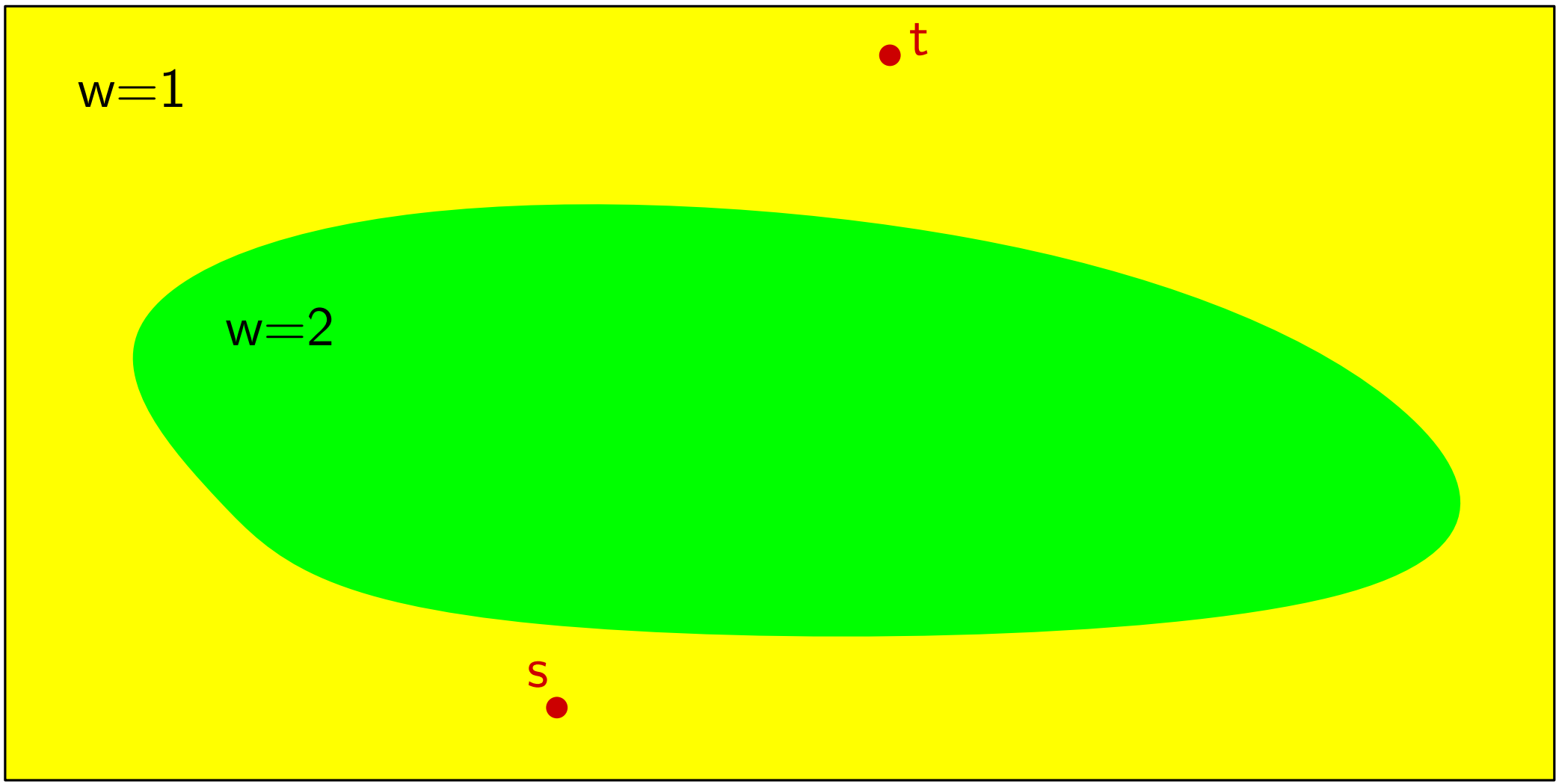
Querying approximate shortest paths in anisotropic regions

Siu-Wing Cheng (HKUST)
Hyeon-Suk Na (Soongsil University)
Antoine Vigneron (INRA Jouy-en-Josas)
Yajun Wang (HKUST)

The weighted region problem

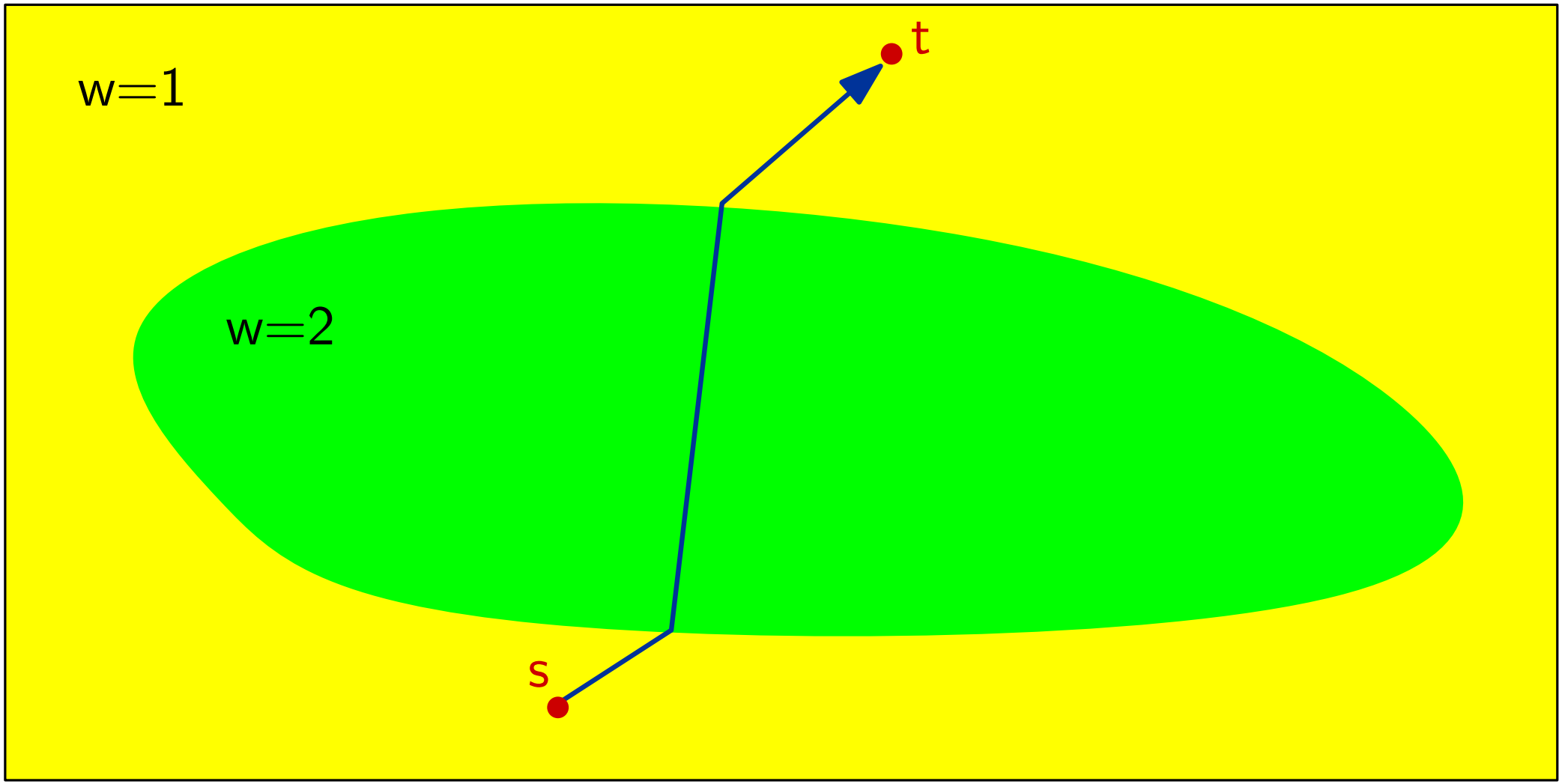
cost = weight $\times$ length



Shortest path

The weighted region problem

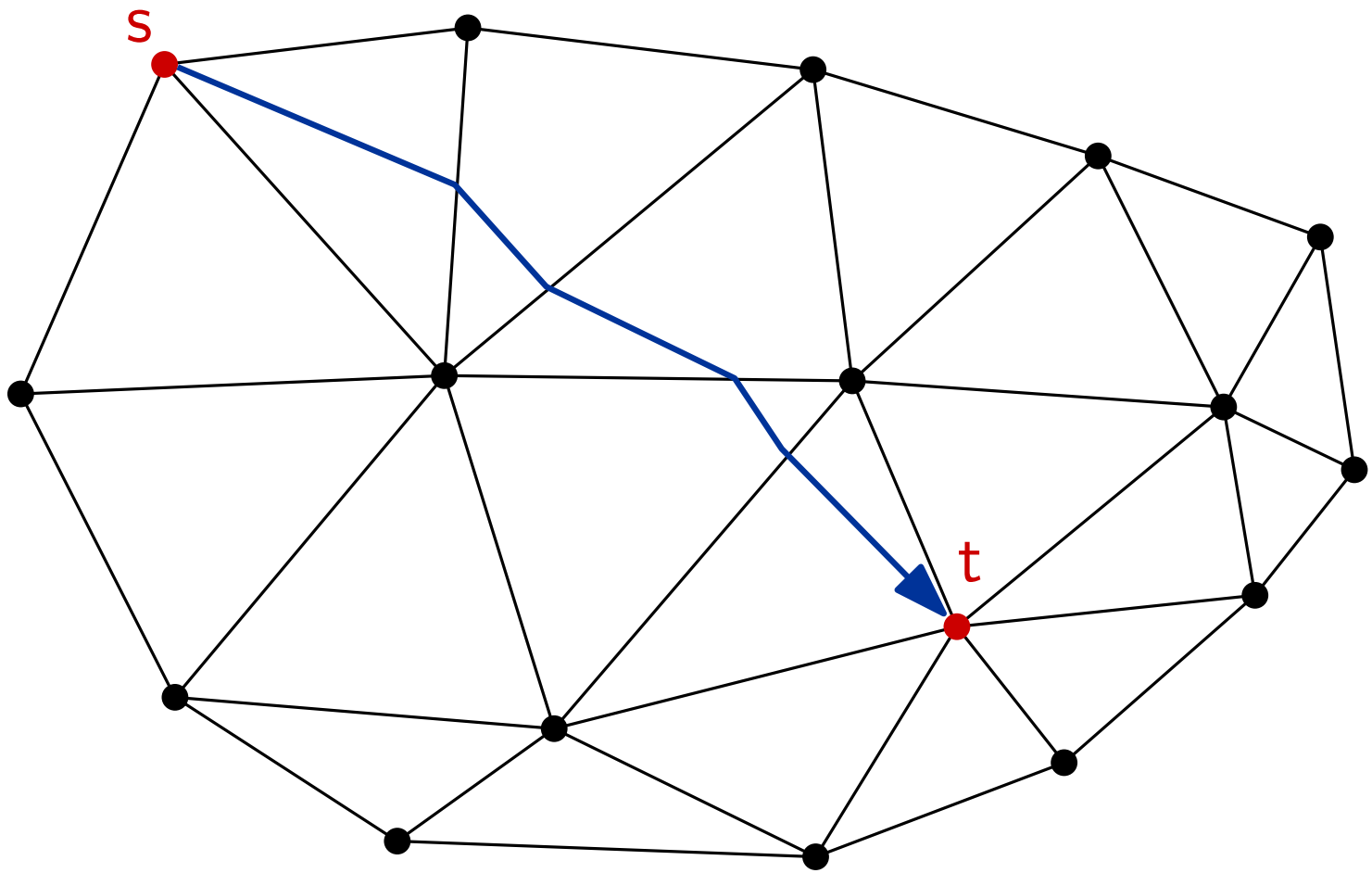
cost = weight $\times$ length



Shortest path

The weighted region problem

- Triangulation with n vertices
- Each face has a weight
- Find an approximate shortest path with cost at most $1 + \epsilon$ times the minimum



No (strongly polynomial) FPTAS is known

No (strongly polynomial) FPTAS is known

Known results: polynomial in n, $1/\epsilon$, and other parameters.

No (strongly polynomial) FPTAS is known

Known results: polynomial in n, $1/\epsilon$, and other parameters.

- ρ: maximum weight/ minimum weight
- θ: minimum angle in the triangulation

No (strongly polynomial) FPTAS is known

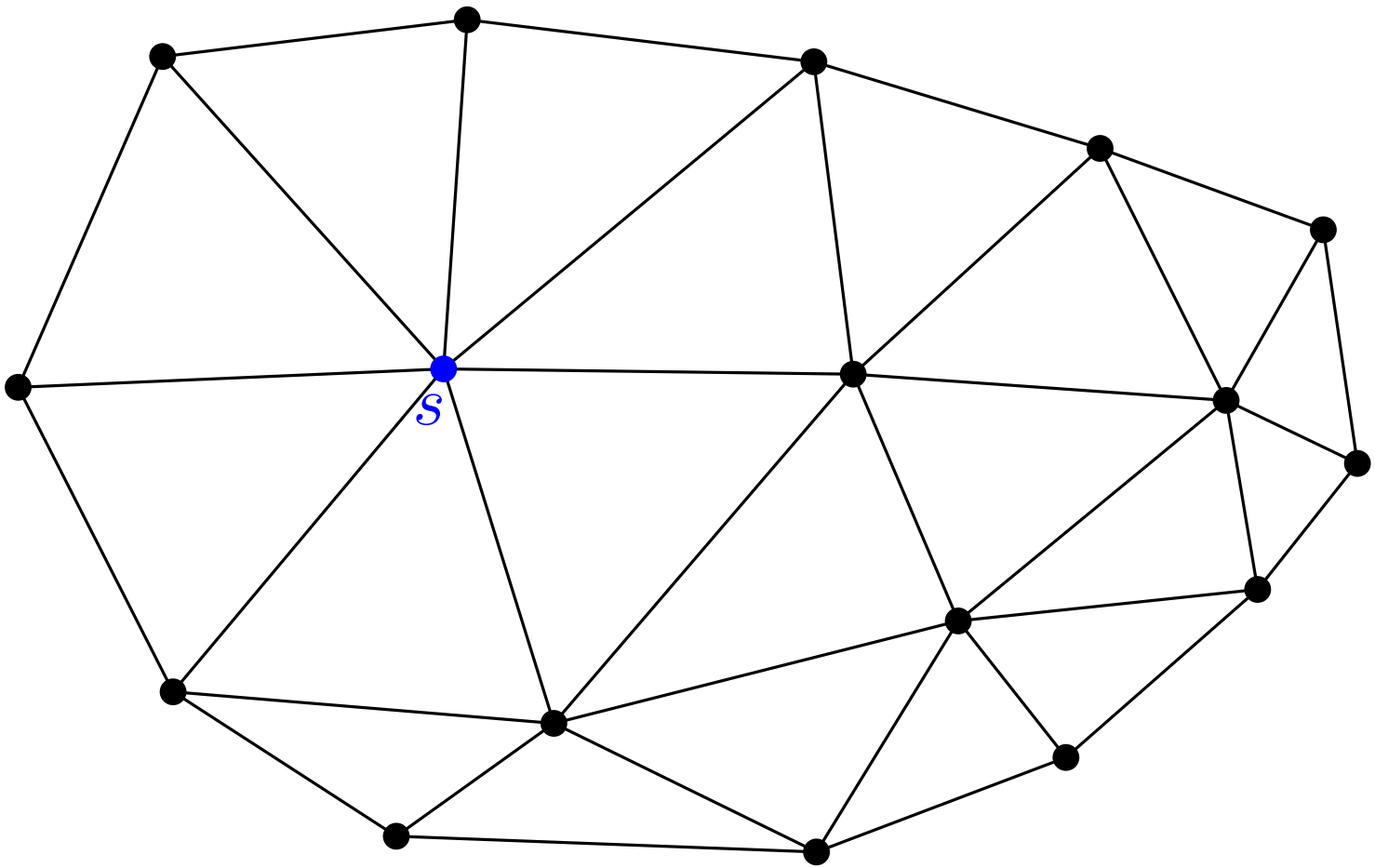
Known results: polynomial in n, $1/\epsilon$, and other parameters.

- ρ: maximum weight/ minimum weight
- θ: minimum angle in the triangulation

Known results

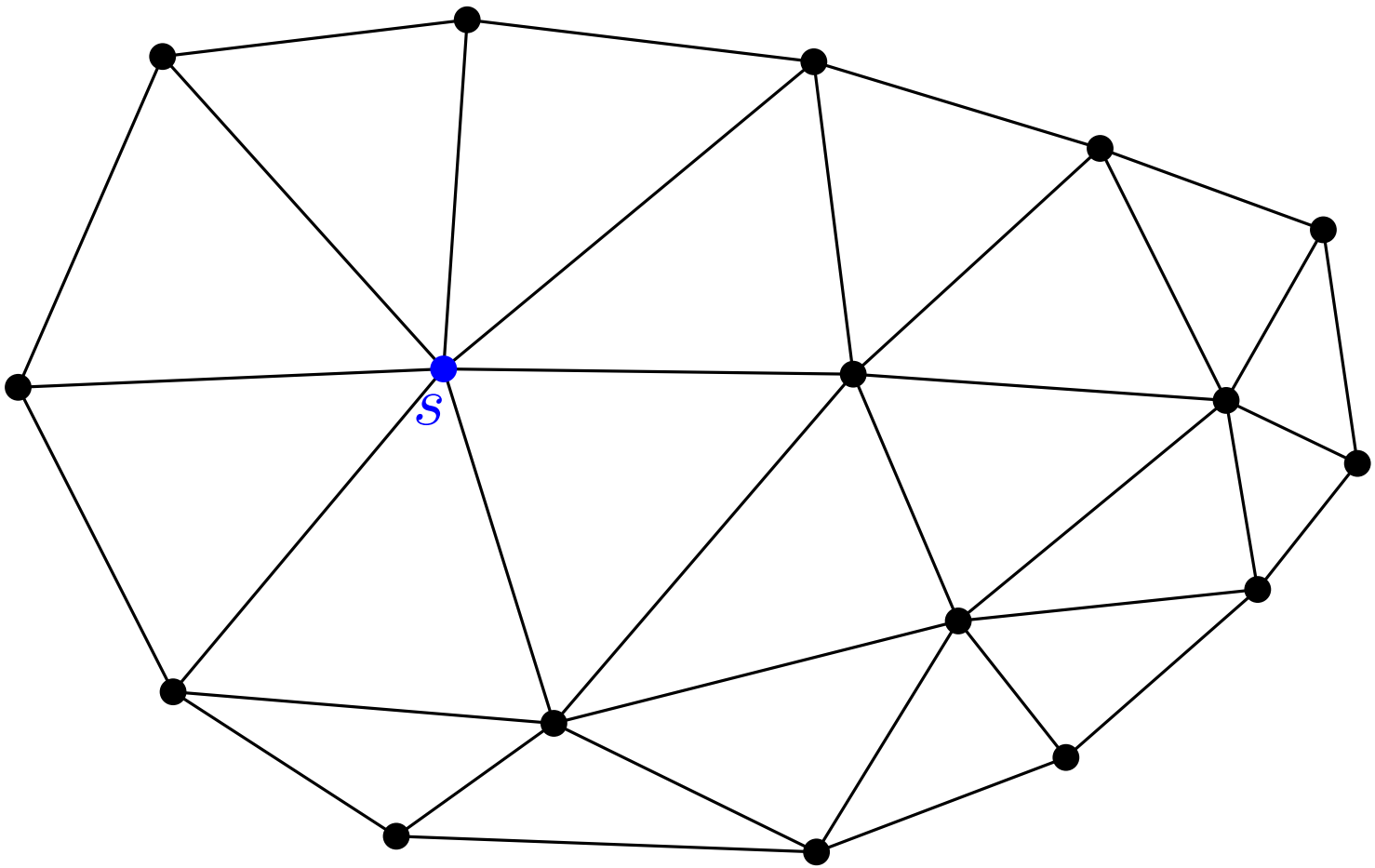
- Mitchell and Papadimitriou (1987): $O(n^8 \log(nN\rho/\epsilon))$
- Aleksandrov, Maheshwari, and Sacks:
  $O(Cn/\sqrt{\epsilon} \times \text{polylog}(n, 1/\epsilon))$,
  C depends on ρ, θ and other parameters.
- Sun and Reif: $O(C'n/\epsilon \times \text{polylog}(n, 1/\epsilon))$,
  C' depends on θ, other parameters, but not ρ.
- Cheng et al. (2007): $O(n^3\rho/\epsilon \times \text{polylog}(n, \rho, 1/\epsilon))$

