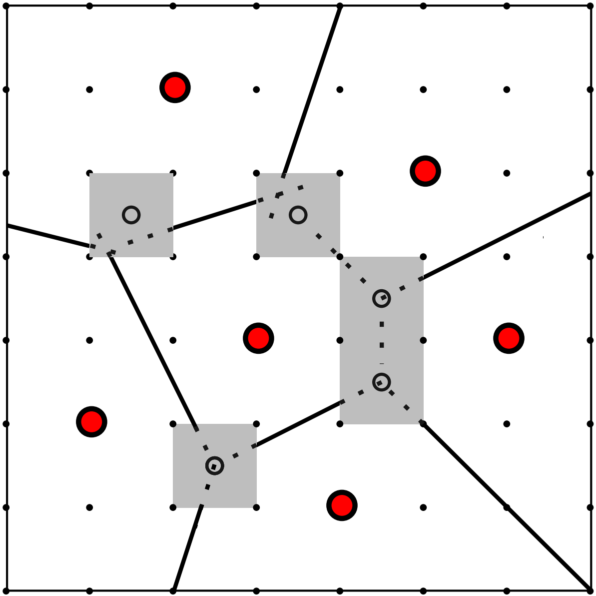

Is $x_q$ left/right of g.c. containing $v$?
degree 1

y-node:

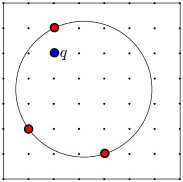



Is $q$ closer to $s_2$ or $s_3$?
degree 2

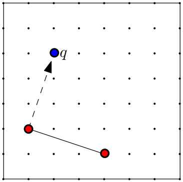
# Implicit Voronoi Diagram[LPT99]



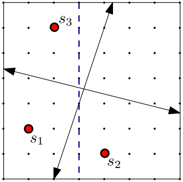
# Predicate Degree



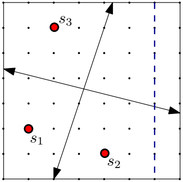

InCrcle
degree 4




Orient grid points
degree 2

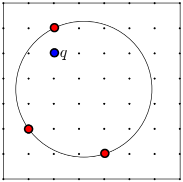



Order on a line
degree 3

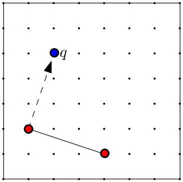



Bisector intersect.
left of vertical
degree 3

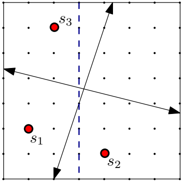
# Predicate Degree



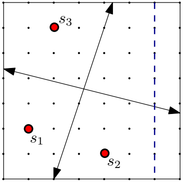

InCrcle
degree 4



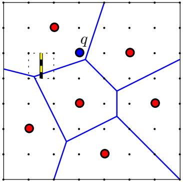

Orient grid points
degree 2



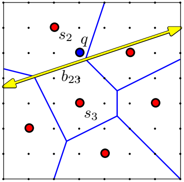

Order on a line
degree 3



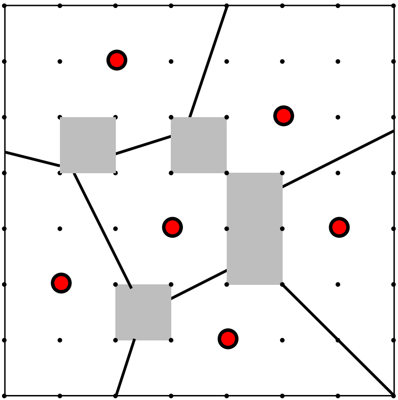

Bisector intersect.
left of vertical
degree 3




Left/right grid cell
degree 1




Side of a bisector
degree 2



**Given**

Sites $S = \{s_1, s_2, \ldots, s_n\} \subset \mathbb{U}$

**Construct**

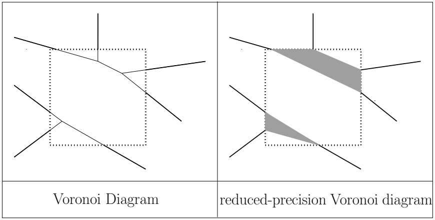
Implicit Voronoi with minimum precision.

