Finding the k-Visibility Region of a Point in a Simple Polygon in the Memory-Constrained Model

Yeganeh Bahoo
University of Manitoba
Winnipeg, Canada

Bahareh Banyassady
Freie Universität
Berlin, Germany

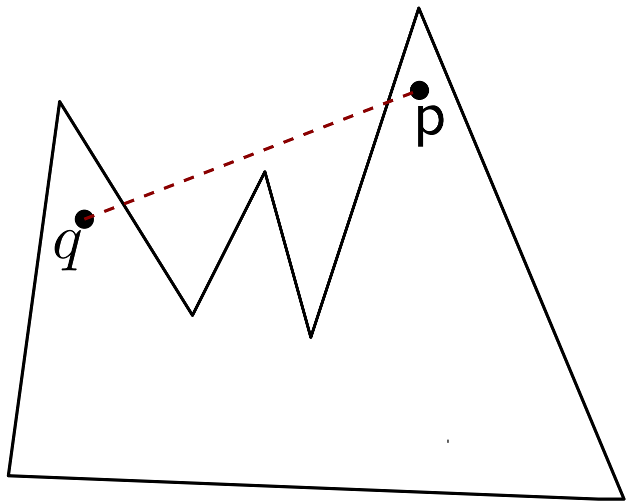
Stephane Durocher
University of Manitoba
Winnipeg, Canada

Prosenjit Bose
Carleton University
Ottawa, Canada

Wolfgang Mulzer
Freie Universität
Berlin, Germany

k-visibility region

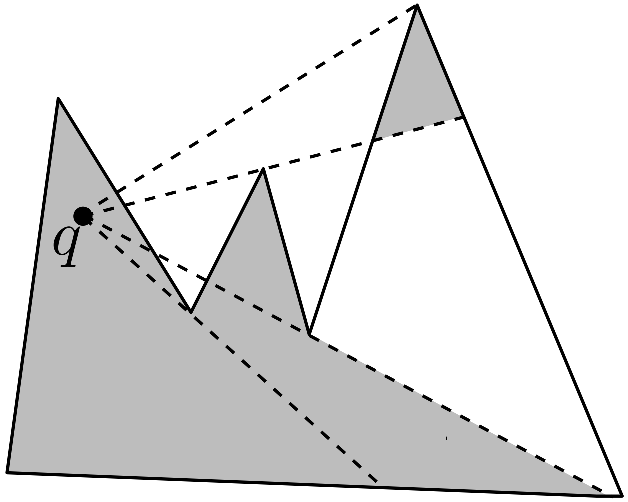
From a given point $q \in P$, a point $p \in P$ is $\boldsymbol{k}$-**visible** iff the segment pq properly intersects ∂P at most k times.



k-visibility region

From a given point $q \in P$, a point $p \in P$ is $\boldsymbol{k}$**-visible** iff the segment pq properly intersects ∂P at most k times.

For a given polygon P and a given point $q \in P$, the set of k-visible points of P from q is called the $\boldsymbol{k}$**-visibility region** of q within P, and is denoted by $V_k(P, q)$.