Query problem



Query problem

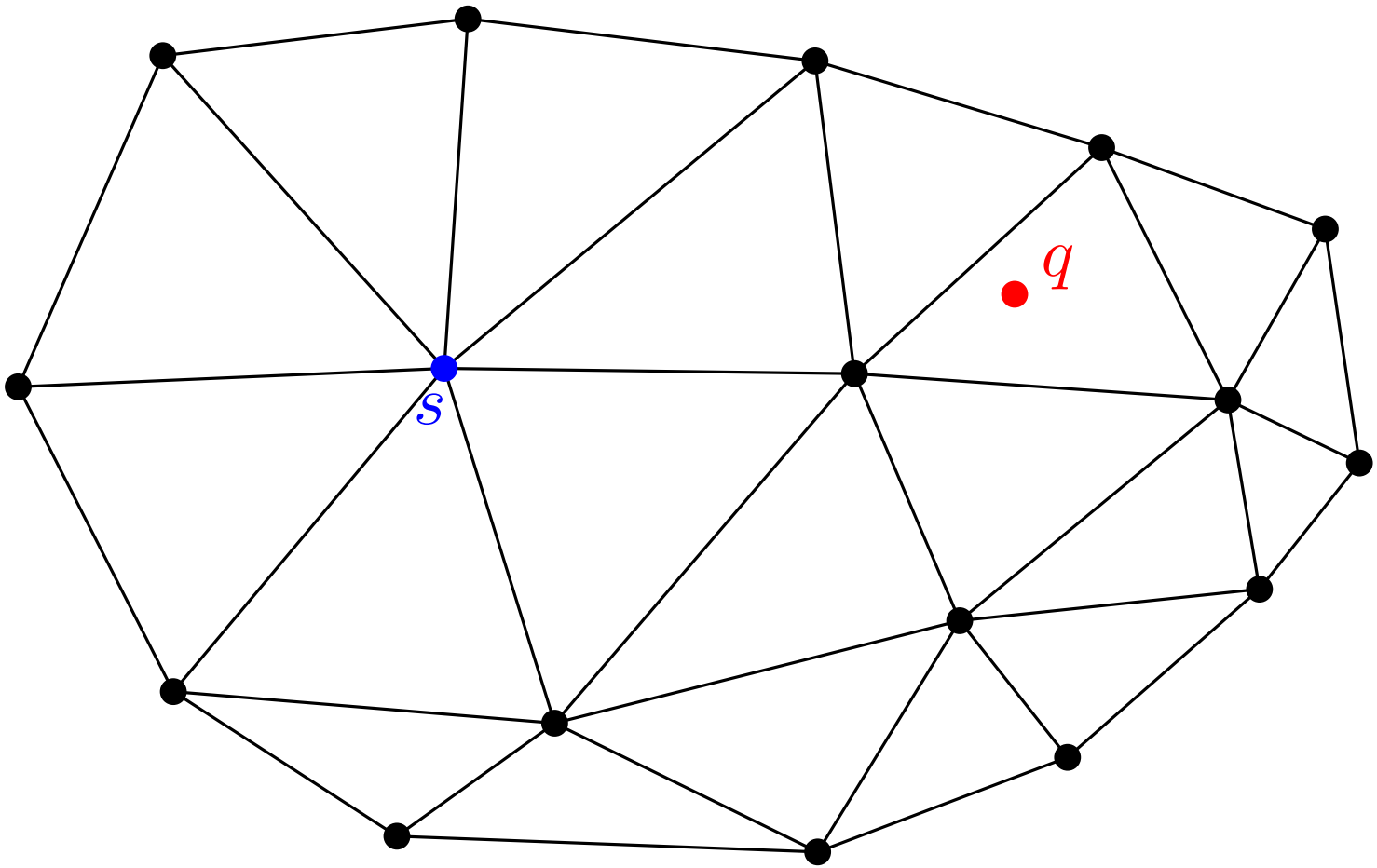
Build a data structure



Query problem

Build a data structure

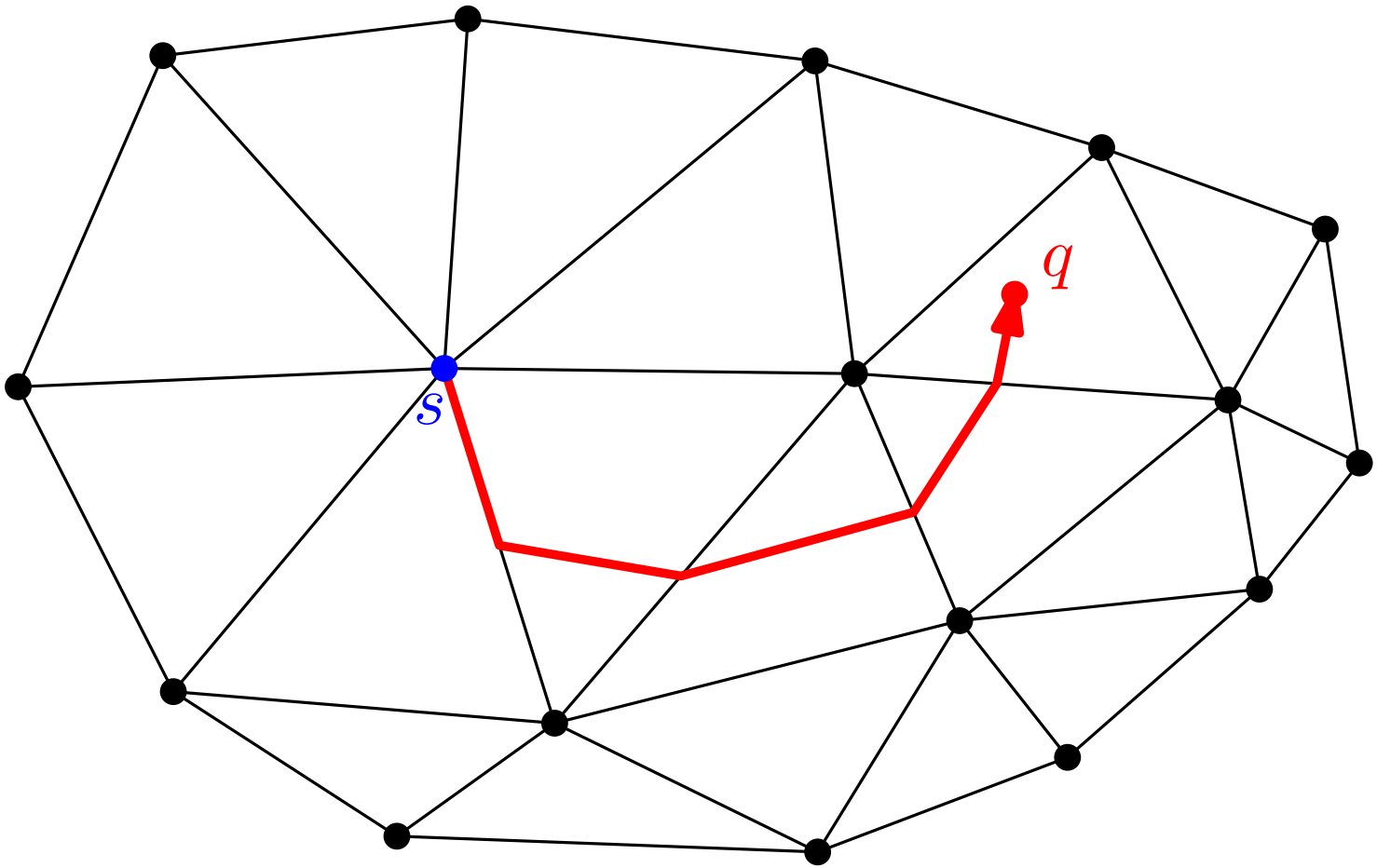
Given a query point q, find a $(1+\varepsilon)$-approximation of the cost of the shortest path from s to q.



Query problem

Build a data structure

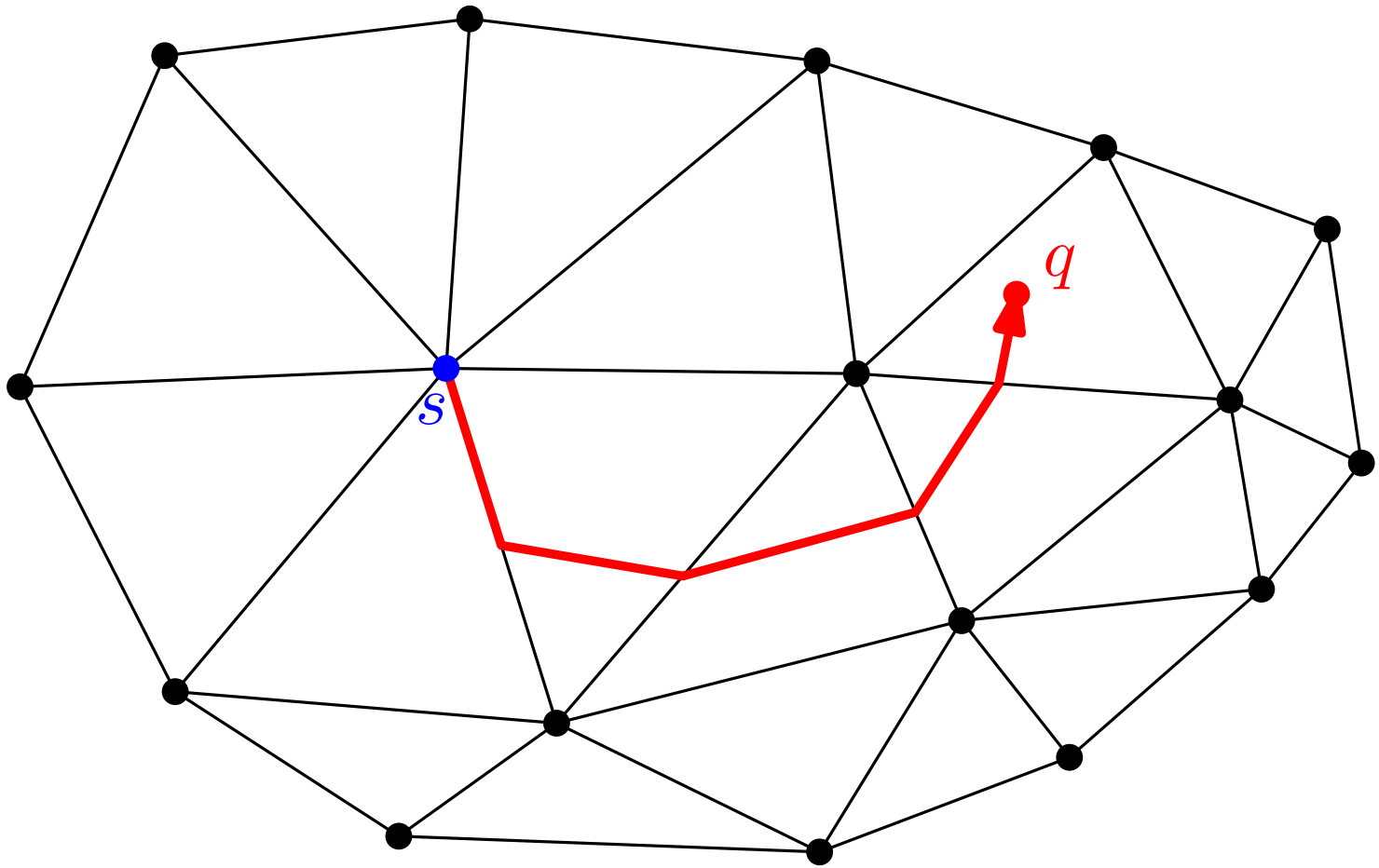
Given a query point q, find a $(1+\varepsilon)$-approximation of the cost of the shortest path from s to q.



Query problem

Build a data structure

Given a query point q, find a $(1+\varepsilon)$-approximation of the cost of the shortest path from s to q.



Output an approximate shortest path in time linear in its size

Previous results

Aleksandrov et al.: Space and preprocessing time depend on several parameters including n, ε, ρ and θ.

Previous results

Aleksandrov et al.: Space and preprocessing time depend on several parameters including n, ε, ρ and θ.

Applies to polyhedral surfaces

Previous results

Aleksandrov et al.: Space and preprocessing time depend on several parameters including n, ε, ρ and θ.

Applies to polyhedral surfaces

Our results

Query time: $O(\log(\rho n/\varepsilon))$

Space: $O(\rho^2 n^4/\varepsilon^2 \log(\rho n/\varepsilon))$

Preprocessing time: $O(\rho^2 n^4/\varepsilon^2 \log^2(\rho n/\varepsilon))$

Previous results

Aleksandrov et al.: Space and preprocessing time depend on several parameters including n, ε, ρ and θ.

Applies to polyhedral surfaces

Our results

Query time: $O(\log(\rho n/\varepsilon))$

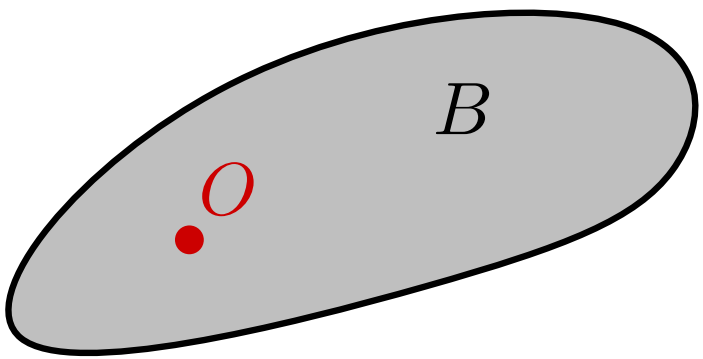
Space: $O(\rho^2 n^4/\varepsilon^2 \log(\rho n/\varepsilon))$