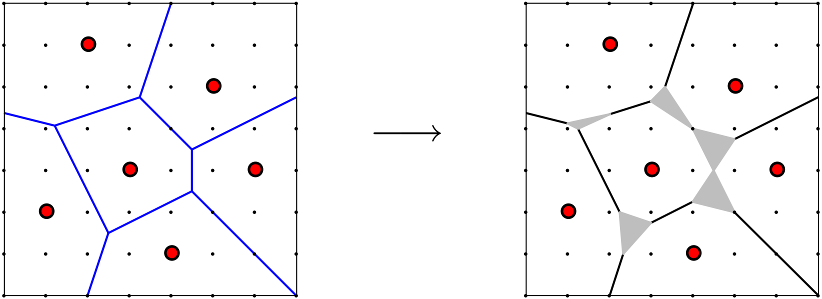
**Note**

Precision $<$ degree 4 precludes computing the Voronoi diagram.

# Reduced Precision Voronoi diagram

Replace connected subtrees of Voronoi edges inside a cell with their convex hulls.



# Construction (preview)

**Given** $n$ sites in $\mathbb{U}$.

## RP-Voronoi

Rand inc construction of the RP-Voronoi of $n$ sites in $\mathbb{U}$.

- Time: $O(n \log(Un))$ expected
- Space: $O(n)$ expected
- Precision: degree 3

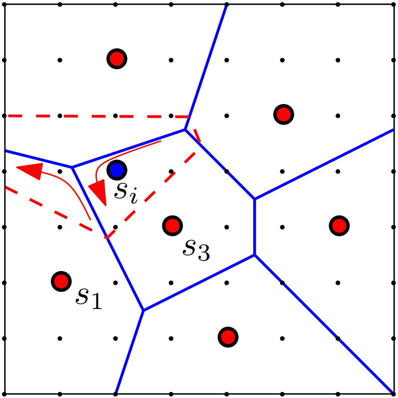
## Implicit Voronoi

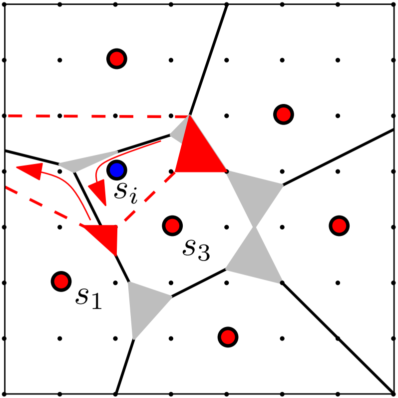
Construct LPT's implicit Voronoi from RP-Voronoi.

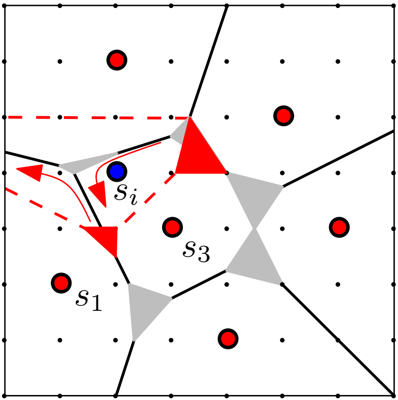
- Time: $O(n)$
- Space: $O(n)$ expected
- Precision: degree 3

# Rand Inc Construction

**Invariant:** Maintain RP-Voronoi as each new site is added.
**Update step:** Extension of [SI92], walk the deleted tree.



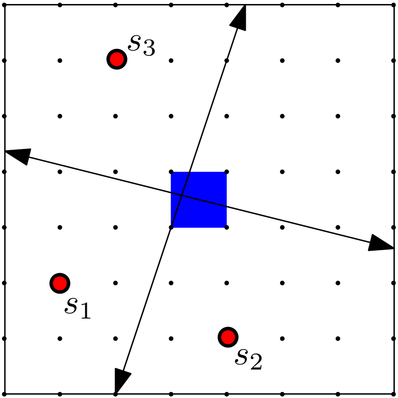






**Operations:**

- identify the grid cell containing a bisector intersection.
- determine the next edge in the tree walk.