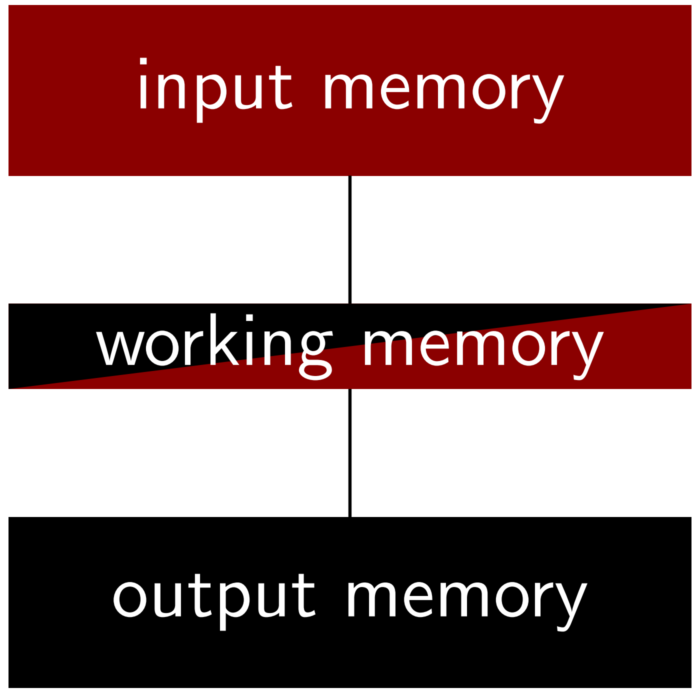


Model

read-only n words

read/write $O(s)$ words

write-only



word $= O(\log n)$ bits

k-visibility region in constrained-memory model

Input: A simple polygon P in a read-only array, a point $q \in P$ and a constant $k \in \mathbb{N}$.
Output: A representation of $V_k(P, q)$.

Theorem:
For a given simple polygon P, a given point $q \in P$ and $k \in \mathbb{N}$, we can report $V_k(P, q)$ in $O((cn + kn)/s + n \log s)$ time using $O(s)$ workspace.

- $O(1)$ space: $O(cn + kn)$ time
- $O(n)$ space: $O(n \log n)$ time

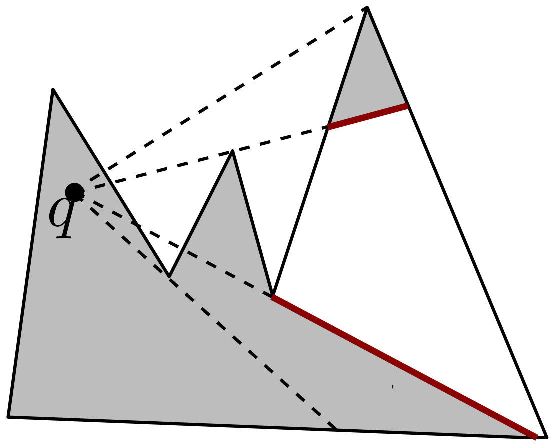
Known results

	Space	Running time	Authors
0-visibility	$O(n)$	$O(n)$	Joe & B. Simpson BIT Nume. Math. 1987
	$O(1)$	$O(n\bar{r})$	Barba, Korman, Langerman & I. Silveira JoCG 2014
	$O(s)$ $s \in O(\log r)$	$O(nr/2^s + n\log^2 r)$	
		$O(nr/2^s + n\log r)$ expected time	
k-visibility	$O(n^2)$	$O(n^2)$	Bajuelos, et al. J. UCS 2012
	$O(1)$	$O(cn + kn)$	Bahoo, Banyassady, Droucher, Bose, Mulzer. EuroCG 2016.
	$O(s)$	$O((c+k)n/s + n\log s)$	

Properties of $V_k(P, q)$

$\partial V_k(P, q)$ consists of

- part of ∂P
- windows: some chords inside P



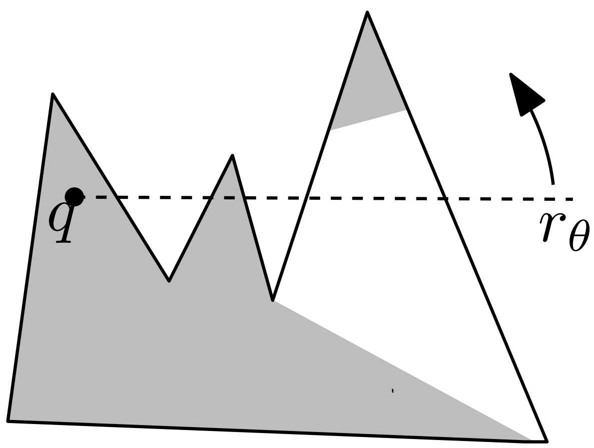
Properties of $V_k(P, q)$

$\partial V_k(P, q)$ consists of

- part of ∂P
- windows: some chords inside P

Suppose r_θ is a ray from q in direction θ:

- Only the first $k + 1$ intersections of $r_\theta \cap \partial P$ are k-visible from q.
- The list of intersecting edges of r_θ changes only if r_θ stabs a vertex of P.