Preprocessing time: $O(\rho^2 n^4/\varepsilon^2 \log^2(\rho n/\varepsilon))$

Applies to anisotropic distance functions

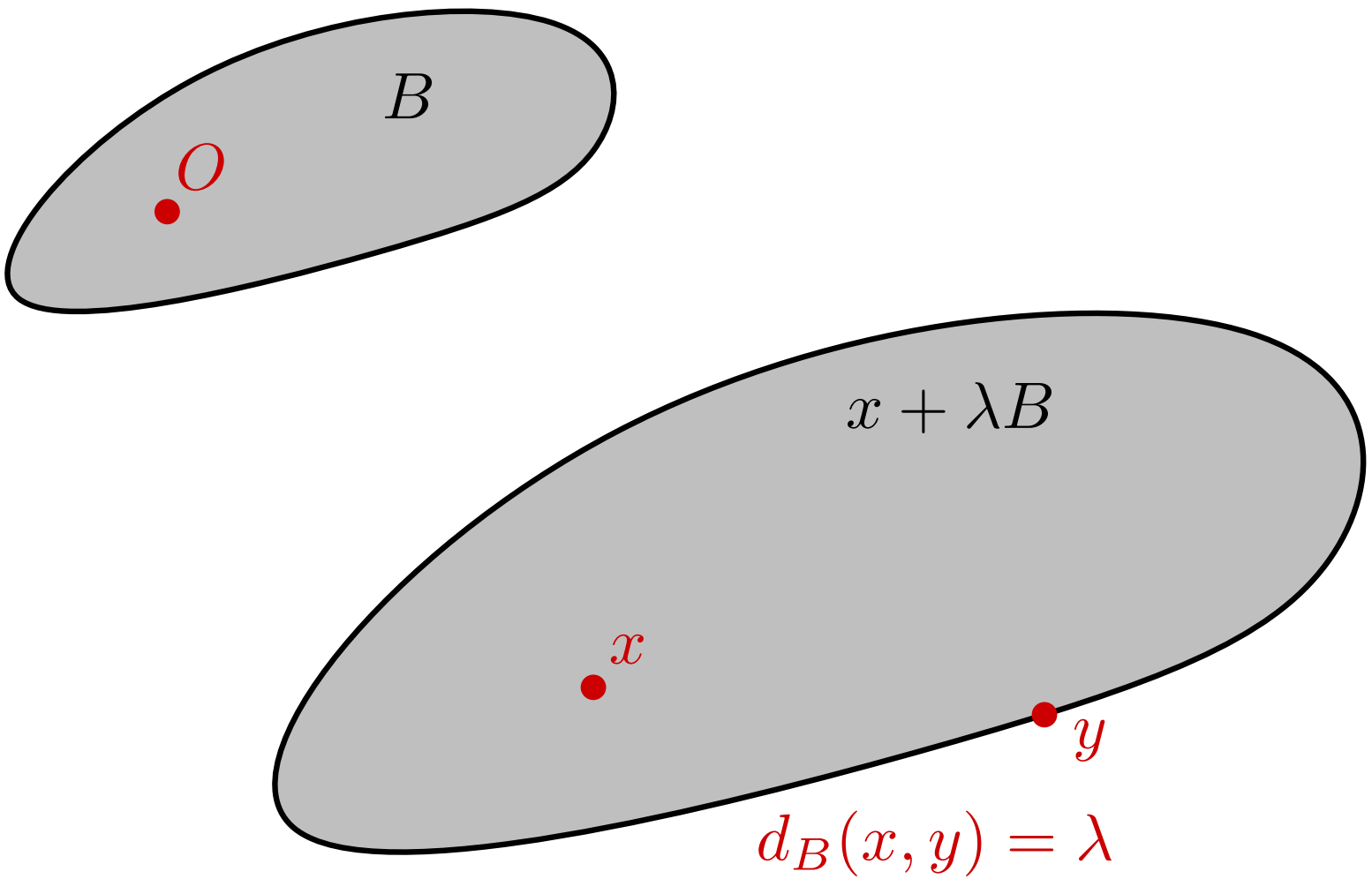
Convex distance function

Unit ball B: set of points at distance $\leqslant 1$ from O



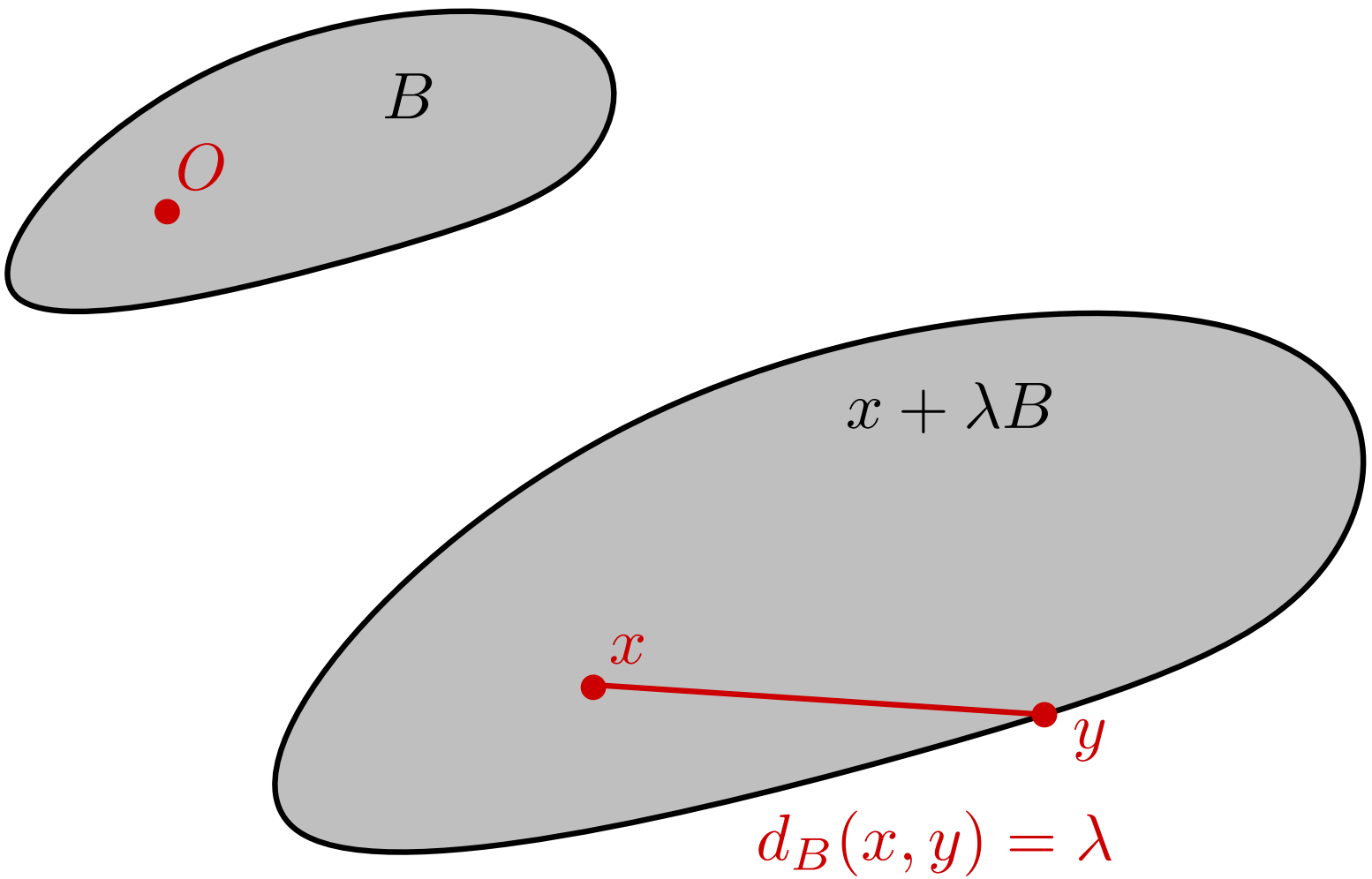
Convex distance function

Unit ball B: set of points at distance $\leqslant 1$ from O



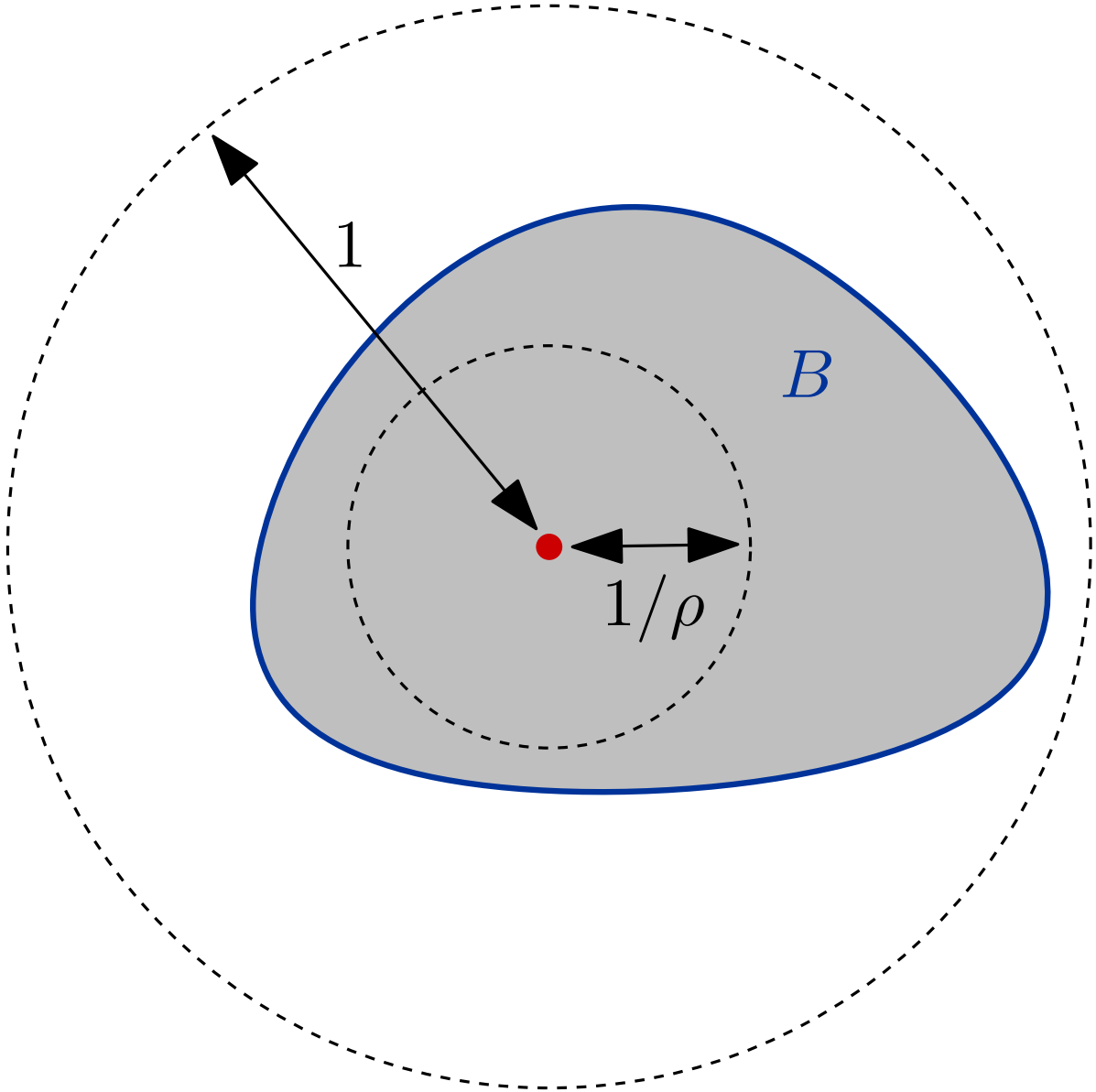
Convex distance function

Unit ball B: set of points at distance $\leqslant 1$ from O



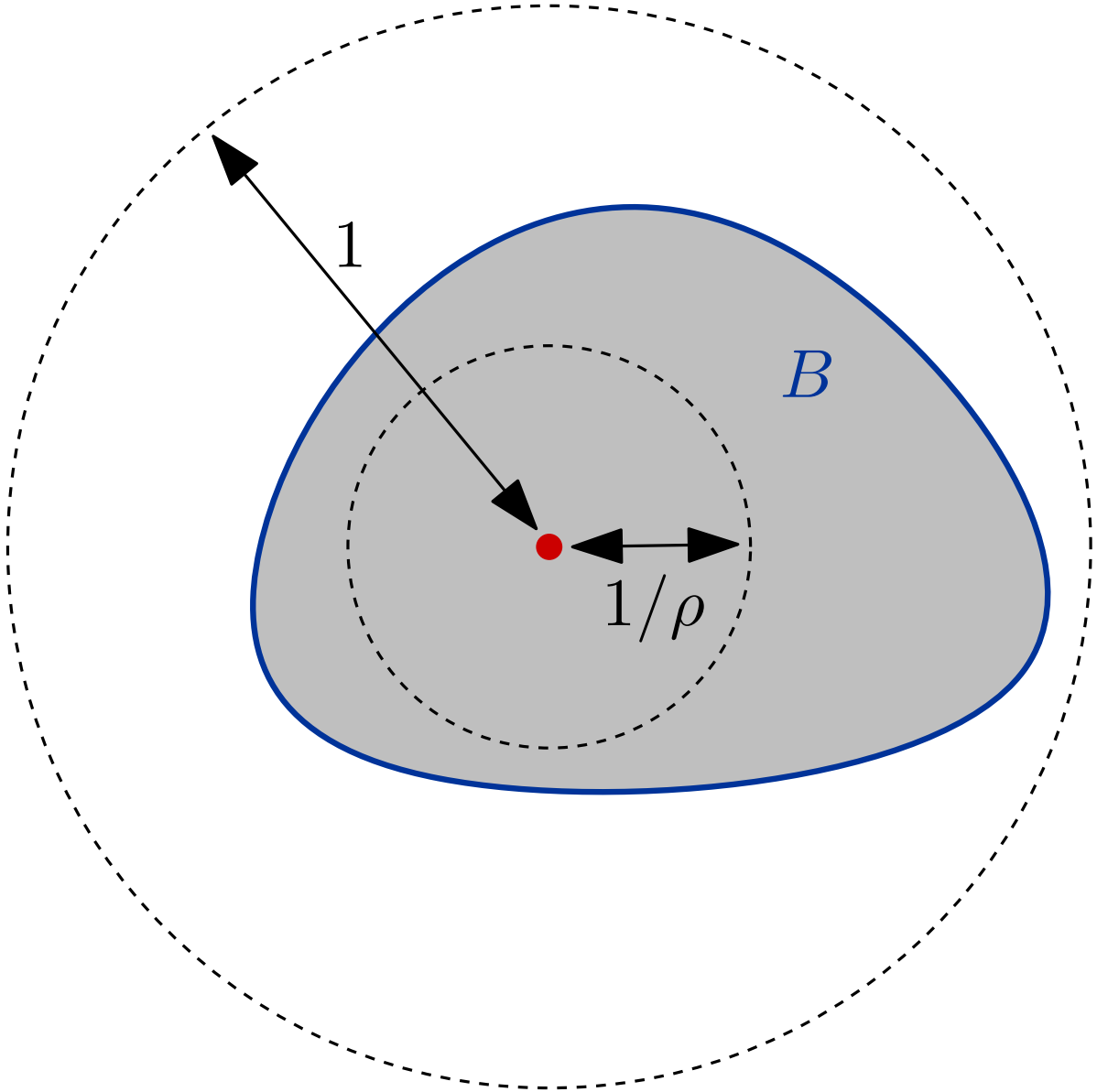
Shortest path is a straight line segment

Model



One convex distance function per face

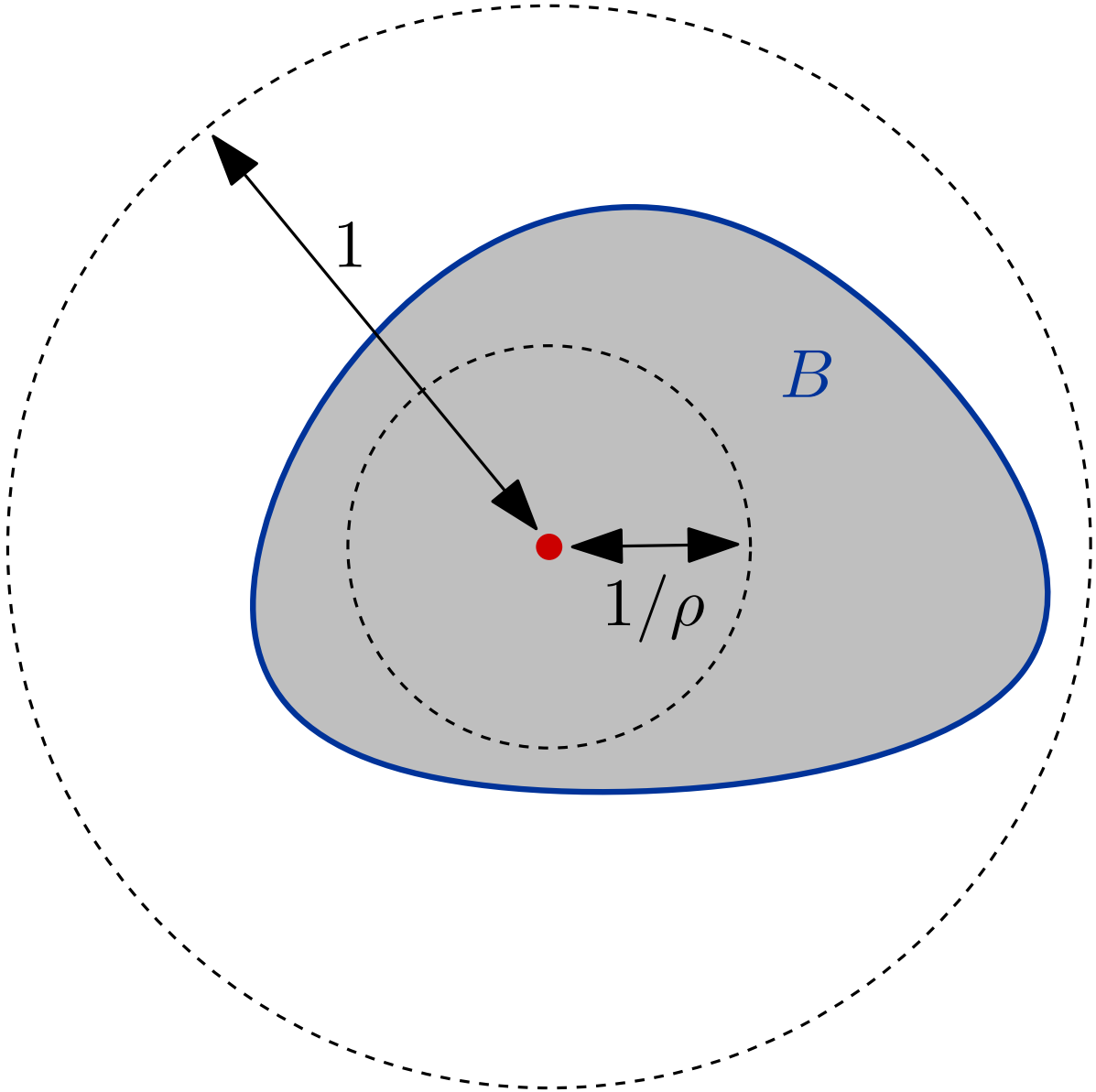
Model



One convex distance function per face

Speed is in interval $[1/\rho, 1]$

Model

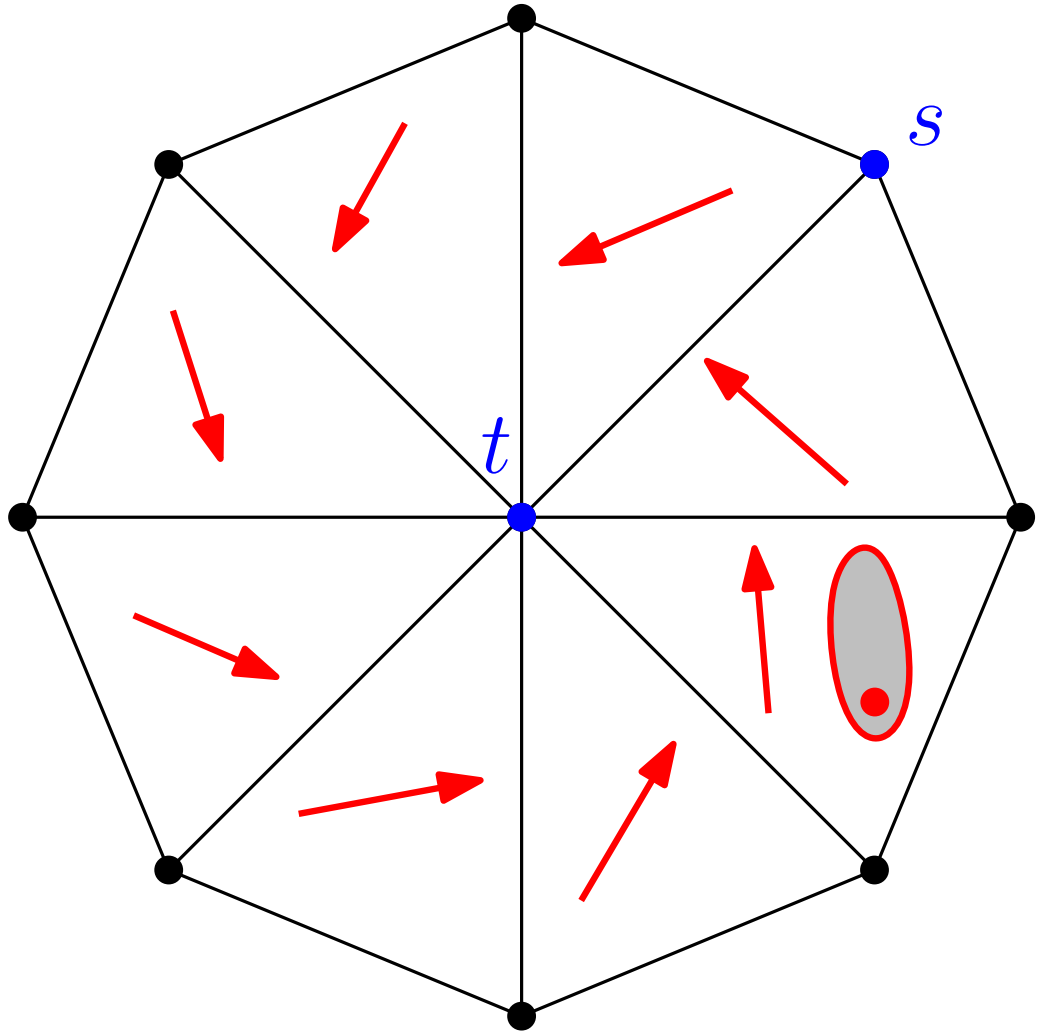


One convex distance function per face

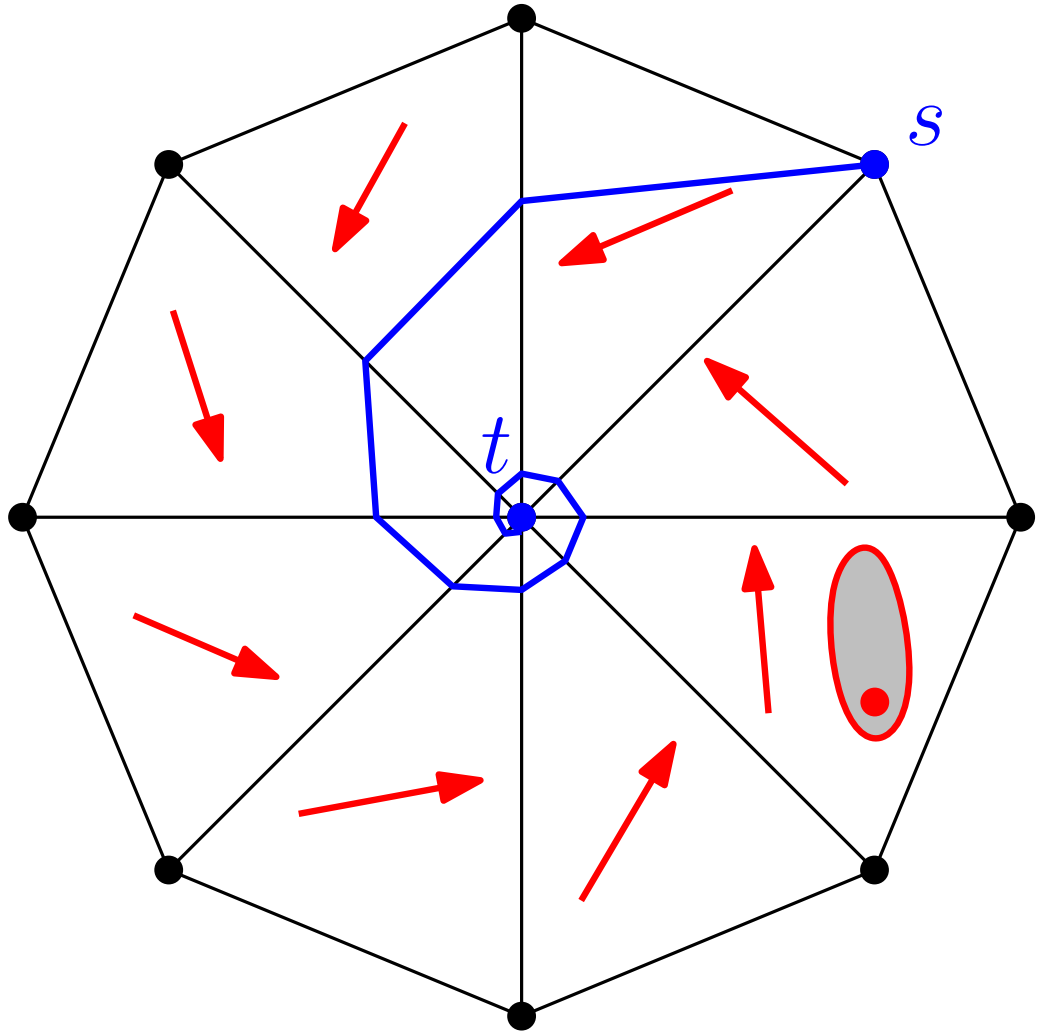
Speed is in interval $[1/\rho, 1]$

Cost of a length ℓ path is in $[\ell, \rho\ell]$.