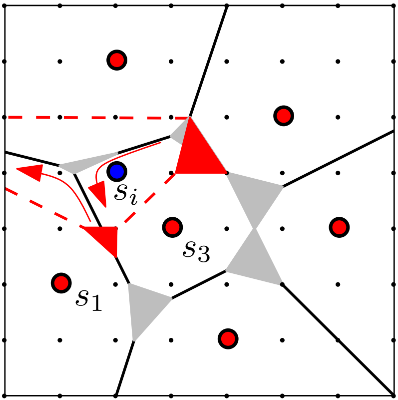
# Bisector Intersection




**Given**
Three sites $s_1$, $s_2$ and $s_3$.

**Find**
Grid cell containing
the intersection
of bisectors $b_{12}$ and $b_{13}$.

Time: $O(\log U)$
Precision: degree 3

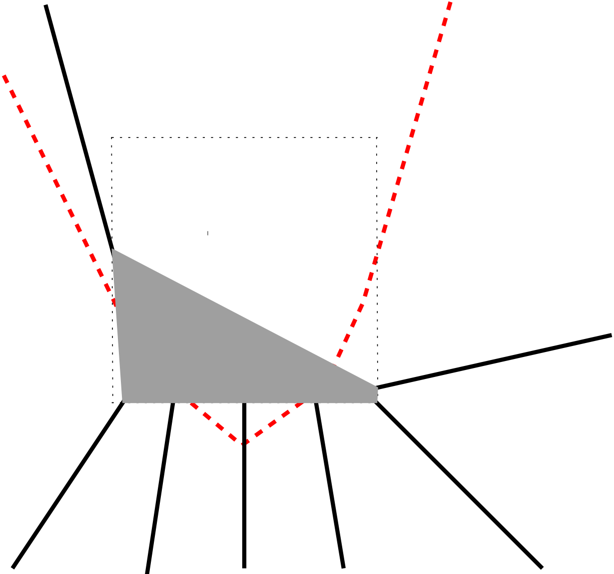



**Operations:**

- identify the grid cell containing a bisector intersection.
  Time: $O(\log U)$
  Precision: degree 3
- determine the next edge in the tree walk.

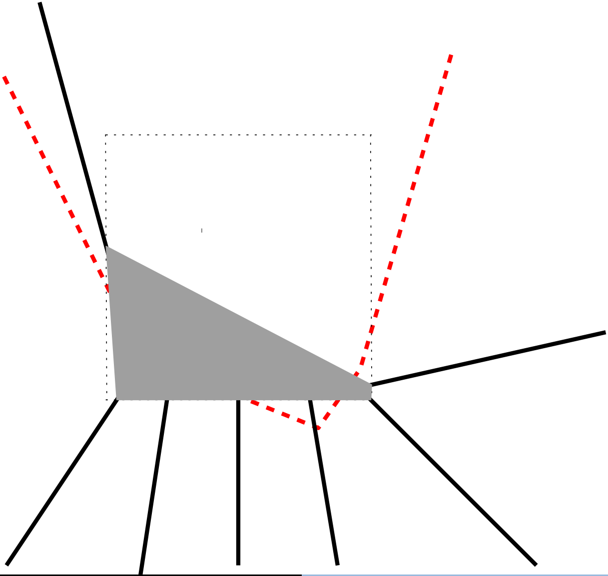
# Next Edge of the Tree Walk

Where do we walk once we have found an intersection?