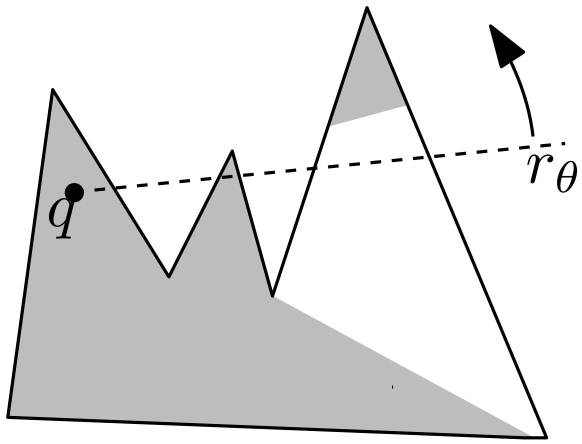
Properties of $V_k(P, q)$

$\partial V_k(P, q)$ consists of

- part of ∂P
- windows: some chords inside P

Suppose r_θ is a ray from q in direction θ:

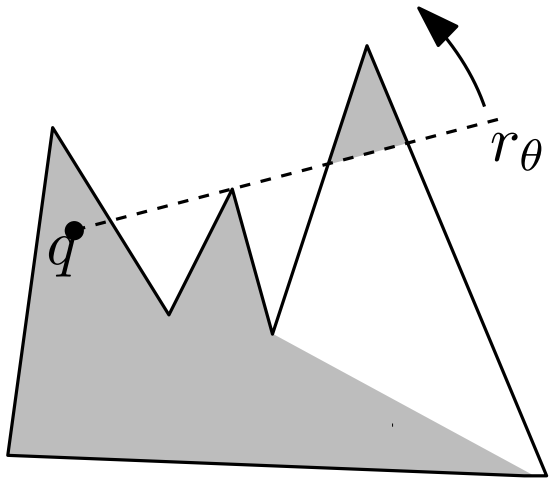
- Only the first $k + 1$ intersections of $r_\theta \cap \partial P$ are k-visible from q.
- The list of intersecting edges of r_θ changes only if r_θ stabs a vertex of P.



Properties of $V_k(P,q)$

$\partial V_k(P,q)$ consists of

- part of ∂P
- windows: some chords inside P



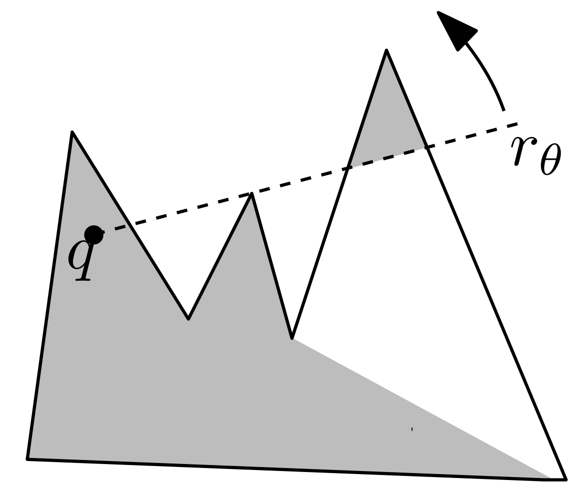
Suppose r_θ is a ray from q in direction θ:

- Only the first $k+1$ intersections of $r_\theta \cap \partial P$ are k-visible from q.
- The list of intersecting edges of r_θ changes only if r_θ stabs a vertex of P.

Properties of $V_k(P, q)$

$\partial V_k(P, q)$ consists of

- part of ∂P
- windows: some chords inside P



Suppose r_θ is a ray from q in direction θ:

- Only the first $k + 1$ intersections of $r_\theta \cap \partial P$ are k-visible from q.
- The list of intersecting edges of r_θ changes only if r_θ stabs a vertex of P.