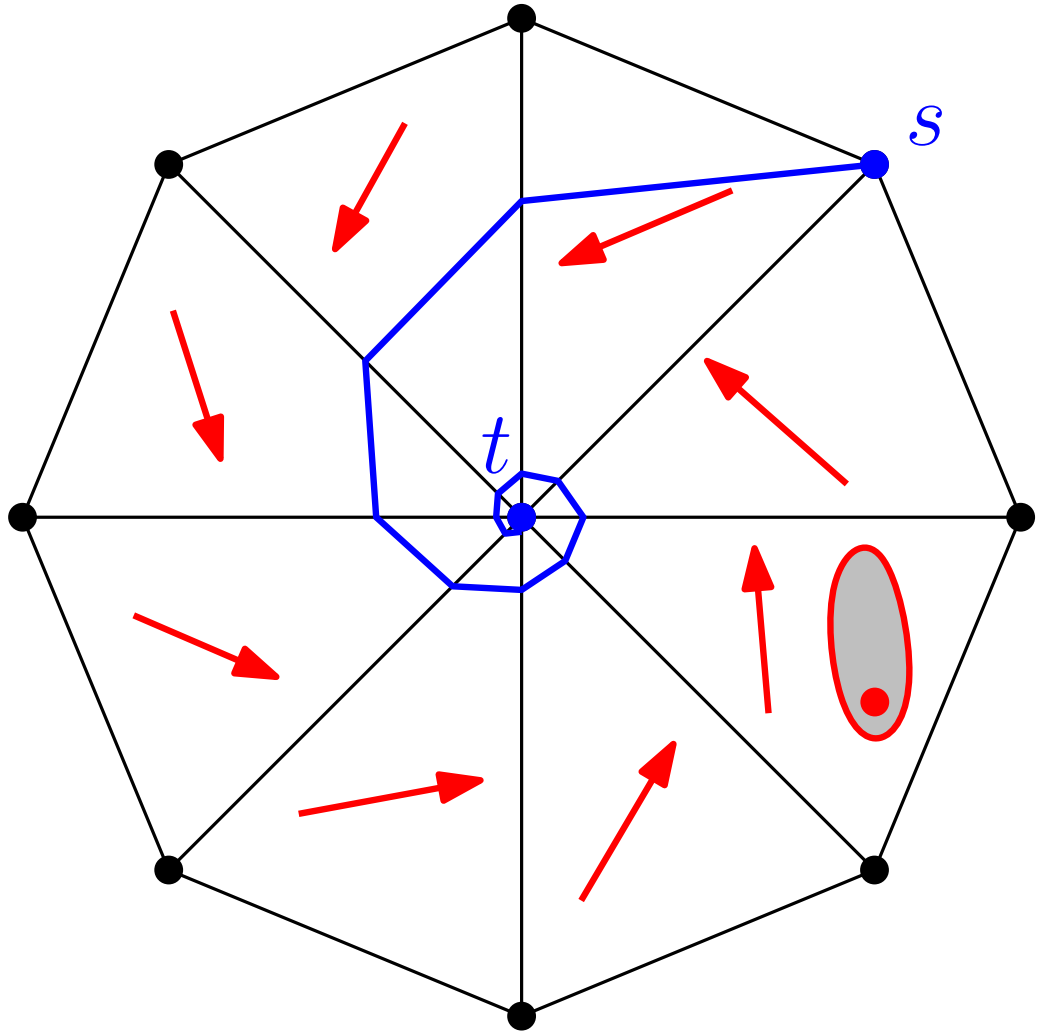
Exemple of a non-polygonal shortest path



Exemple of a non-polygonal shortest path

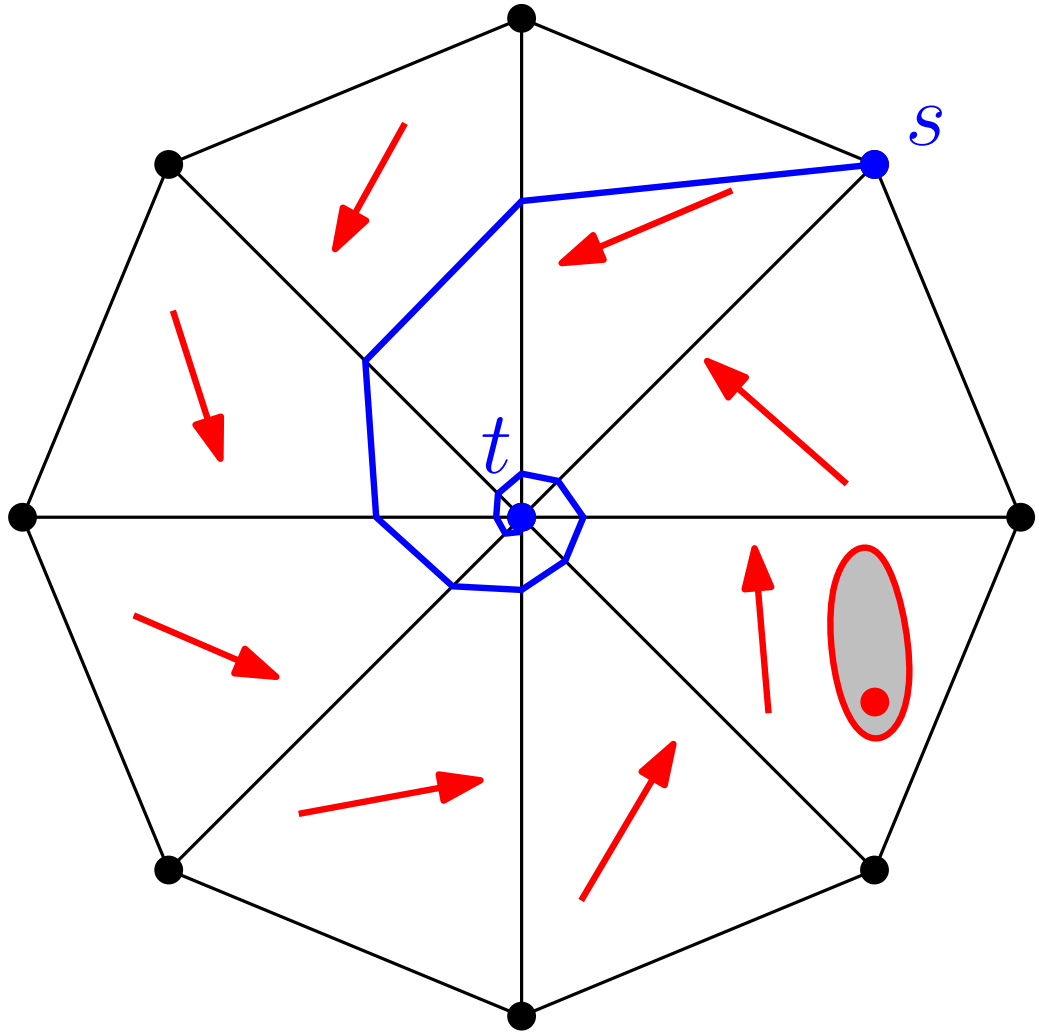


Exemple of a non-polygonal shortest path



There exists a rectifiable (=finite length) shortest path

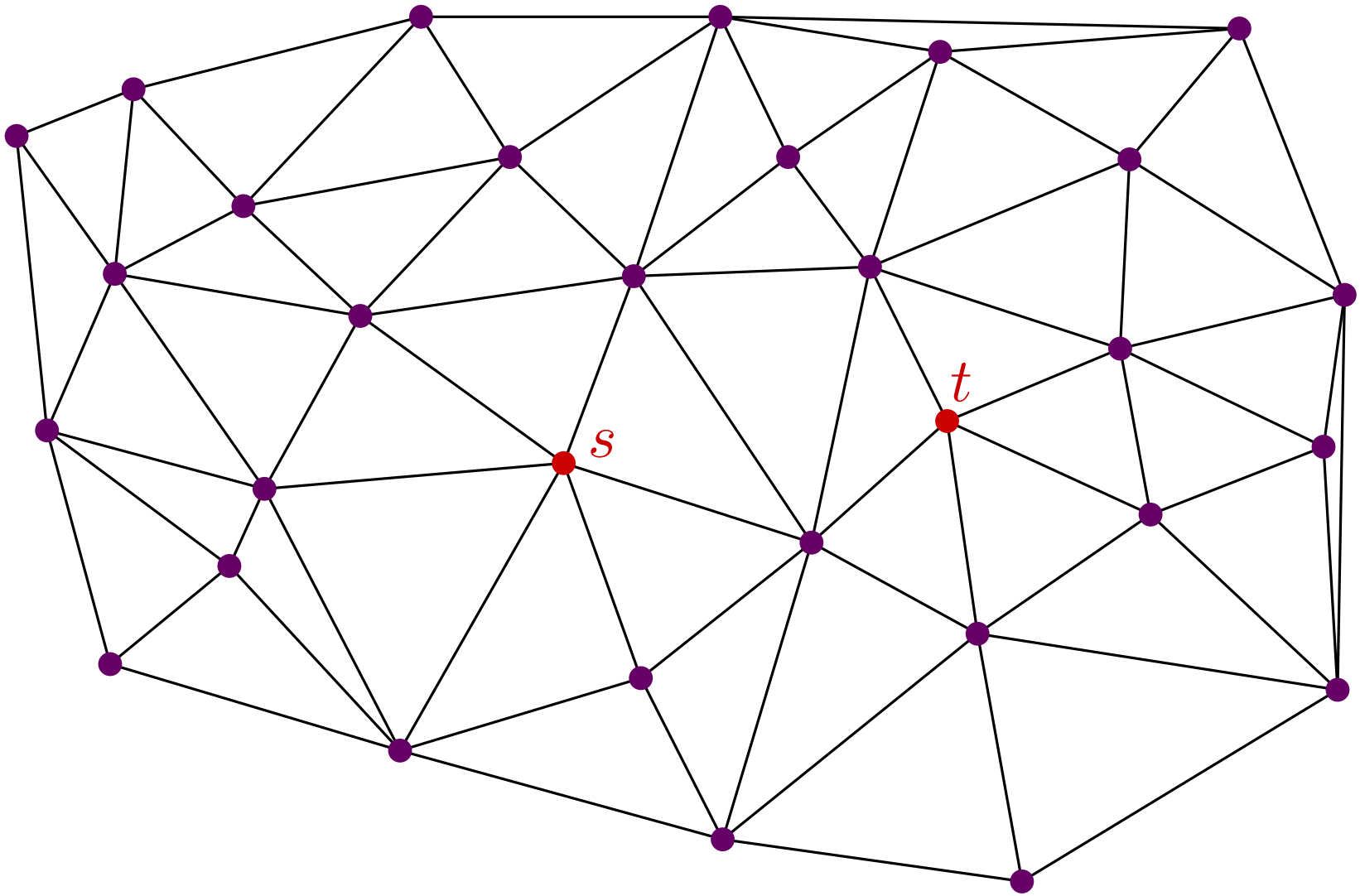
Exemple of a non-polygonal shortest path



There exists a rectifiable (=finite length) shortest path

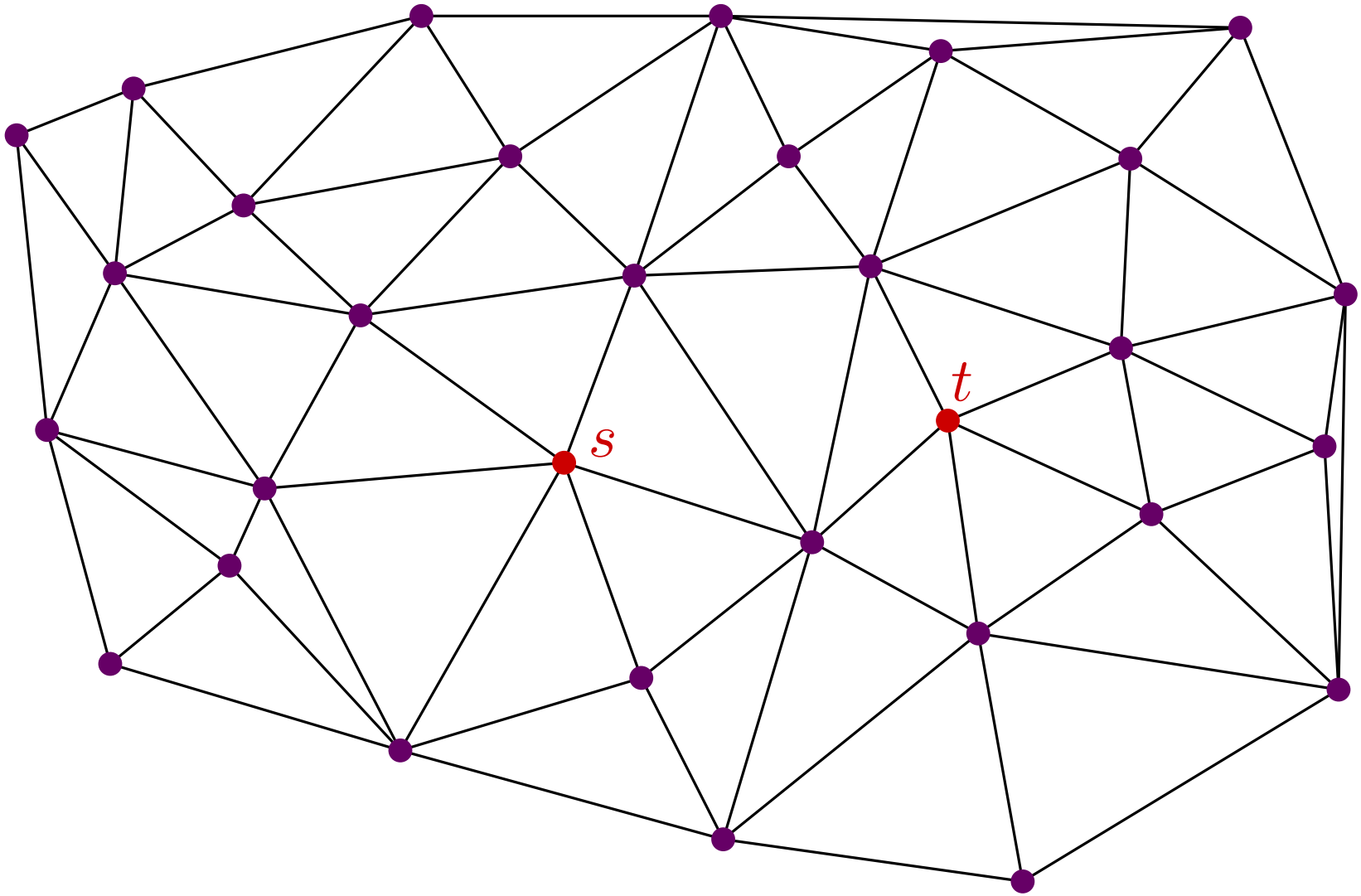
There exists a $(1+\varepsilon)$-approximate shortest path with $O(\rho n^2/\varepsilon)$ edges.