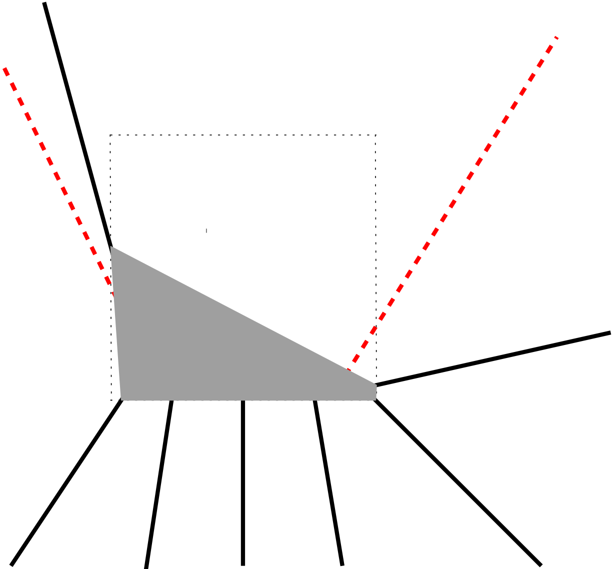
# Next Edge of the Tree Walk

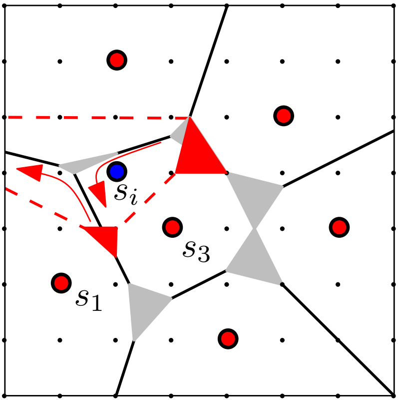
Where do we walk once we have found an intersection?



# Next Edge of the Tree Walk

Where do we walk once we have found an intersection?






**Operations:**

- identify the grid cell containing a bisector intersection.
  - Time: $O(\log U)$
  - Precision: degree 3
- determine the next edge in the tree walk.
  - Time: $O(\log n)$
  - Precision: degree 3

# RIC update for RP-Voronoi

Update step to add a new site $s_i$

(1) Find cell containing $s_i$
(2) Identify bisectors and cells in the new cell of $s_i$
(3) Walk tree inside cell of $s_i$
   (a) Binary search bisector for crossing
   (b) Compute grid intersection with cell $s_i$

# Analysis

RIC Facts for Voronoi

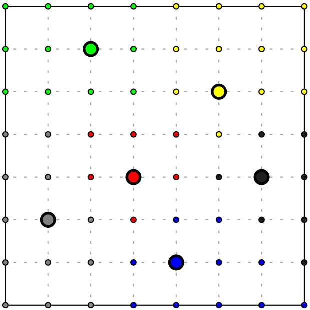
- Creates $\Theta(n)$ vertices expected
- Point location takes $\Theta(n \log n)$ expected

Charging scheme

Binary searchers for (3a) $O(\log U)$ and (3b) $O(\log n)$ charged to vertex creation.

Therefore,
It takes $O(n)$ space and $O(n(\log n + \log U))$ time to build the reduced-precision Voronoi diagram of $n$ sites on a grid of size $U$ with degree 3.

# Discrete Voronoi Diagram



**Given**

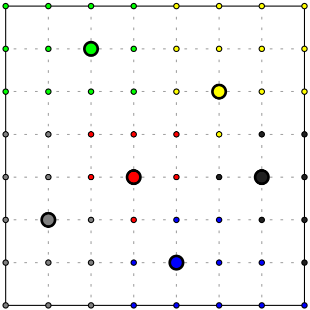
A grid of size $U$ and

Sites $S = \{s_1, \ldots, s_n\} \subset \mathbb{U}$

**Label**

Each grid point of $\mathbb{U}$ with the closest site of $S$

# Discrete Voronoi Diagram



**Given**

A grid of size $U$ and

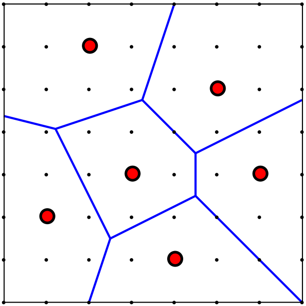
Sites $S = \{s_1, \ldots, s_n\} \subset \mathbb{U}$

**Label**

Each grid point of $\mathbb{U}$ with the closest site of $S$

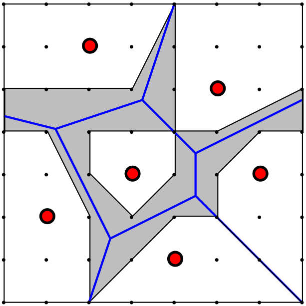
| | Alg | Time |
|---|---|---|
| Brute Force | deg 2 | $O(nU^2)$ |
| Query the Voronoi diagram | deg 4 | $O(U^2 \log n)$ |
| Nearest Neighbor Trans. [B90] | deg 4 | $O(U^2)$ |
| Discrete Voronoi diagram [C06] | deg 3 | $O(U^2)$ |
| GPU Hardware [H99] | - | $\Theta(nU^2)$ |