Non-critical vertex

Critical vertex

Properties of $V_k(P, q)$

$\partial V_k(P, q)$ consists of

- part of ∂P
- windows: some chords inside P

Suppose r_θ is a ray from q in direction θ:

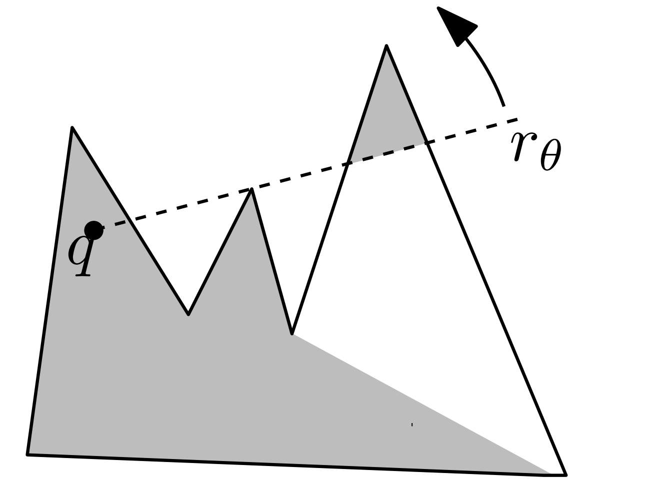
- Only the first $k+1$ intersections of $r_\theta \cap \partial P$ are k-visible from q.
- The list of intersecting edges of r_θ changes only if r_θ stabs a vertex of P.

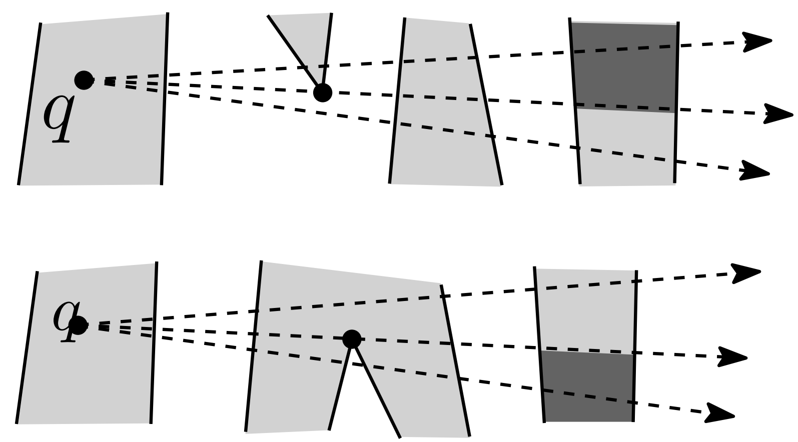
Non-critical vertex

Critical vertex

If it is k-visible

Window: the segment on r_θ between $e_\theta(k+2)$ and $e_\theta(k+3)$ (if they exist) is an edge of $\partial V_k(P, q)$.

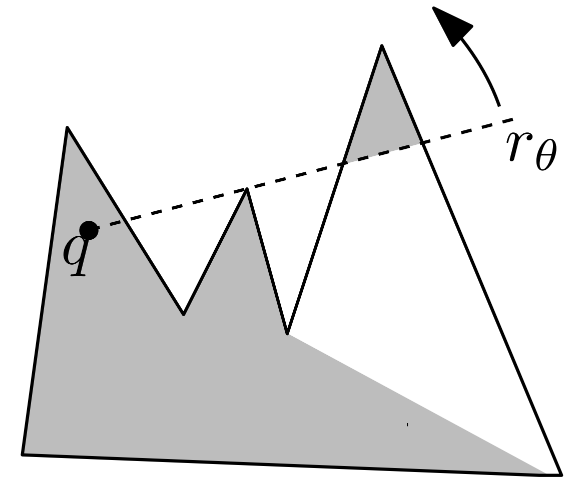




Properties of $V_k(P, q)$

$\partial V_k(P, q)$ consists of

- part of ∂P
- windows: some chords inside P



Suppose r_θ is a ray from q in direction θ:

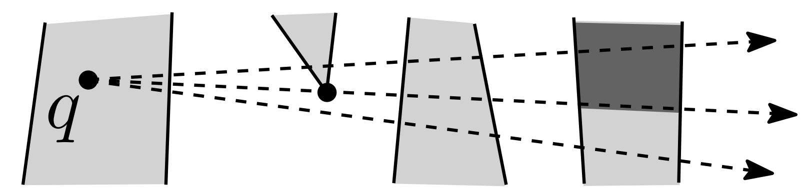
- Only the first $k + 1$ intersections of $r_\theta \cap \partial P$ are k-visible from q.
- The list of intersecting edges of r_θ changes only if r_θ stabs a vertex of P.

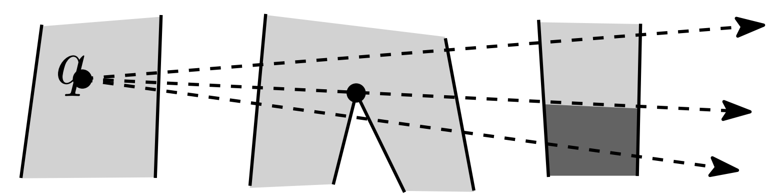
Non-critical vertex

Critical vertex

If it is k-visible

Window: the segment on r_θ between $e_\theta(k + 2)$ and $e_\theta(k + 3)$ (if they exist) is an edge of $\partial V_k(P, q)$.





Given P and the set of windows, $W_k(P, q)$, we can uniquely report $\partial V_k(P, q)$.

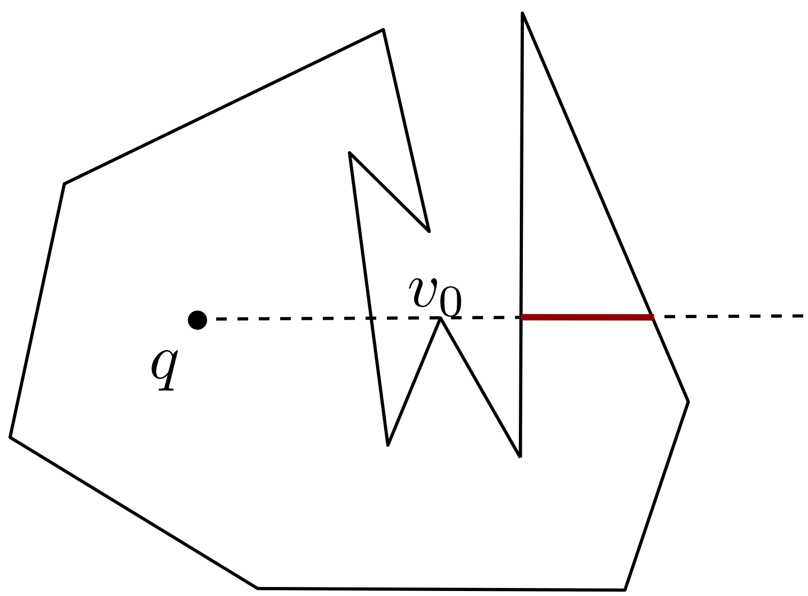
Computing $V_k(P, q)$ in $O(1)$ space –Overview

- Select v_0, the critical vertex with smallest angle. $\rightarrow O(n)$
- Find $e_0(k+1)$ using the k-selection algorithm. $\rightarrow O(kn)$
- If v_0 is k-visible then find the window of qv_0. $\rightarrow O(n)$

- Find v_1, next critical vertex with smallest angle. $\rightarrow O(n)$
- Find $e_1(k+1)$ using $e_0(k+1)$. $\rightarrow O(n)$ $\Big\}$ $O(c)$ times
- If v_1 is k-visible then find the window of qv_1. $\rightarrow O(n)$

- Repeat the last three steps for all critical vertices.