Static (non query) algorithm



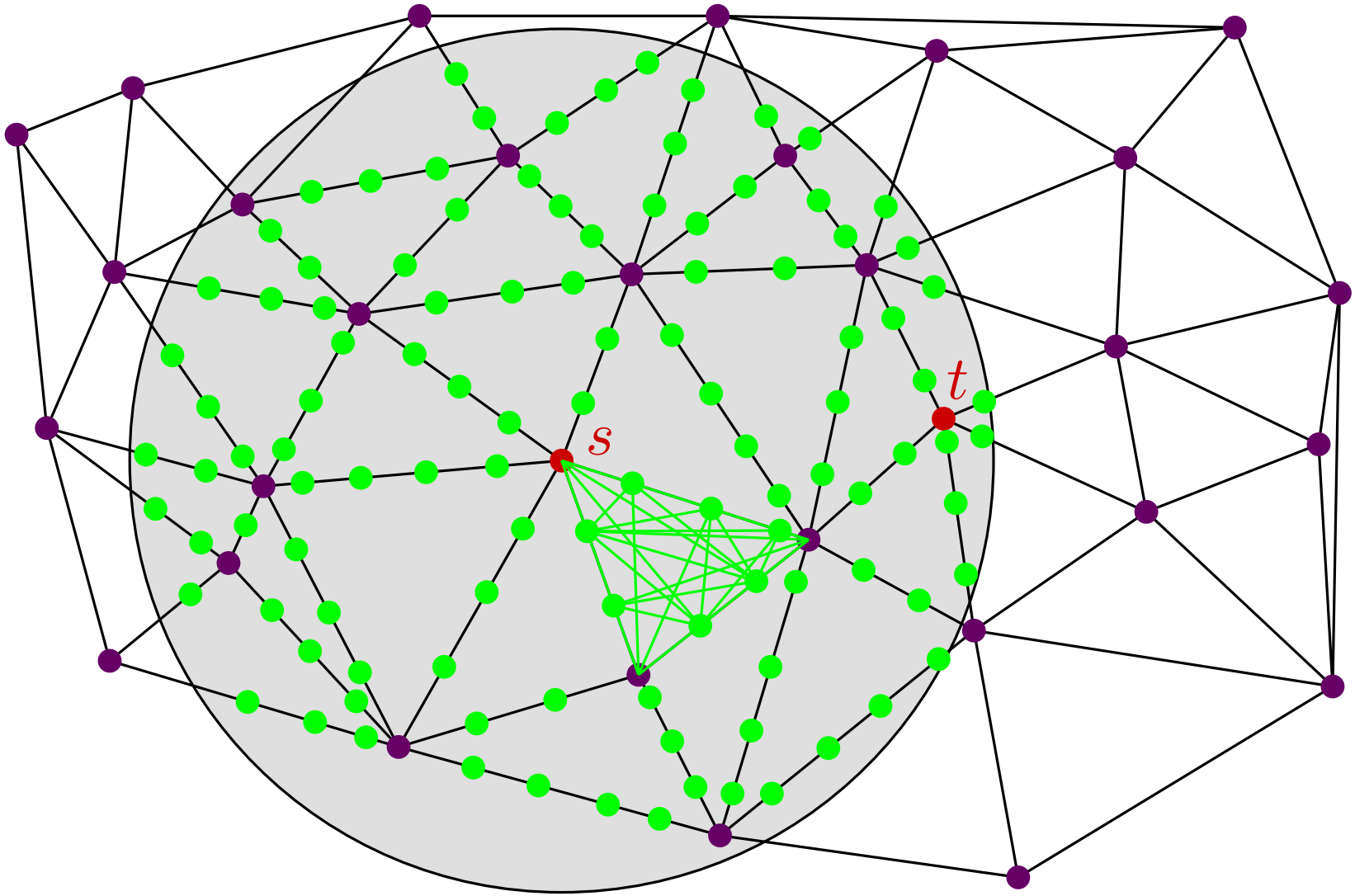
Static (non query) algorithm

Circle with radius $\rho\ d(s,t)$ and center s



Static (non query) algorithm

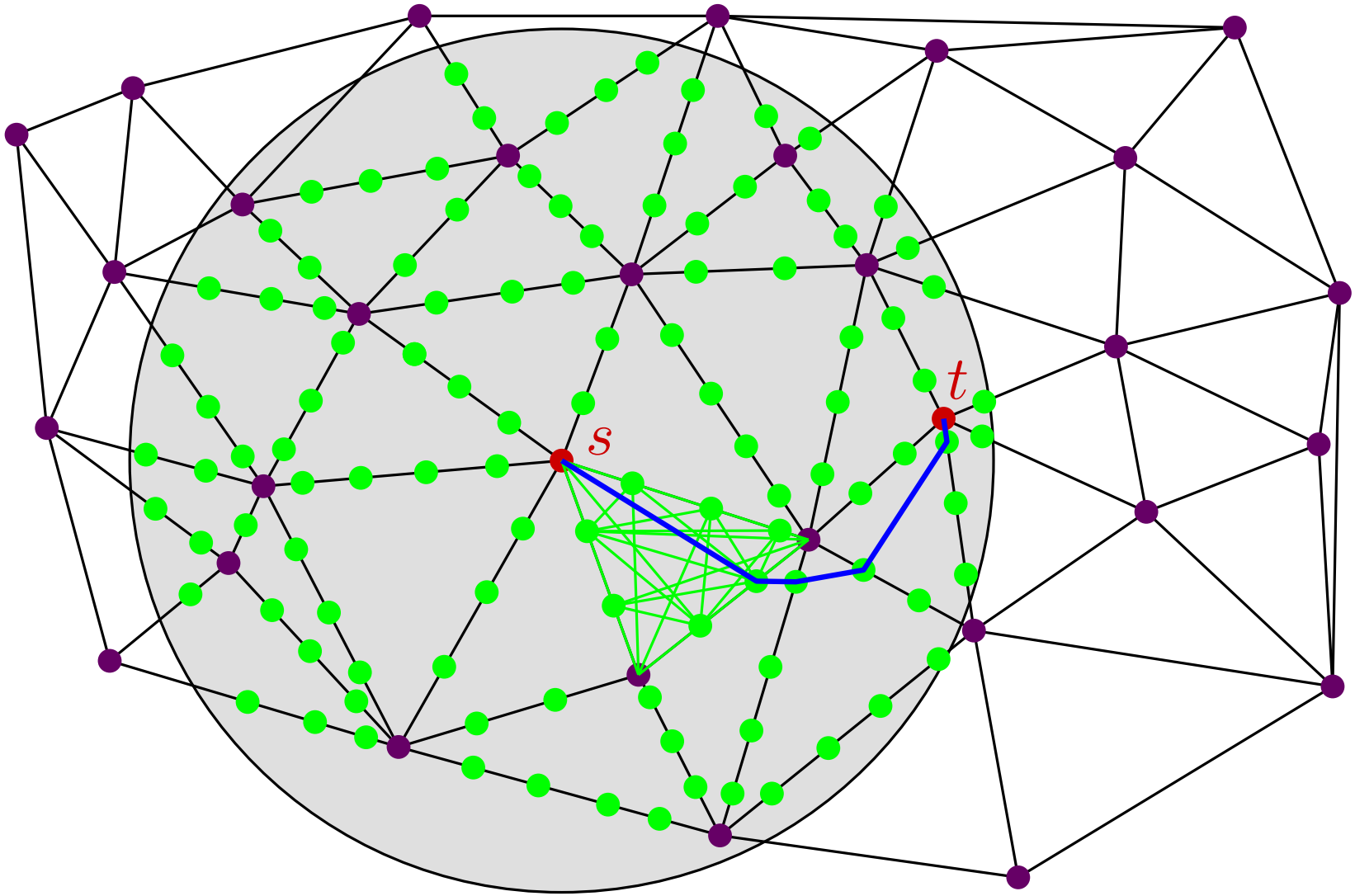
Circle with radius $\rho\ d(s,t)$ and center s



Steiner points with uniform spacing $\delta = 1/\mathrm{poly}(\rho, n, 1/\varepsilon)$

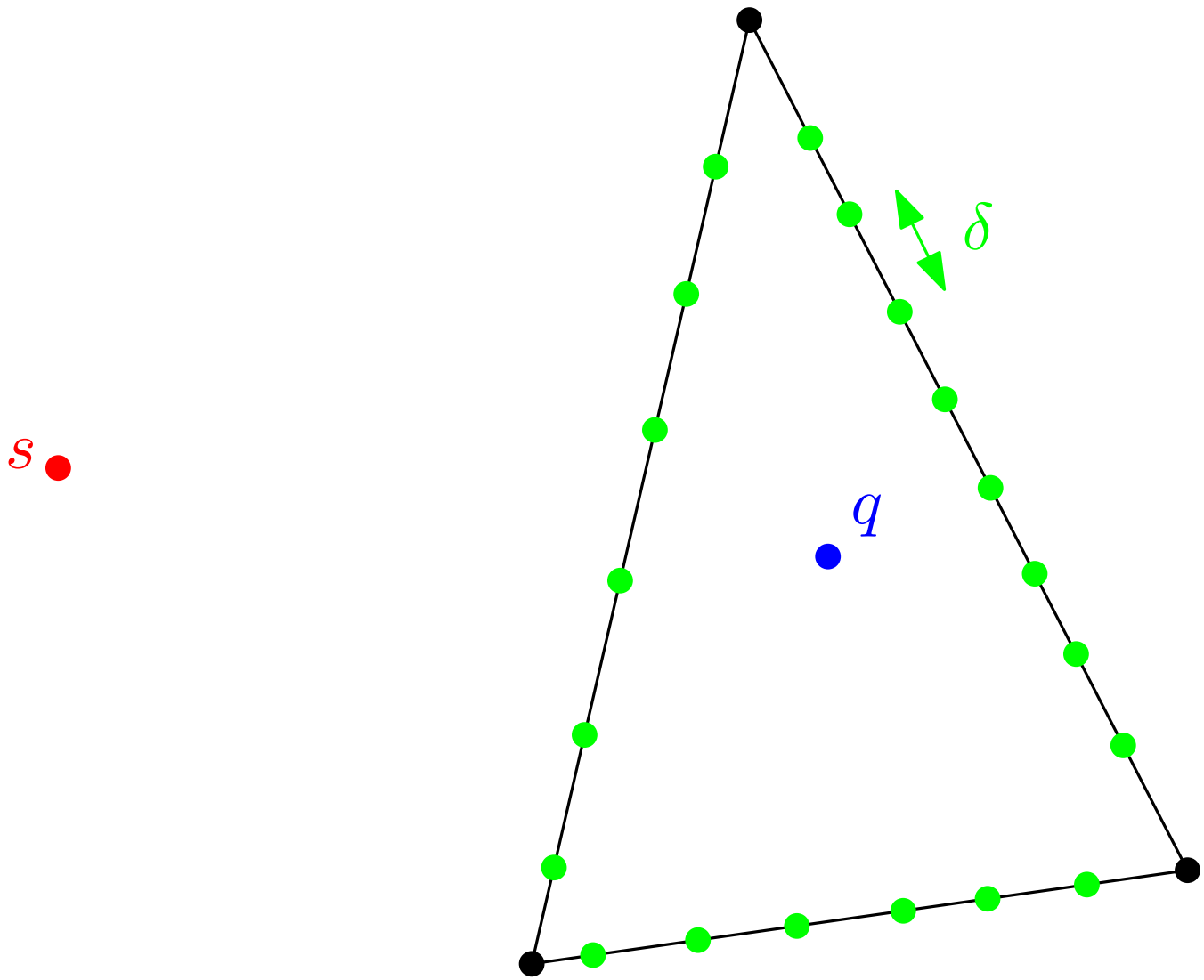
Static (non query) algorithm

Circle with radius $\rho\ d(s,t)$ and center s

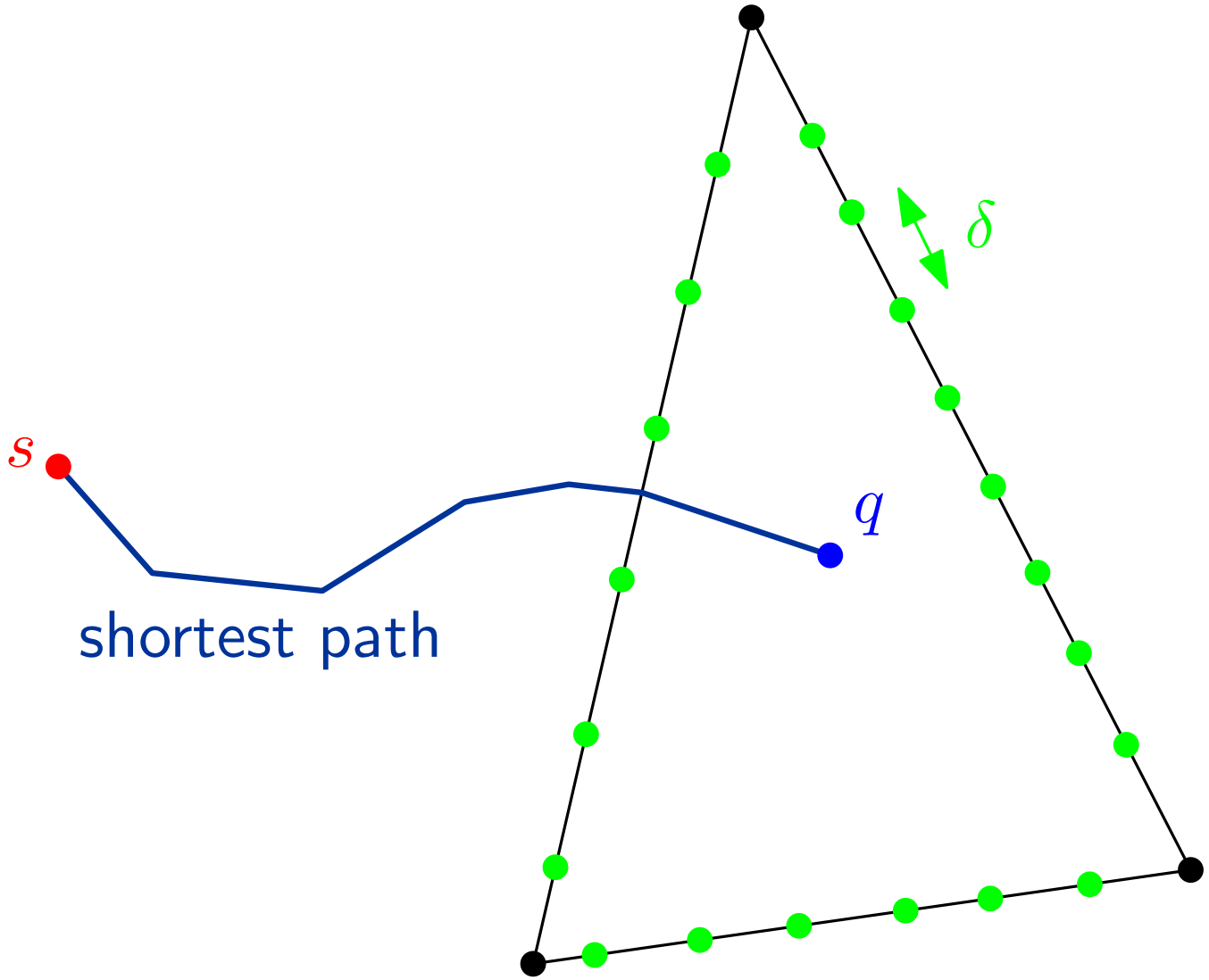


Steiner points with uniform spacing $\delta = 1/\mathrm{poly}(\rho, n, 1/\varepsilon)$