# Voronoi Polygon Set



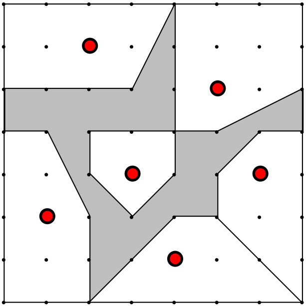
- Partition of $\mathbb{U}$
- $n$ convex polygons $\{C(s_1), \ldots, C(s_n)\}$

# Voronoi Polygon Set



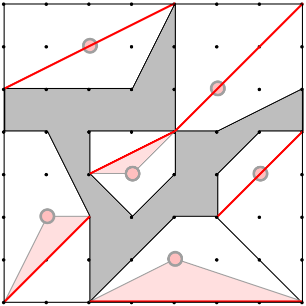
- Partition of $\mathbb{U}$
- $n$ convex polygons $\{C(s_1), \ldots, C(s_n)\}$
- Where $C(s_i)$ is the convex hull of the grid points in the Voronoi cell of $s_i$

# Voronoi Polygon Set

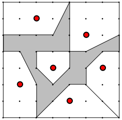


- Partition of $\mathbb{U}$
- $n$ convex polygons $\{C(s_1), \ldots, C(s_n)\}$
- Where $C(s_i)$ is the convex hull of the grid points in the Voronoi cell of $s_i$
- Grey gaps

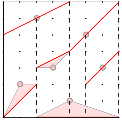
# Voronoi Polygon Set



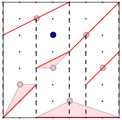
- Partition of $\mathbb{U}$
- $n$ convex polygons $\{C(s_1), \ldots, C(s_n)\}$
- Where $C(s_i)$ is the convex hull of the grid points in the Voronoi cell of $s_i$
- Grey gaps
- Proxy segment