Running time: $O(kn + cn)$



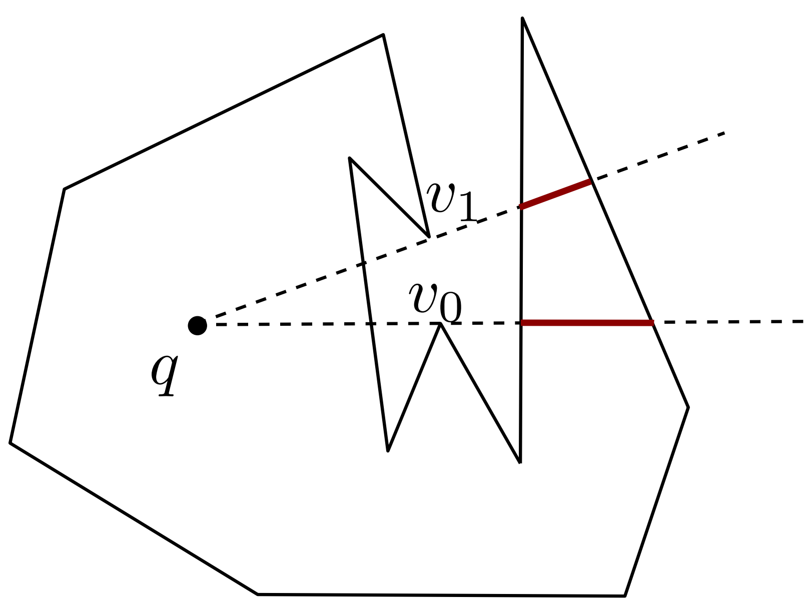
Computing $V_k(P, q)$ in $O(1)$ space –Overview

- Select v_0, the critical vertex with smallest angle. $\to O(n)$
- Find $e_0(k+1)$ using the k-selection algorithm. $\to O(kn)$
- If v_0 is k-visible then find the window of qv_0. $\to O(n)$

$$\left.\begin{array}{ll} \bullet\ \text{Find } v_1\text{, next critical vertex with smallest angle.} & \to O(n) \\ \bullet\ \text{Find } e_1(k+1) \text{ using } e_0(k+1). & \to O(n) \\ \bullet\ \text{If } v_1 \text{ is } k\text{-visible then find the window of } qv_1. & \to O(n) \end{array}\right\} O(c) \text{ times}$$

- Repeat the last three steps for all critical vertices.

$\boxed{\text{Running time: } O(kn + cn)}$



Computing $V_k(P, q)$ in $O(s)$ space –Overview

- Select v_0, find $e_0(k+1)$ and the window of qv_0. $\rightarrow O(n + k - selection)$
- Find the next s critical vertex with smallest angle, $v_1, v_2, \ldots, v_s$, and insert them in a BST. $\rightarrow O(n + s \log s)$
- ? $\rightarrow O(?)$

$O(c/s)$ times

- Repeat for the next s critical vertices.

$\boxed{\text{Running time: } O(c/s(n + s\log s + ?) + kn/s)}$

Computing $V_k(P,q)$ in $O(s)$ space –Overview

- Select v_0, find $e_0(k+1)$ and the window of qv_0. $\rightarrow O(n + k - selection)$
- Find the next s critical vertex with smallest angle, $v_1, v_2, \ldots, v_s$, and insert them in a BST. $\rightarrow O(n + s\log s)$
- ? $\rightarrow O(?)$
- Repeat for the next s critical vertices.

$O(c/s)$ times

Running time: $O(c/s(n + s\log s + ?) + kn/s)$

There is an algorithm that finds the s smallest element in a read-only array of size n, in $O(n)$ time using $O(s)$ workspace.

M. Chan & Y. Chen. DCG 2007

Computing $V_k(P,q)$ in $O(s)$ space –Overview

- Select v_0, find $e_0(k+1)$ and the window of qv_0. $\rightarrow O(n+k-selection)$
- Find the next s critical vertex with smallest angle, $v_1, v_2, \ldots, v_s$, and insert them in a BST. $\rightarrow O(n + s\log s)$
- ? $\rightarrow O(?)$
- Repeat for the next s critical vertices.