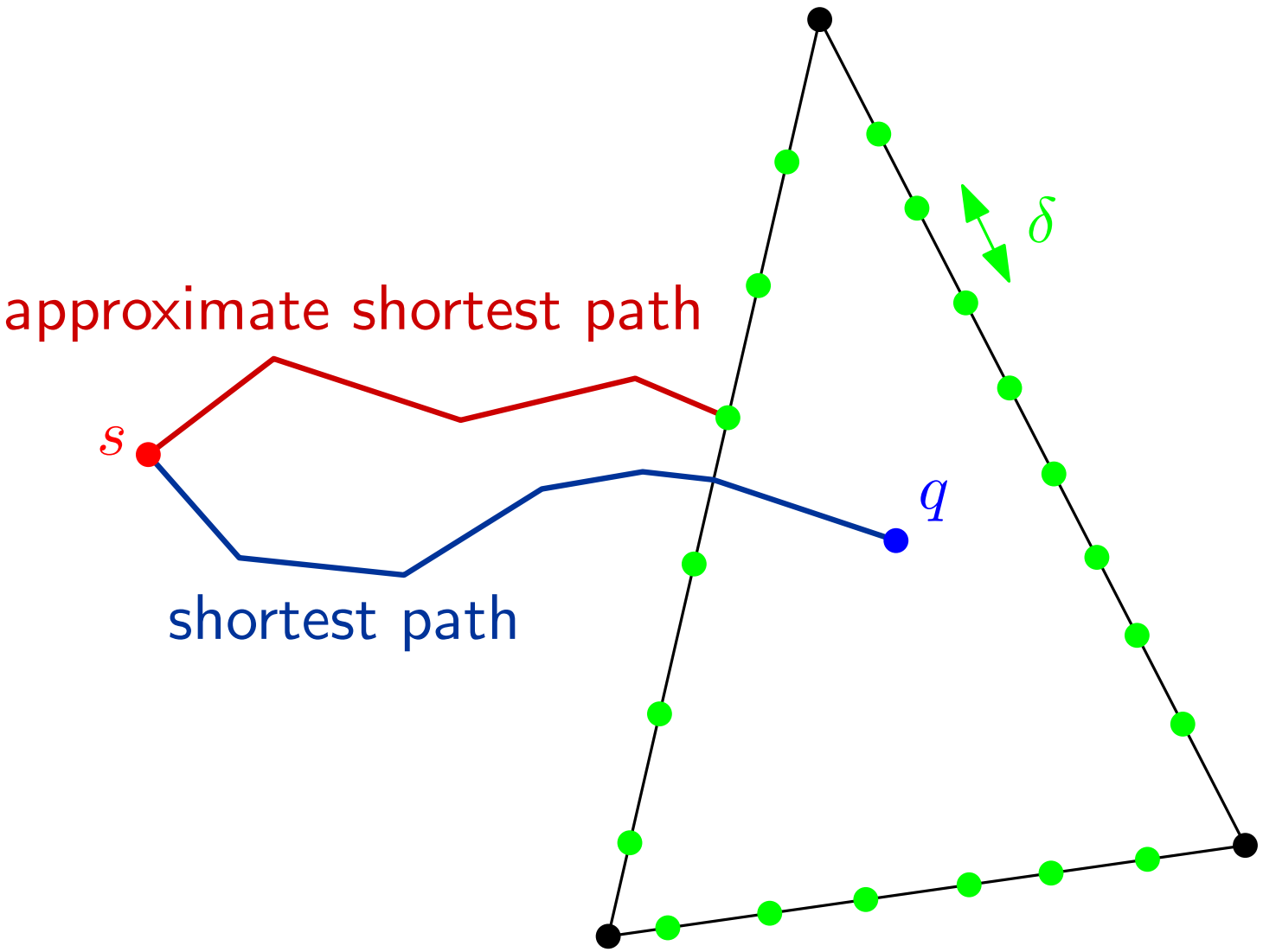
Observation



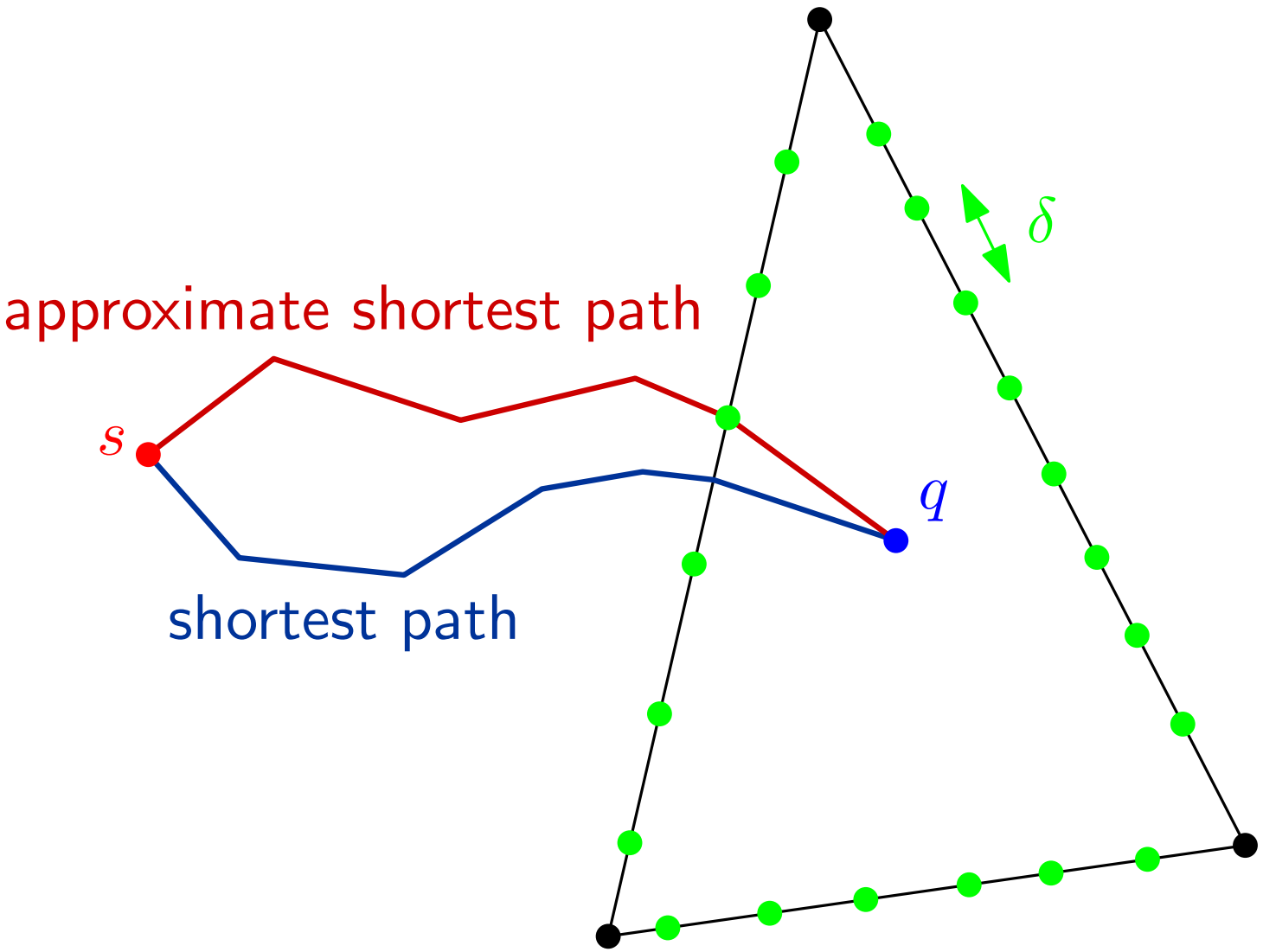
Observation



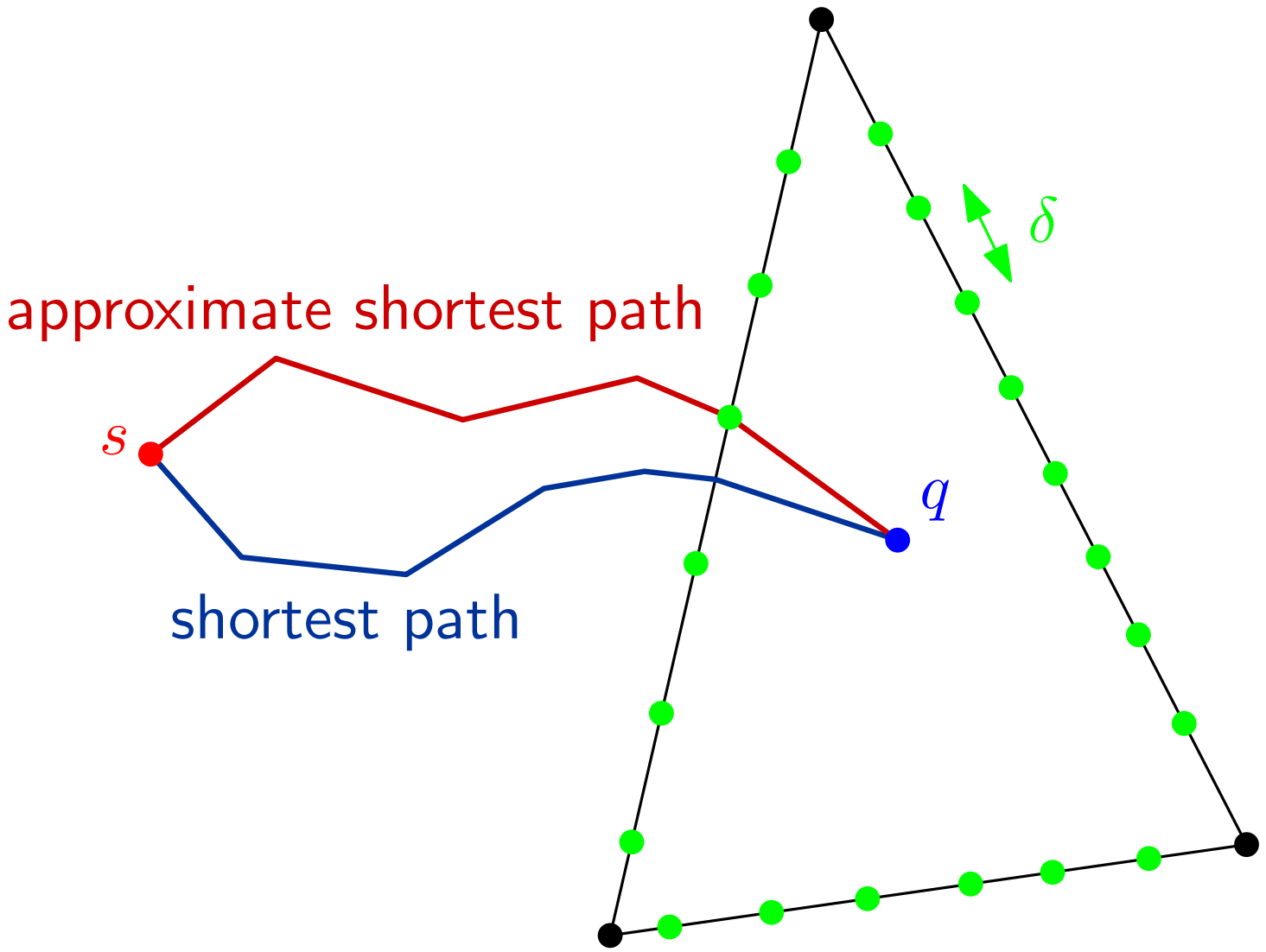
Observation



Observation



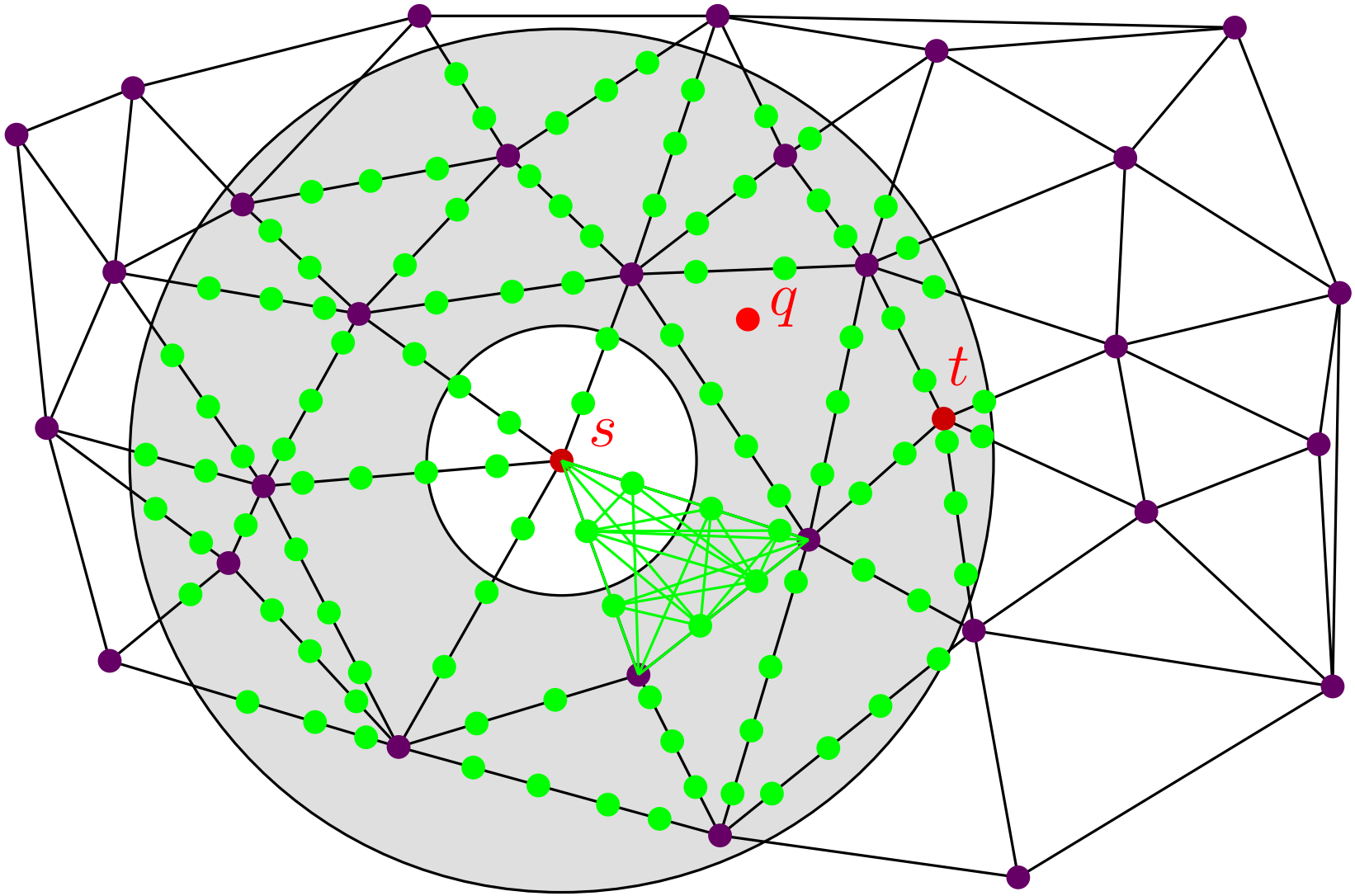
Observation



additive error at most δ

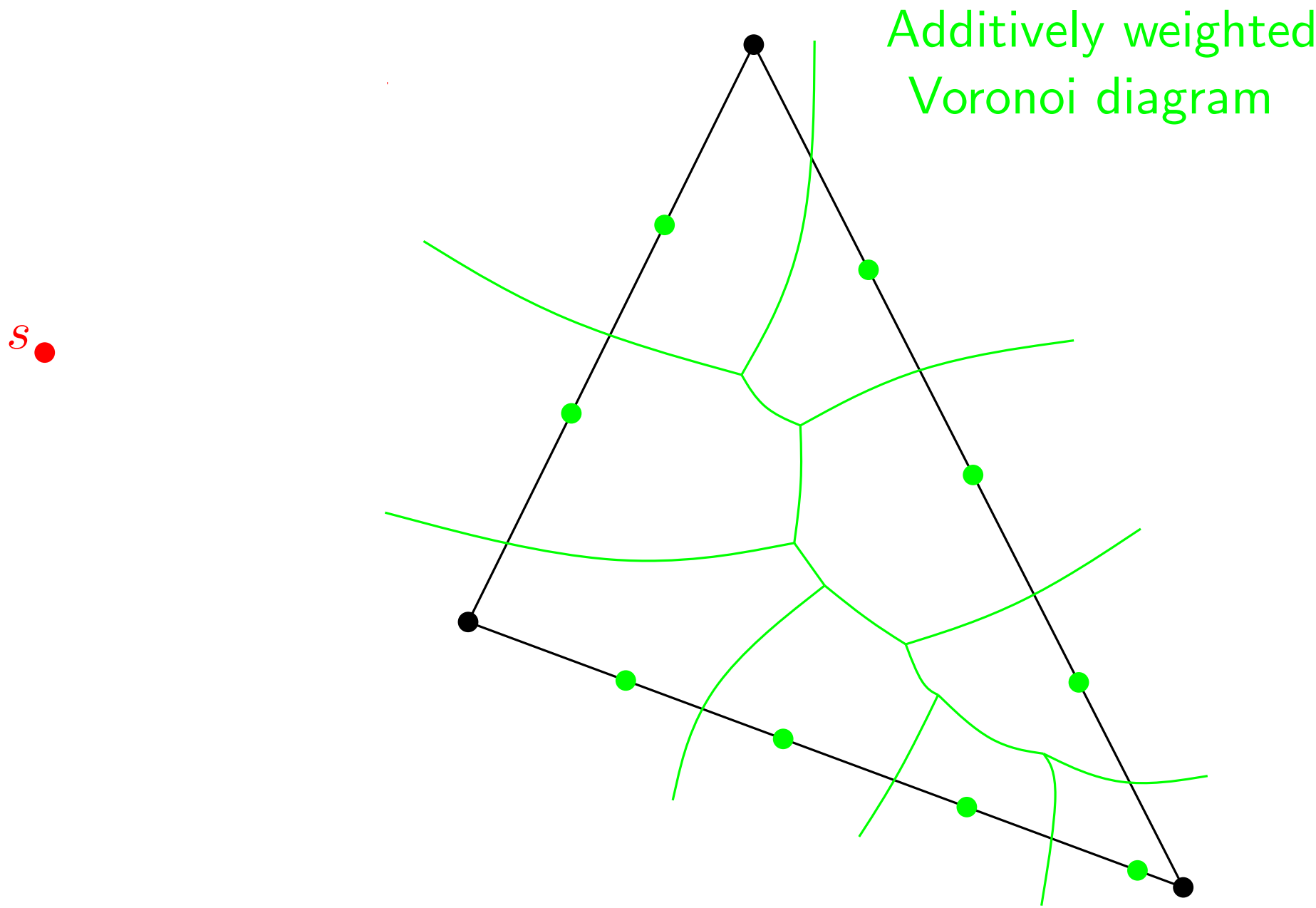
Observation

$(1+\varepsilon)$ approximation when q is in an annulus

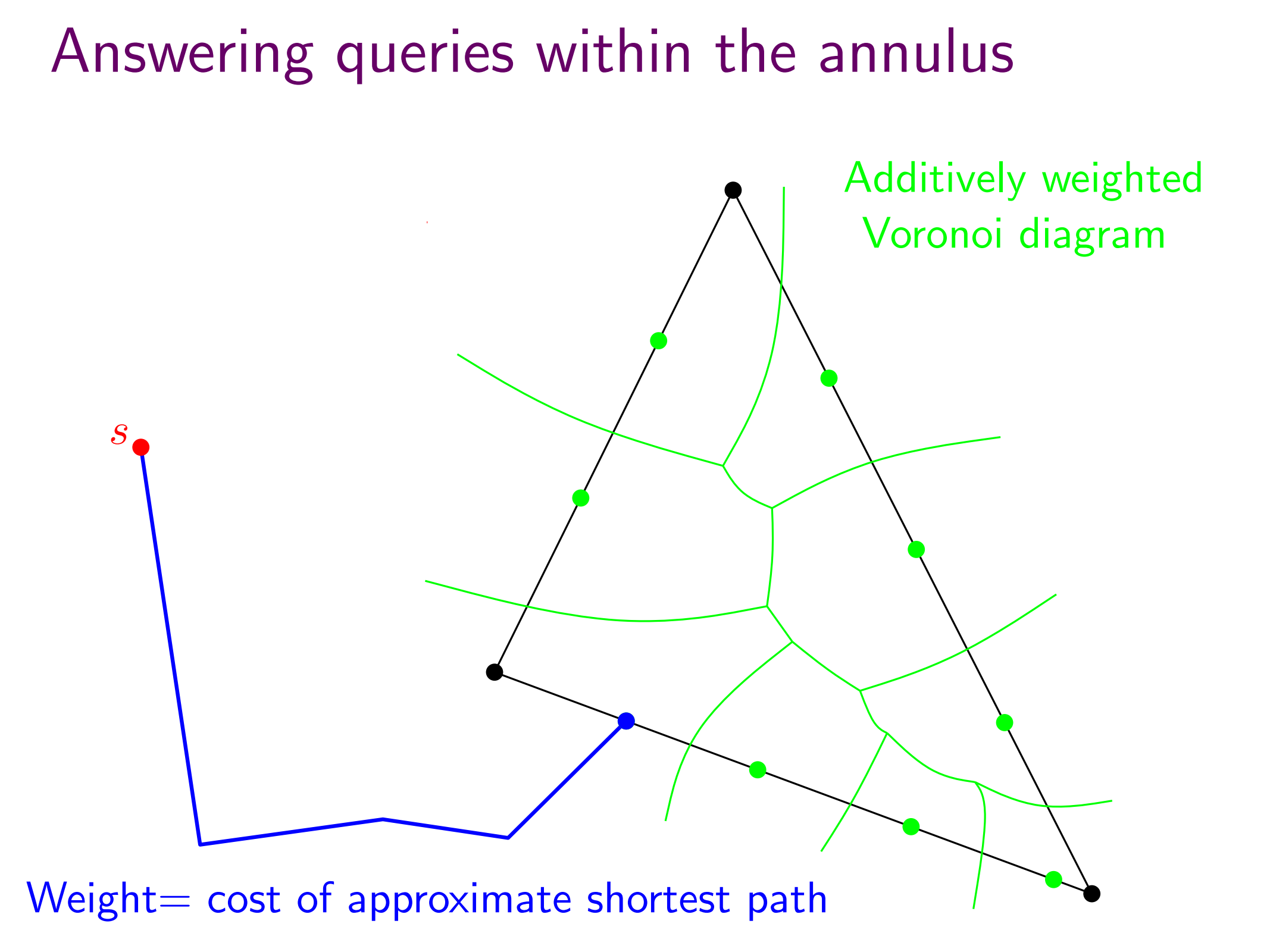


radius ratio=poly$(\rho, n, 1/\varepsilon)$

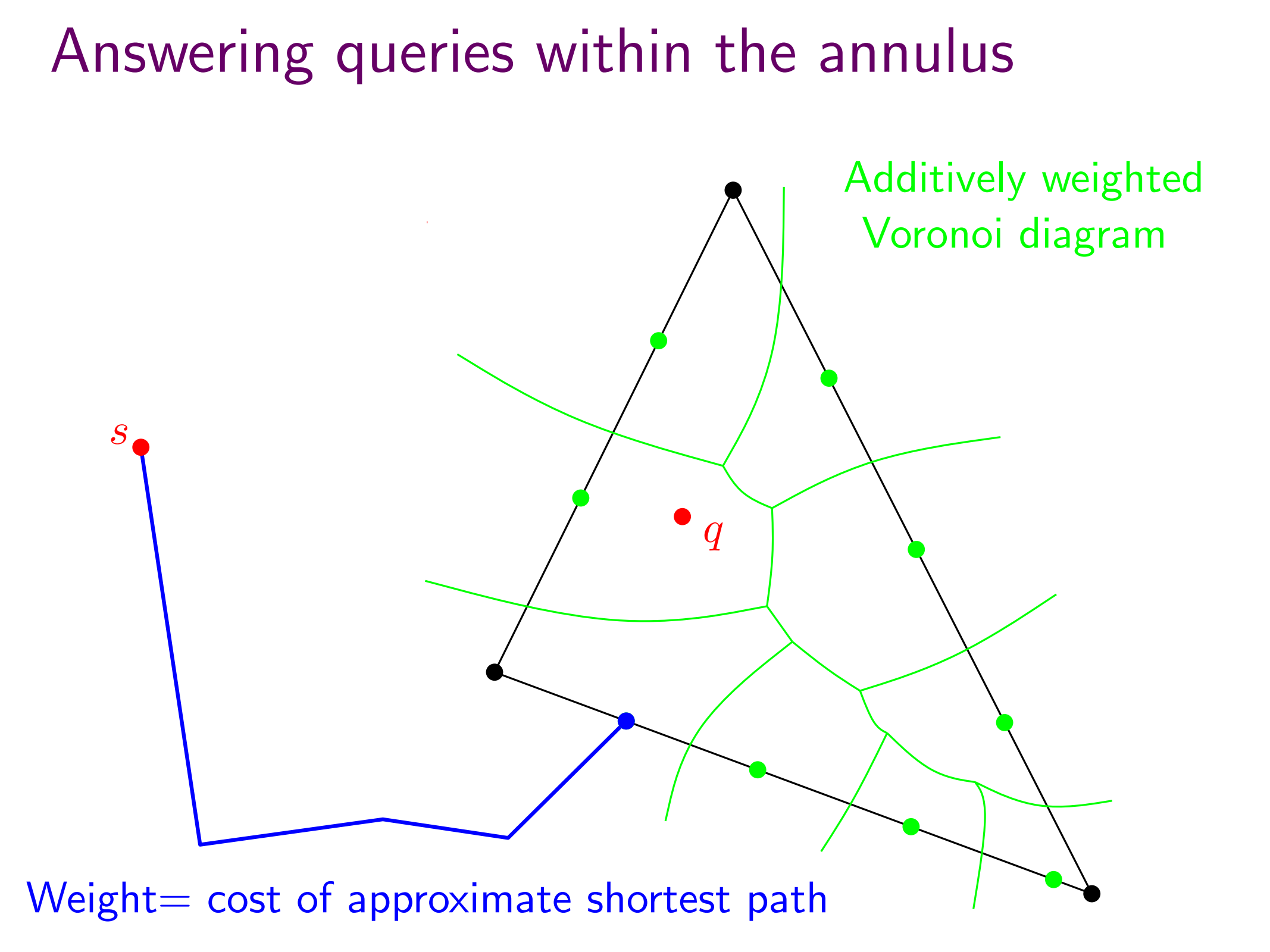
Answering queries within the annulus



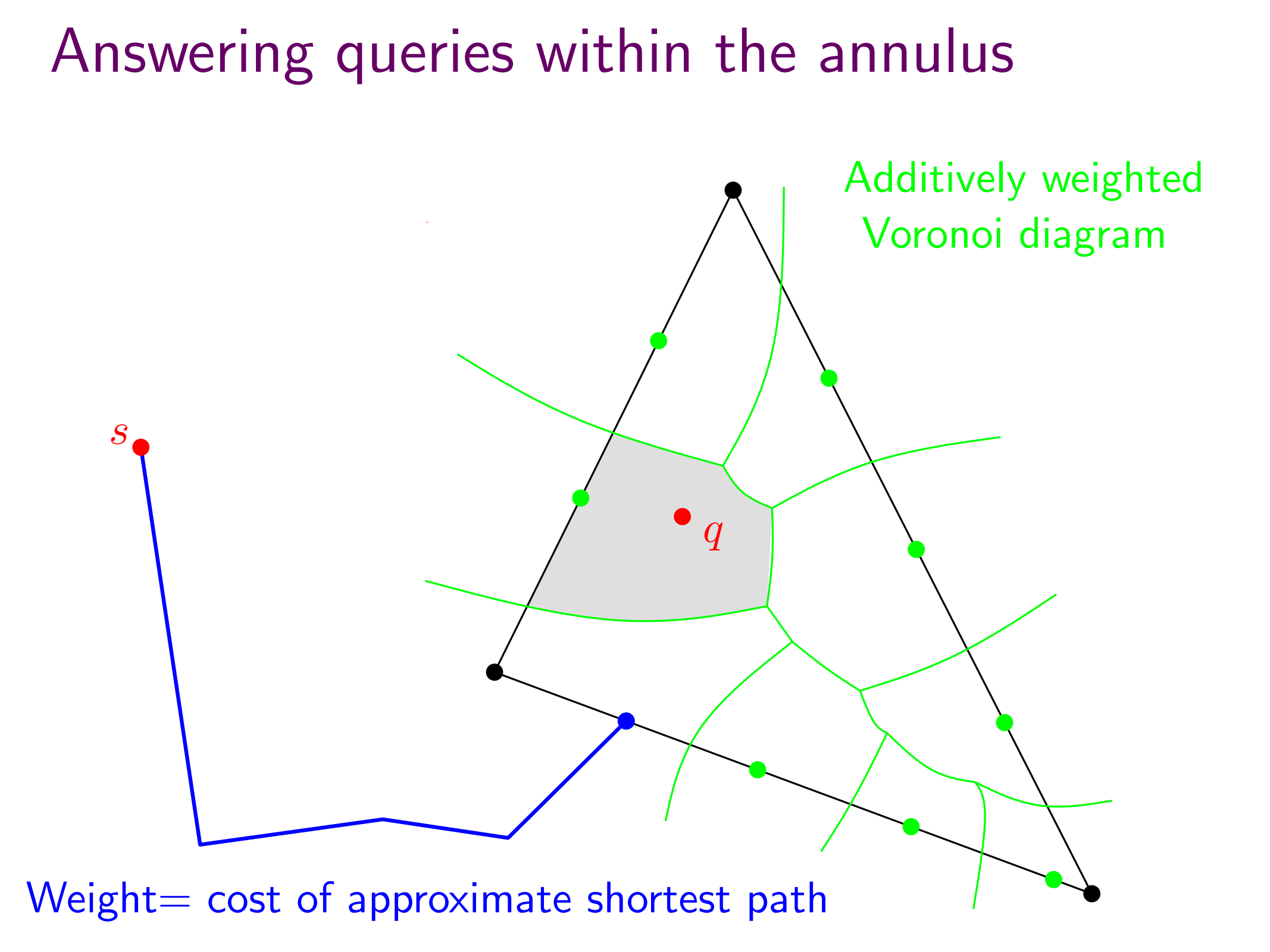
Answering queries within the annulus



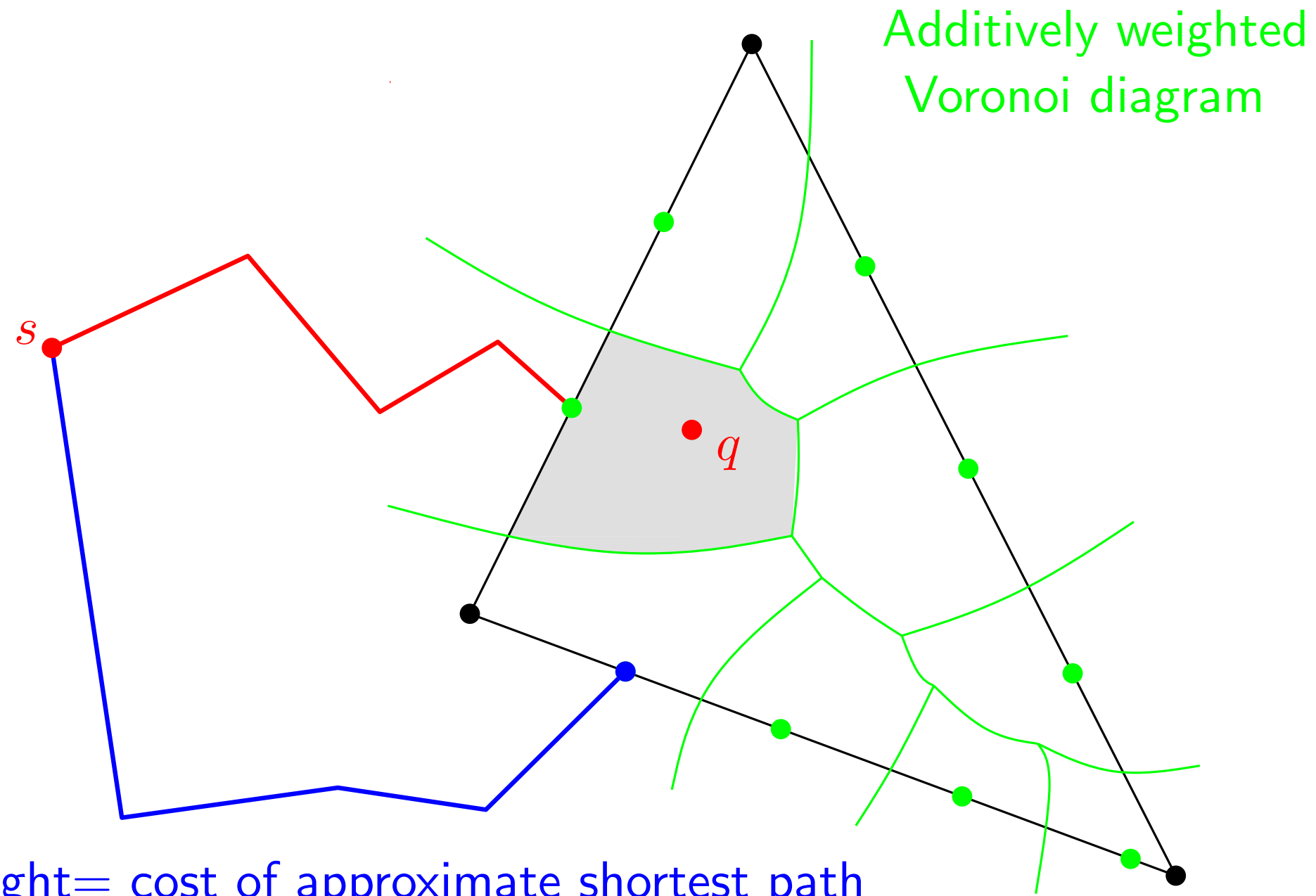
Answering queries within the annulus



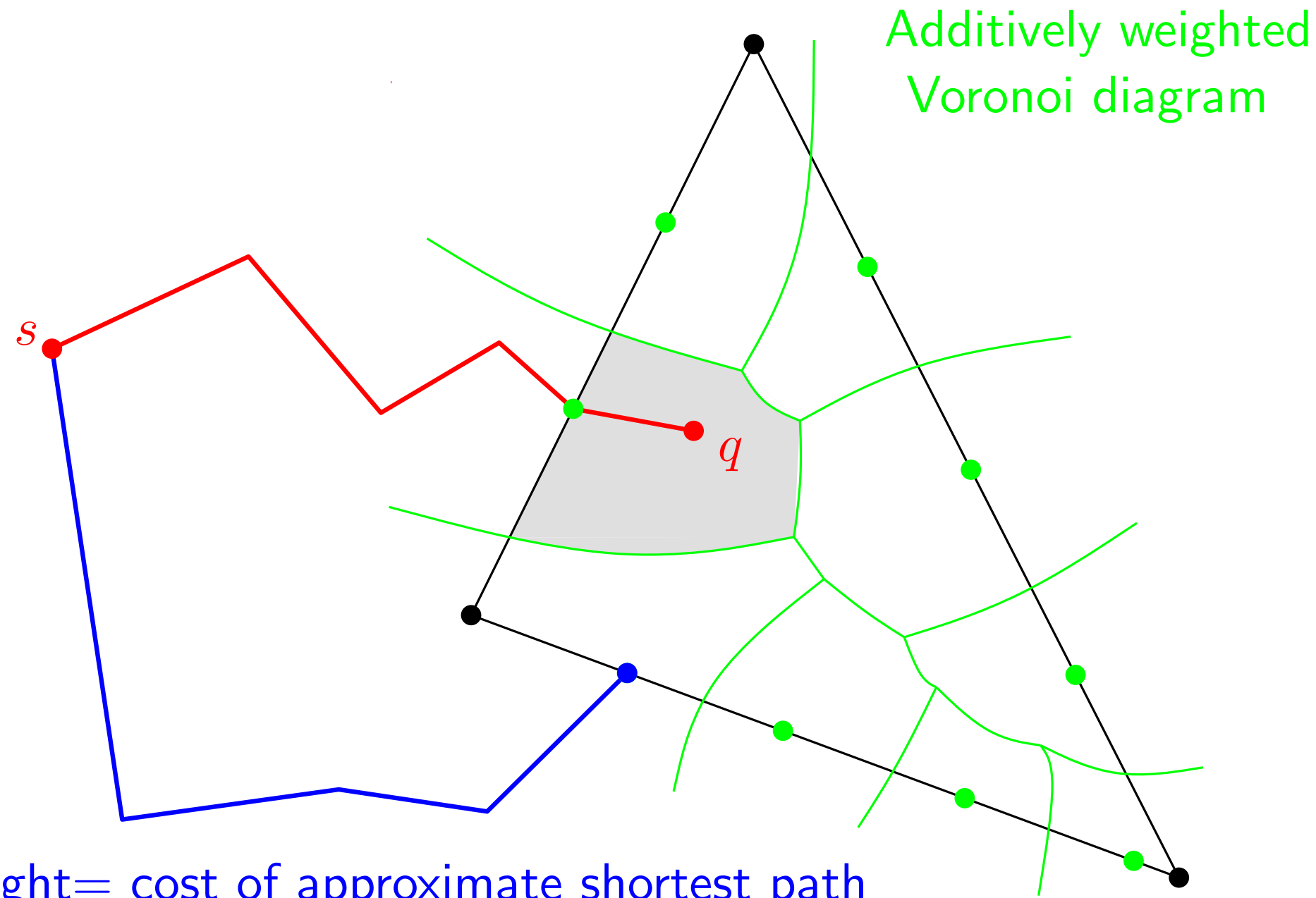