Compute Voronoi Polygon Set

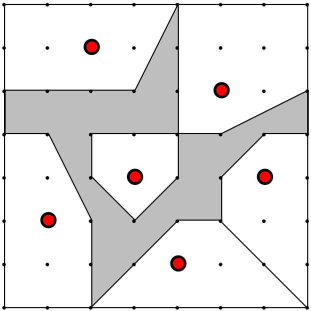


Compute trapezoid graph of proxies



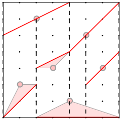
Query the post office structure

# Construct and Query Post Office Structure

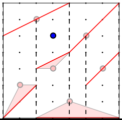


Compute *Voronoi Polygon Set*

- Time: $O(U^2)$ expected time
- Space: $O(n \log U)$
- Precision: degree 2

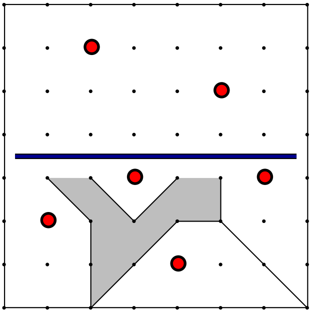


Compute trapezoid graph of proxies



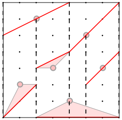
Query the post office structure

# Construct and Query Post Office Structure

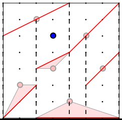


Compute *Voronoi Polygon Set*

- Time: $O(U^2)$ expected time
- Space: $O(n \log U)$
- Precision: degree 2

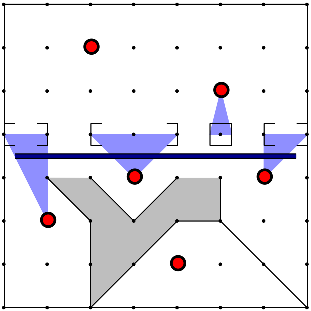


Compute trapezoid graph of proxies



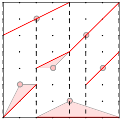
Query the post office structure

# Construct and Query Post Office Structure

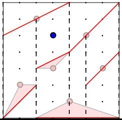


Compute *Voronoi Polygon Set*

- Time: $O(U^2)$ expected time
- Space: $O(n \log U)$
- Precision: degree 2

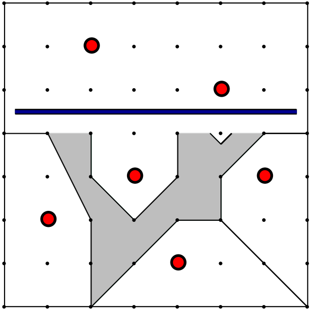


Compute trapezoid graph of proxies



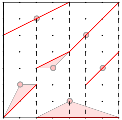
Query the post office structure

# Construct and Query Post Office Structure

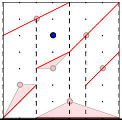


Compute *Voronoi Polygon Set*

- Time: $O(U^2)$ expected time
- Space: $O(n \log U)$
- Precision: degree 2

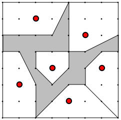


Compute trapezoid graph of proxies

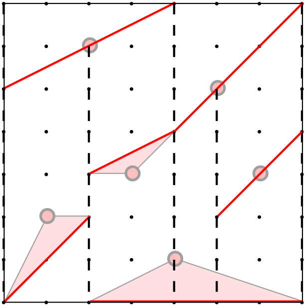


Query the post office structure

# Construct and Query Post Office Structure

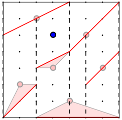


Compute Voronoi Polygon Set



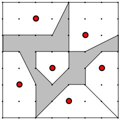
Compute trapezoid graph of proxies

- Time: $O(n \log n)$ expected
- Space: $O(n)$ expected
- Precision: degree 2

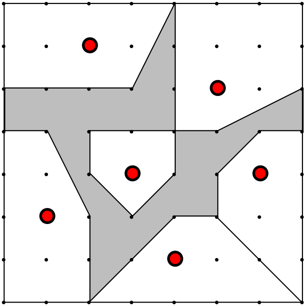


Query the post office structure

# Construct and Query Post Office Structure

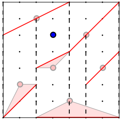


Compute Voronoi Polygon Set



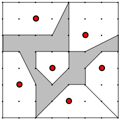
Compute trapezoid graph of proxies

- Time: $O(n \log n)$ expected
- Space: $O(n)$ expected
- Precision: degree 2

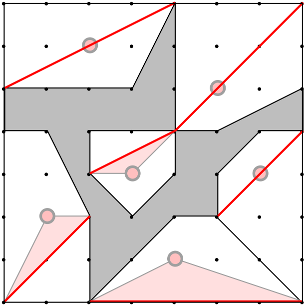


Query the post office structure

# Construct and Query Post Office Structure

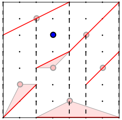


Compute Voronoi Polygon Set



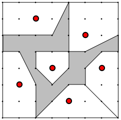
Compute trapezoid graph of proxies

- Time: $O(n \log n)$ expected
- Space: $O(n)$ expected
- Precision: degree 2

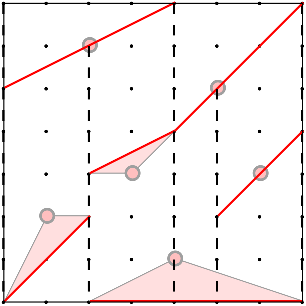


Query the post office structure

# Construct and Query Post Office Structure

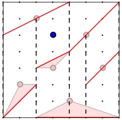


Compute Voronoi Polygon Set



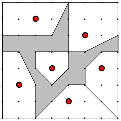
Compute trapezoid graph of proxies

- Time: $O(n \log n)$ expected
- Space: $O(n)$ expected
- Precision: degree 2

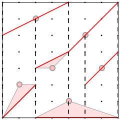


Query the post office structure

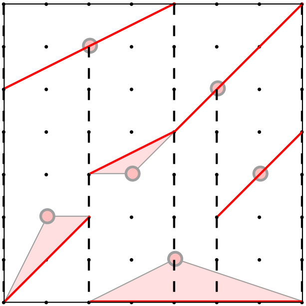
# Construct and Query Post Office Structure