$O(c/s)$ times

Running time: $O(c/s(n + s\log s + ?) + kn/s)$

There is an algorithm that finds the s smallest element in a read-only array of size n, in $O(n)$ time using $O(s)$ workspace.

M. Chan & Y. Chen. DCG 2007

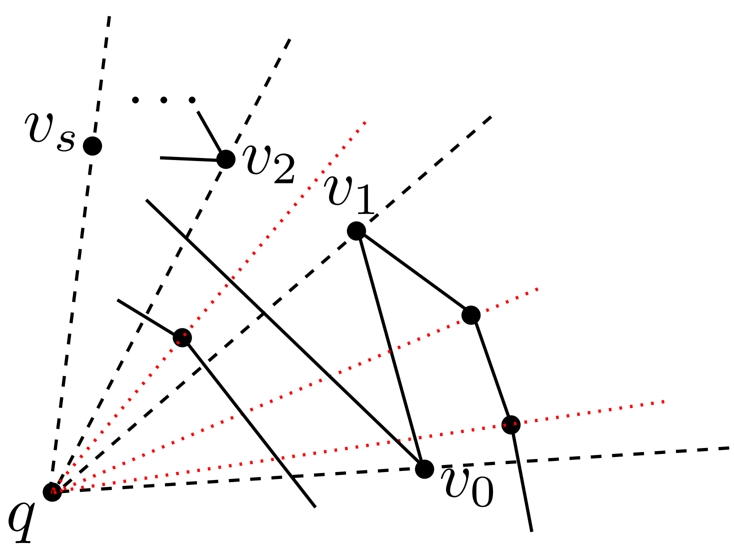
There is an algorithm that finds the k^{th} smallest element in a read-only array array of size n in $O(\lceil k/s \rceil n)$ time using $O(s)$ workspace.

Computing $V_k(P,q)$ in $O(s)$ space –Overview

- Find $2s$ intersecting edges to the right/left of $e_0(k+1)$ on qv_0 and sort them in memory. $\rightarrow O(n + s\log s)$
- For each edge in T determine the larger angle of its endpoints, and insert them in T_θ. $\rightarrow O(s \log s)$
- For $v_{i\in\{1,2,\ldots,s\}}$, find $e_i(k+1)$ and the window of qv_i using $e_{i-1}(k+1)$ and T. $\rightarrow O(1)$
- Update T and T_θ. $\rightarrow O(c_i \log s)$
- Repeat for the s critical vertices.

$O(s)$ times

$\boxed{\text{Running time: } O(n + s\log s + s(c_i \log s))}$



Summary

Theorem: For a given simple polygon P in a read-only array, a point $q \in P$ and a constant $k \in \mathbb{N}$, we can report $W_k(P, q)$ using $O(s)$ **workspace** in $O((cn + kn)/s + n \log s)$ **time**.

Summary

Theorem: For a given simple polygon P in a read-only array, a point $q \in P$ and a constant $k \in \mathbb{N}$, we can report $W_k(P, q)$ and $\partial V_k(P, q)$ using $O(s)$ **workspace** in $O((cn + kn)/s + n \log s)$ **time**.

Summary

non-simple polygon
set of segments

Theorem: For a given simple polygon P in a read-only array, a point $q \in P$ and a constant $k \in \mathbb{N}$, we can report $W_k(P, q)$ and $\partial V_k(P, q)$ using $O(s)$ **workspace** in $O((cn + kn)/s + n \log s)$ **time**.

Summary

non-simple polygon
set of segments

Theorem: For a given simple polygon P in a read-only array, a point $q \in P$ and a constant $k \in \mathbb{N}$, we can report $W_k(P,q)$ and $\partial V_k(P,q)$ using $O(s)$ **workspace** in $O((cn+kn)/s + n\log s)$ **time**.

QUESTIONS