Answering queries within the annulus



Answering queries within the annulus



Answering queries within the annulus

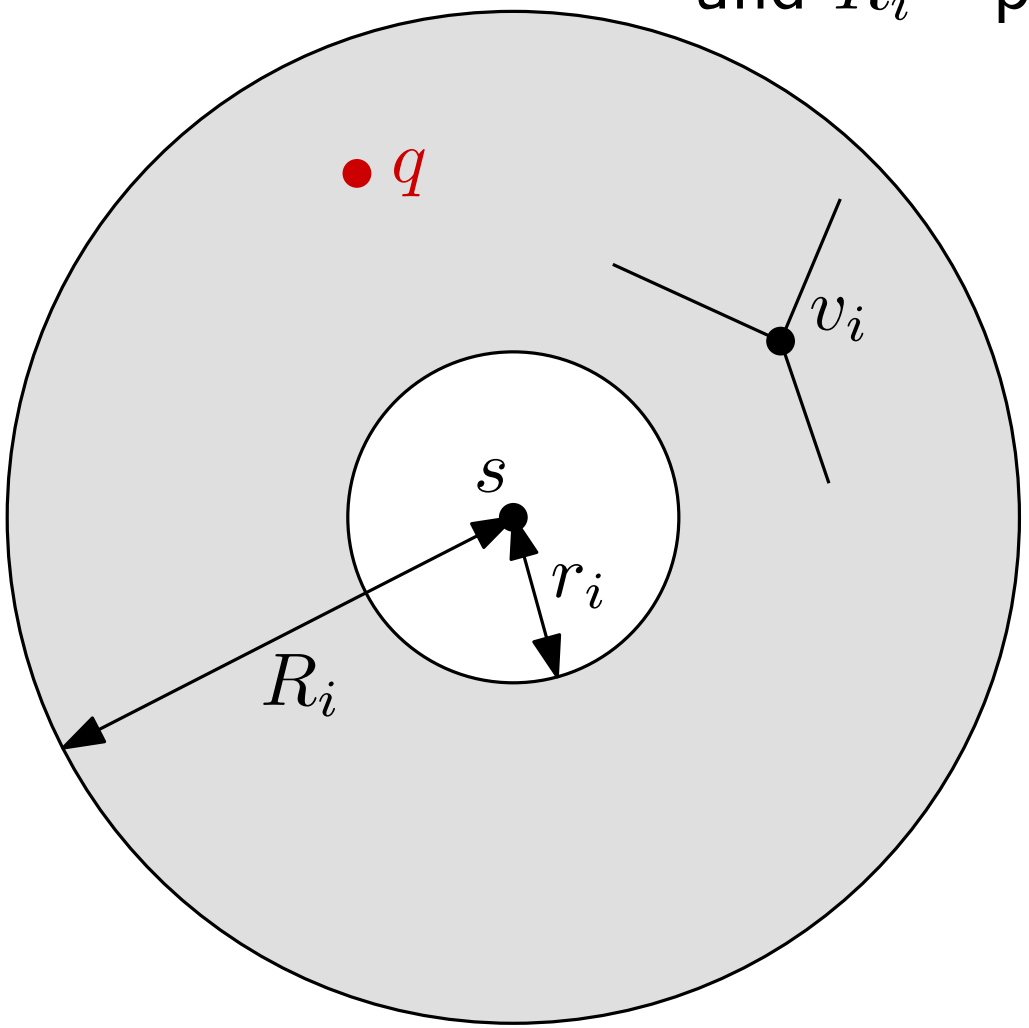


Overview

For each vertex v_i, we construct a data structure for the annulus with radius $r_i = d(s, v_i)/2\rho$ and $R_i = \text{poly}(\rho, n, 1/\varepsilon)d(s, v_i)$

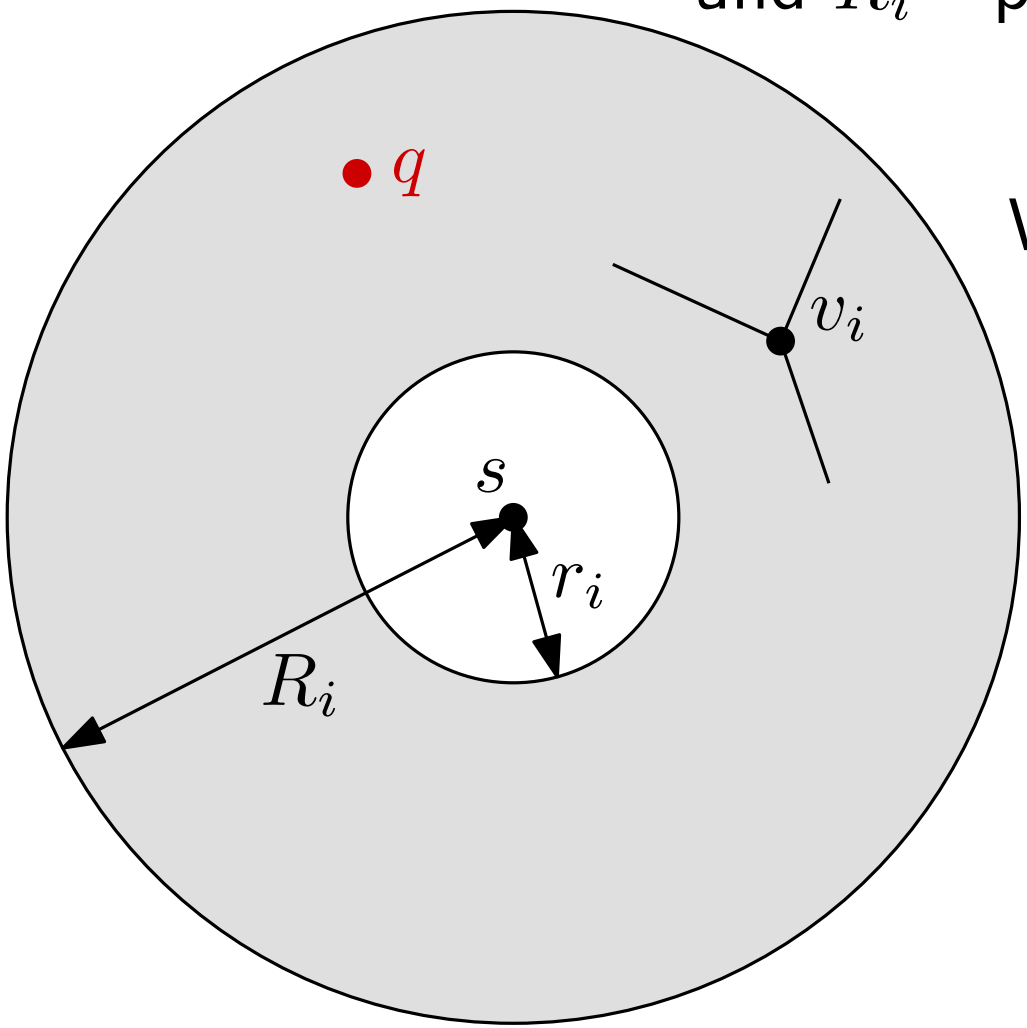
Overview

For each vertex v_i, we construct a data structure for the annulus with radius $r_i = d(s, v_i)/2\rho$ and $R_i = \text{poly}(\rho, n, 1/\varepsilon)d(s, v_i)$



Overview

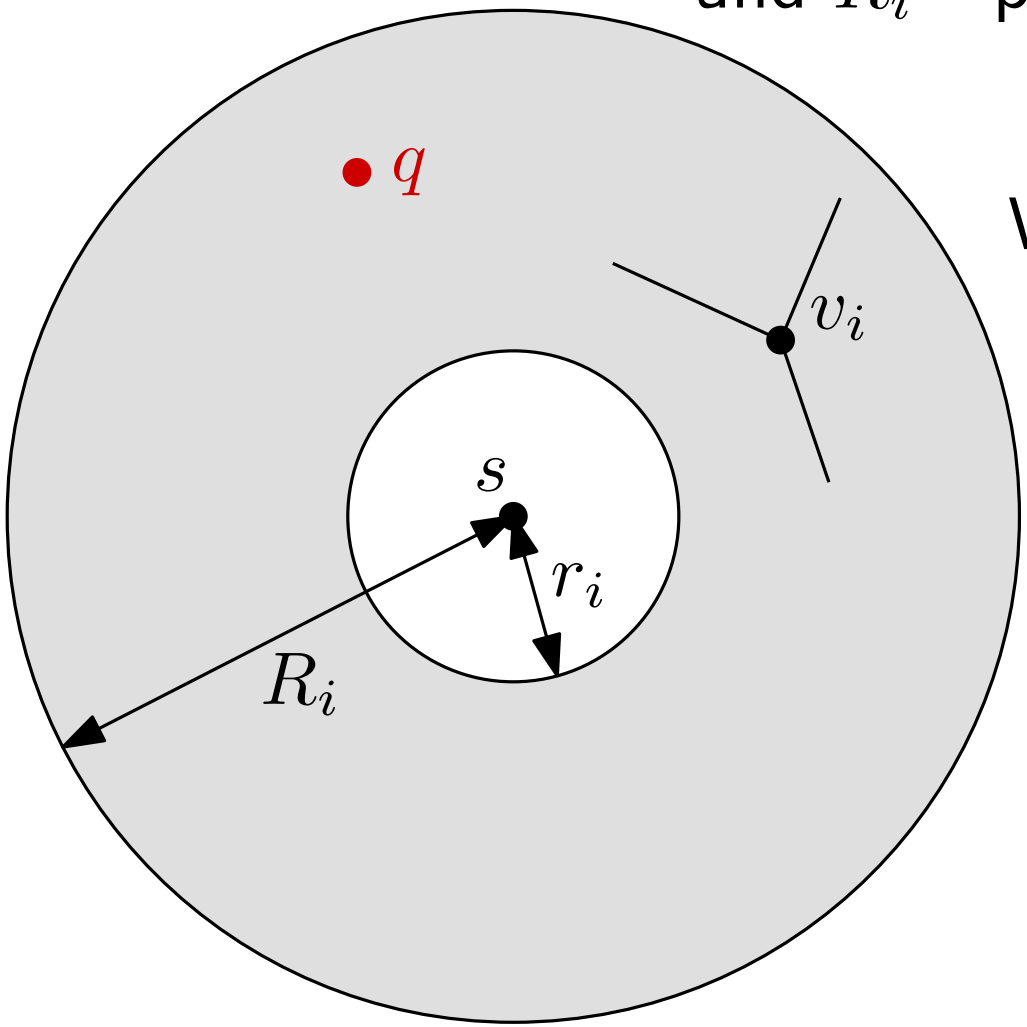
For each vertex v_i, we construct a data structure for the annulus with radius $r_i = d(s, v_i)/2\rho$ and $R_i = \text{poly}(\rho, n, 1/\varepsilon)d(s, v_i)$