Compute Voronoi Polygon Set

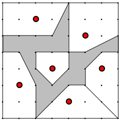


Compute trapezoid graph of proxies

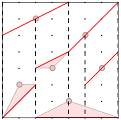


Query the post office structure

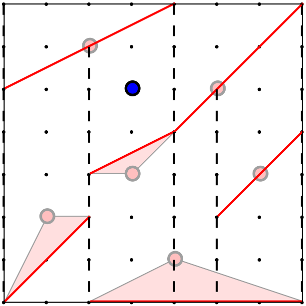
- Time: $O(\log n)$ expected
- Precision: degree 2



Compute Voronoi Polygon Set



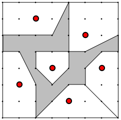
Compute trapezoid graph of proxies



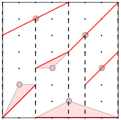
Query the post office structure

- Time: $O(\log n)$ expected
- Precision: degree 2

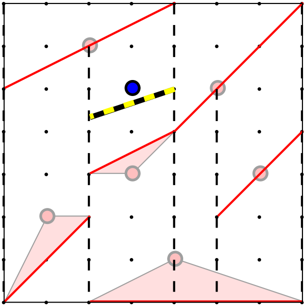
# Construct and Query Post Office Structure



Compute Voronoi Polygon Set



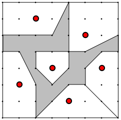
Compute trapezoid graph of proxies



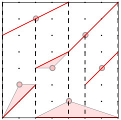
Query the post office structure

- Time: $O(\log n)$ expected
- Precision: degree 2

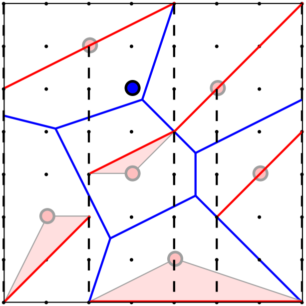
# Construct and Query Post Office Structure



Compute Voronoi Polygon Set



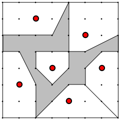
Compute trapezoid graph of proxies



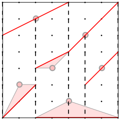
Query the post office structure

- Time: $O(\log n)$ expected
- Precision: degree 2

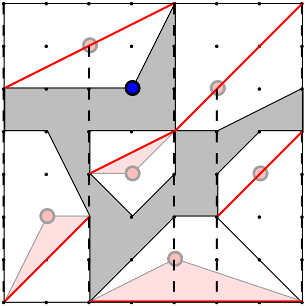
# Construct and Query Post Office Structure



Compute Voronoi Polygon Set



Compute trapezoid graph of proxies



Query the post office structure

- Time: $O(\log n)$ expected
- Precision: degree 2

# Results

Assuming $O(n \log n) < O(U^2)$

### Compute Voronoi Polygon Set

- Time: $O(U^2)$ expected time
- Space: $O(n \log U)$ and $O(n)$ for proxies only
- Precision: degree 2

### Query Post Office Structure

- Time: $O(\log n)$ expected
- Precision: degree 2

### Compute 2D discrete Voronoi

- Time: $O(U^2)$ expected time.
- Space: $O(n + U)$
- Precision: degree 2

# Recent Results

Construct degree 2 post office query structure, called D2-Voronoi

- Time: $O(n \log n \log U)$ expected
- Space: $O(n)$ expected
- Precision: degree 2
- Query time: $O(\log n)$
- Query precision: degree 2

# Open problems

Optimized implementations and experiments for ...

- RP-Voronoi, Discrete Voronoi, D2-Voronoi

Can we ...

- Remove additive $\log U$ factor in RP-Voronoi?
- Remove multiplicative $\log U$ factor in D2-Voronoi?
- Or, show some lower bound inherent in reduced precision?
- generalize to higher dimension?
- treat precision as a limited resource (like time and space) when solving other algorithmic problems?

# Open problems

Optimized implementations and experiments for ...

- RP-Voronoi, Discrete Voronoi, D2-Voronoi

Can we ...

- Remove additive $\log U$ factor in RP-Voronoi?
- Remove multiplicative $\log U$ factor in D2-Voronoi?
- Or, show some lower bound inherent in reduced precision?
- generalize to higher dimension?
- treat precision as a limited resource (like time and space) when solving other algorithmic problems?

# Thank you!


**Contact information**

name: David L. Millman

email: dave@cs.unc.edu

site: cs.unc.edu/∼dave or millman.us