We handle the neighborhood of s through scaling

Overview

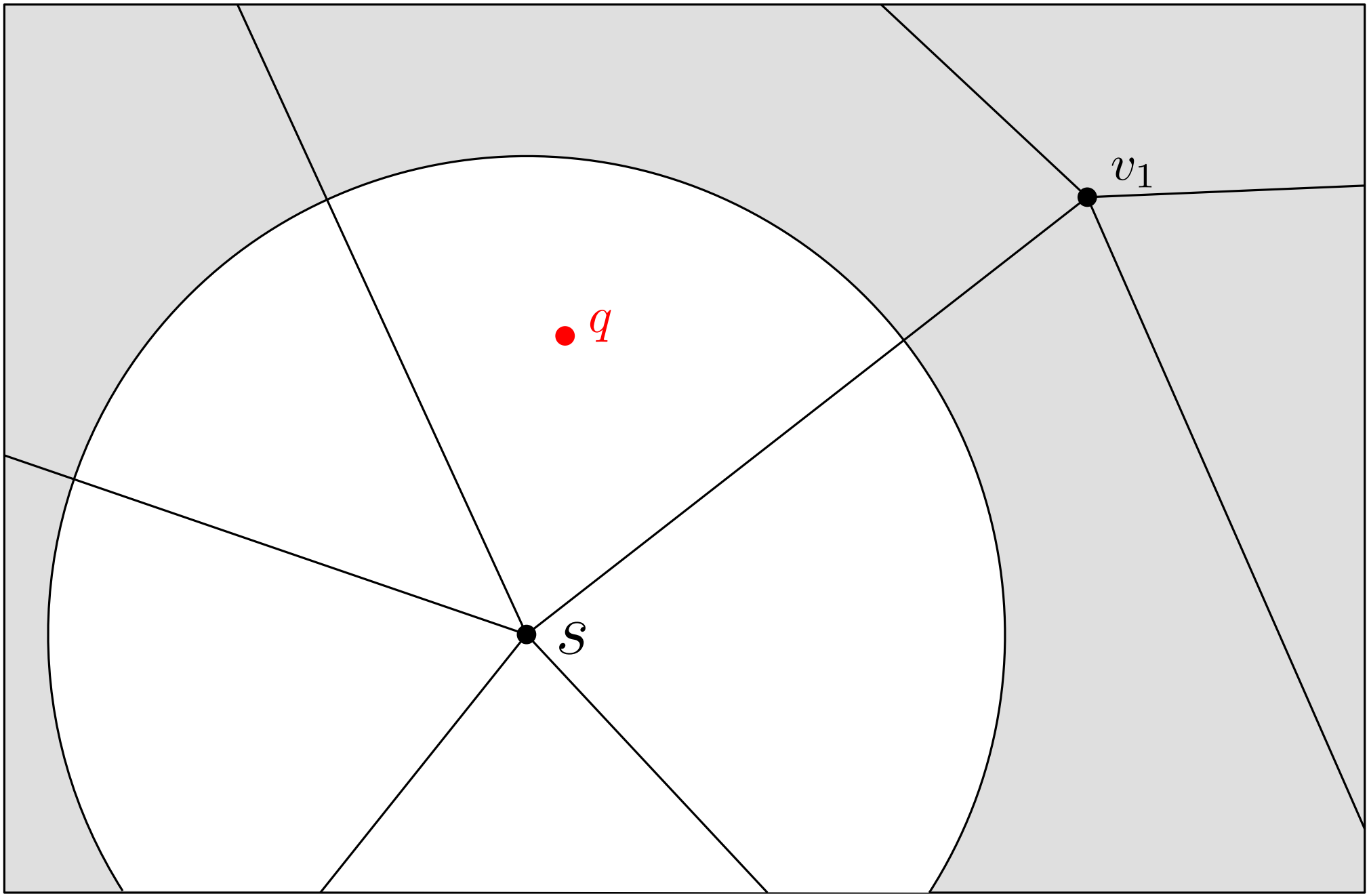
For each vertex v_i, we construct a data structure for the annulus with radius $r_i = d(s, v_i)/2\rho$ and $R_i = \text{poly}(\rho, n, 1/\varepsilon)d(s, v_i)$



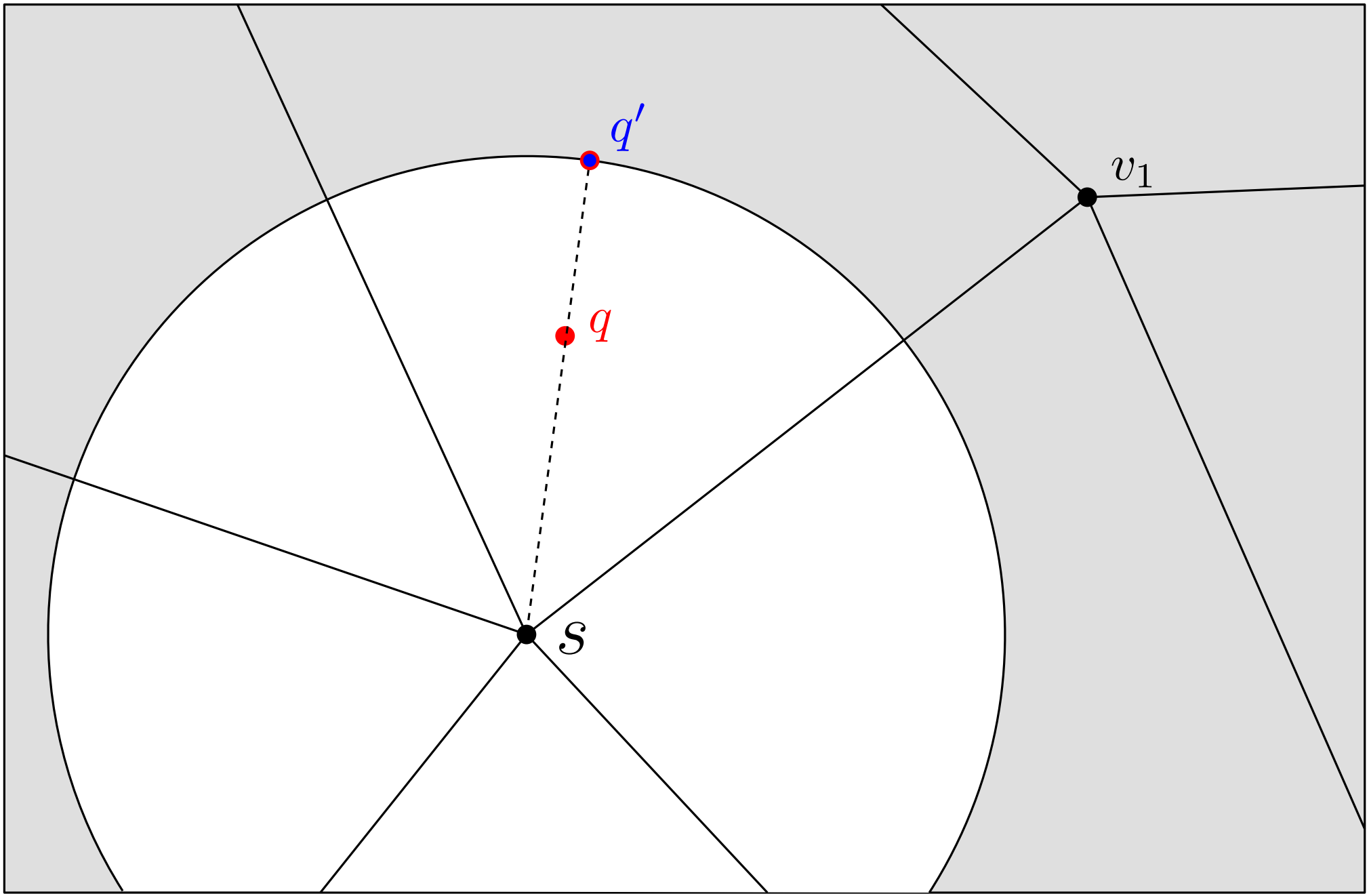
We handle the neighborhood of s through scaling

We handle the empty space (if any) through scaling and perturbation

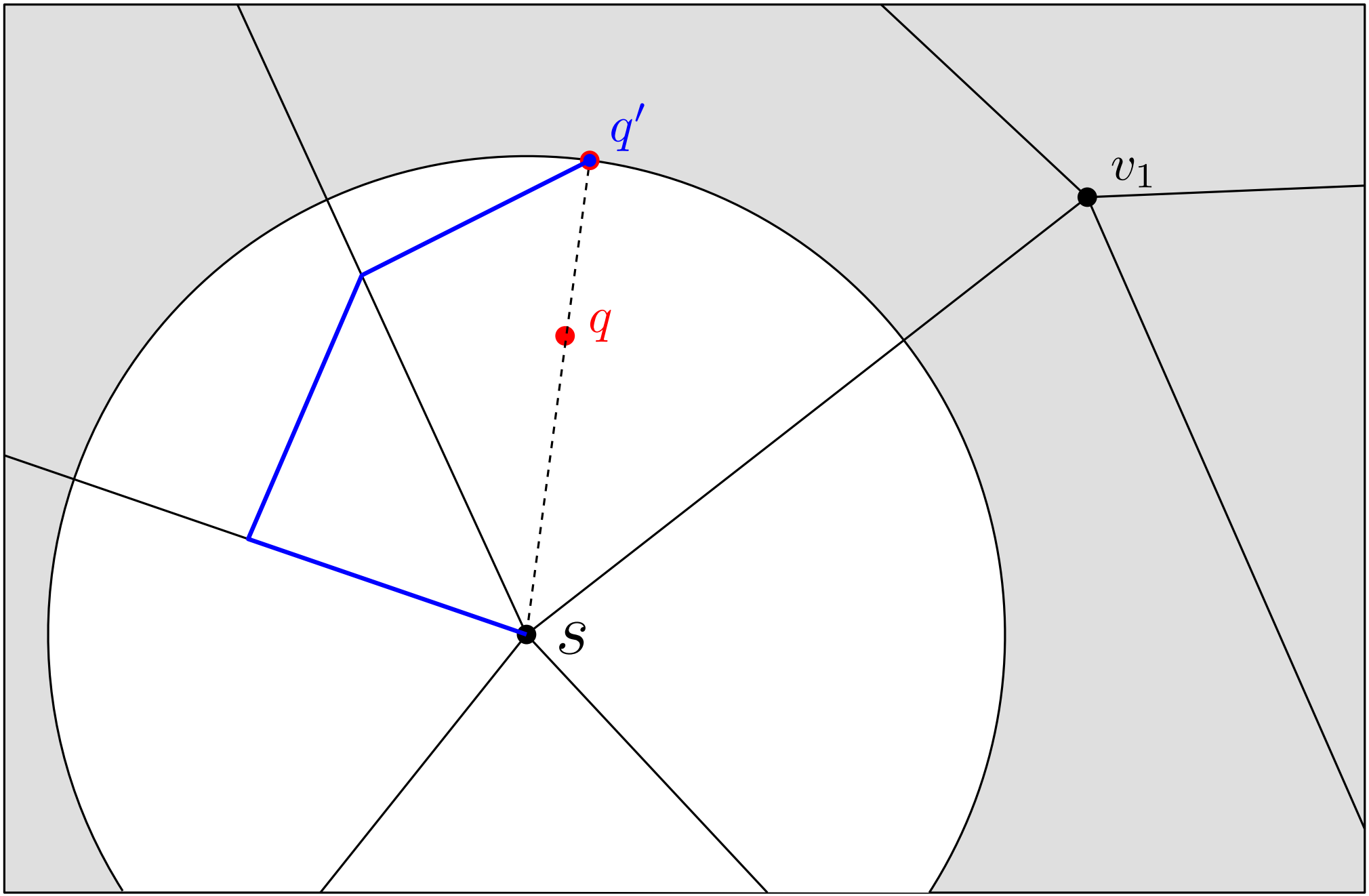
Close range queries



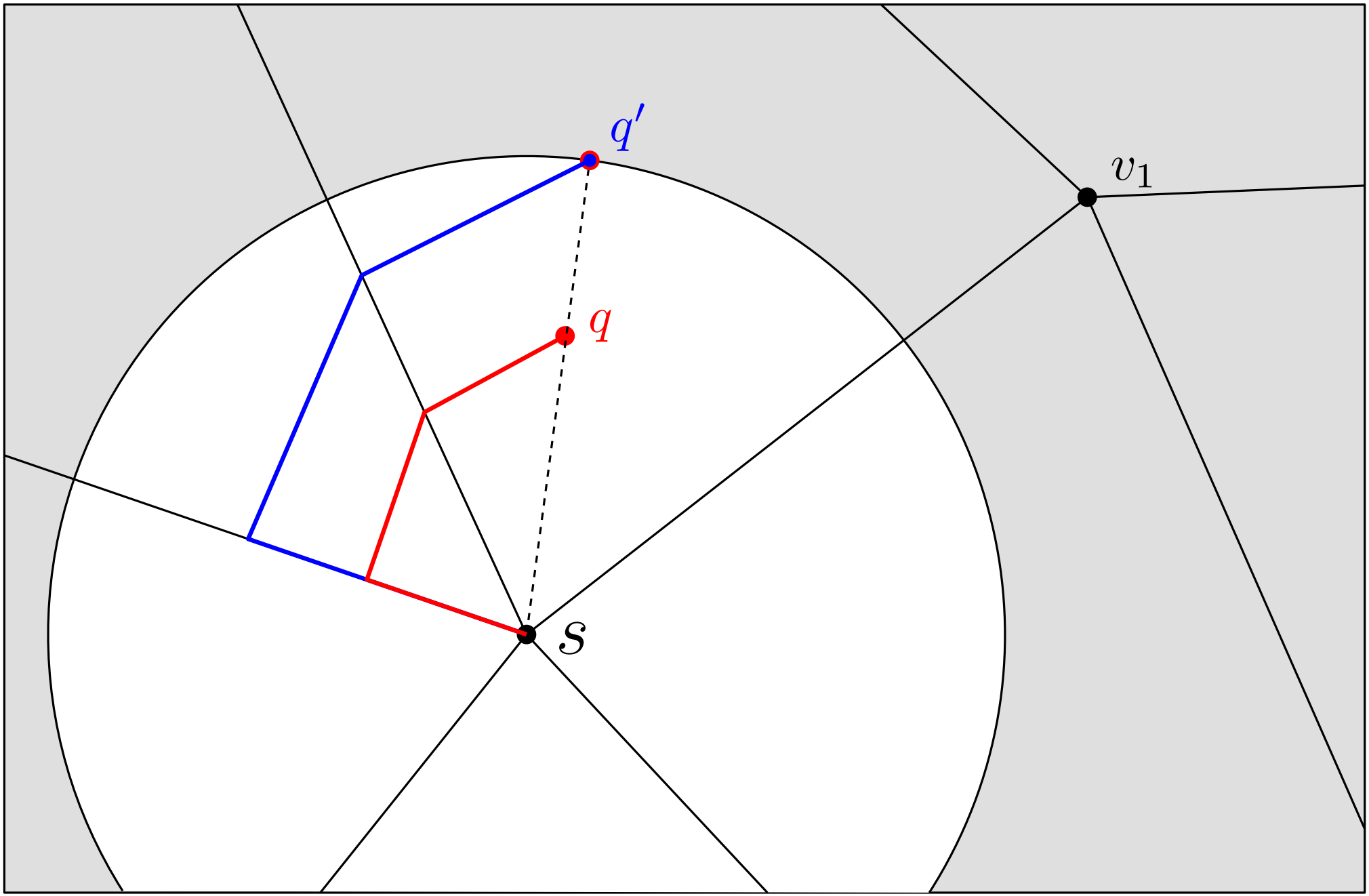
Close range queries



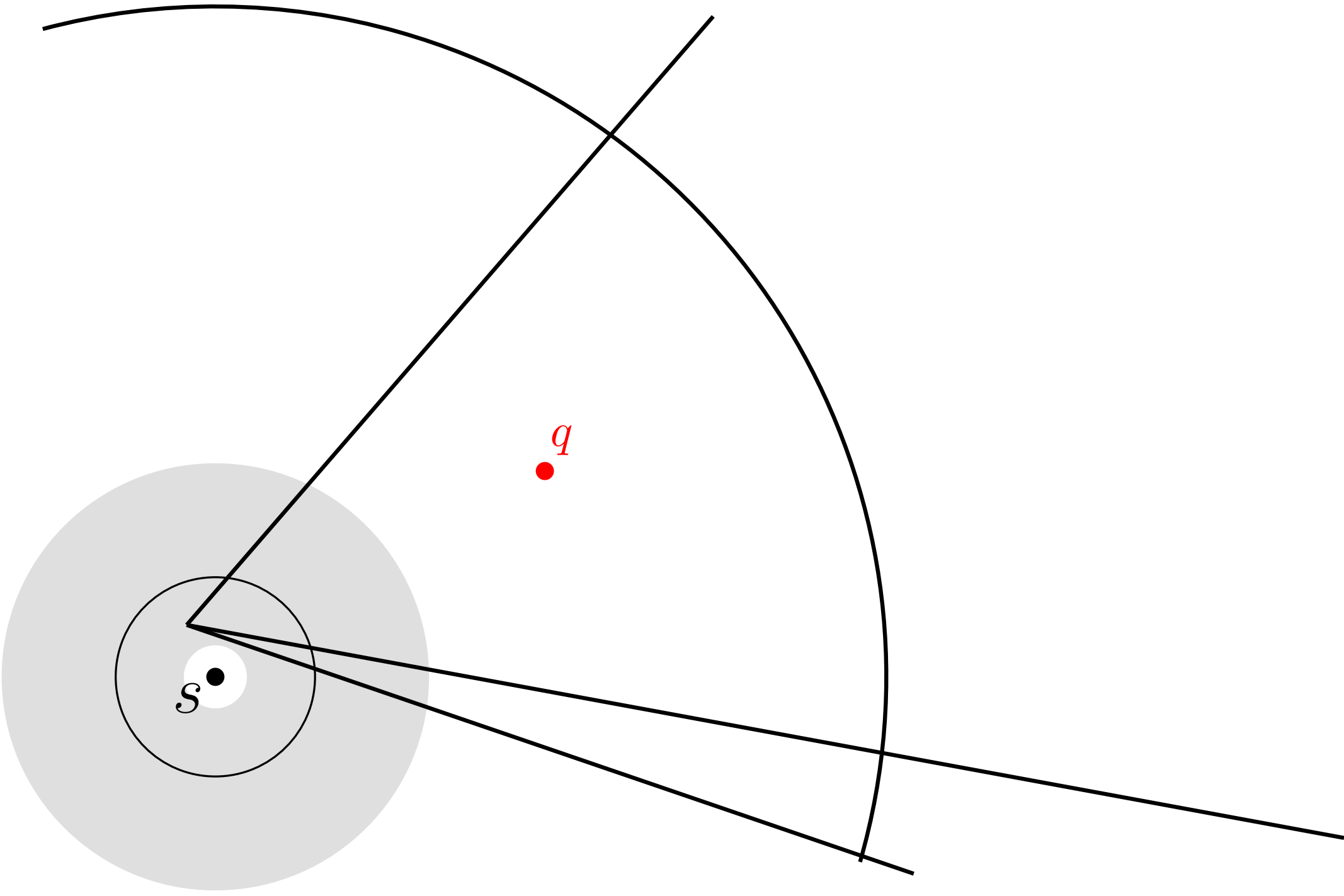
Close range queries



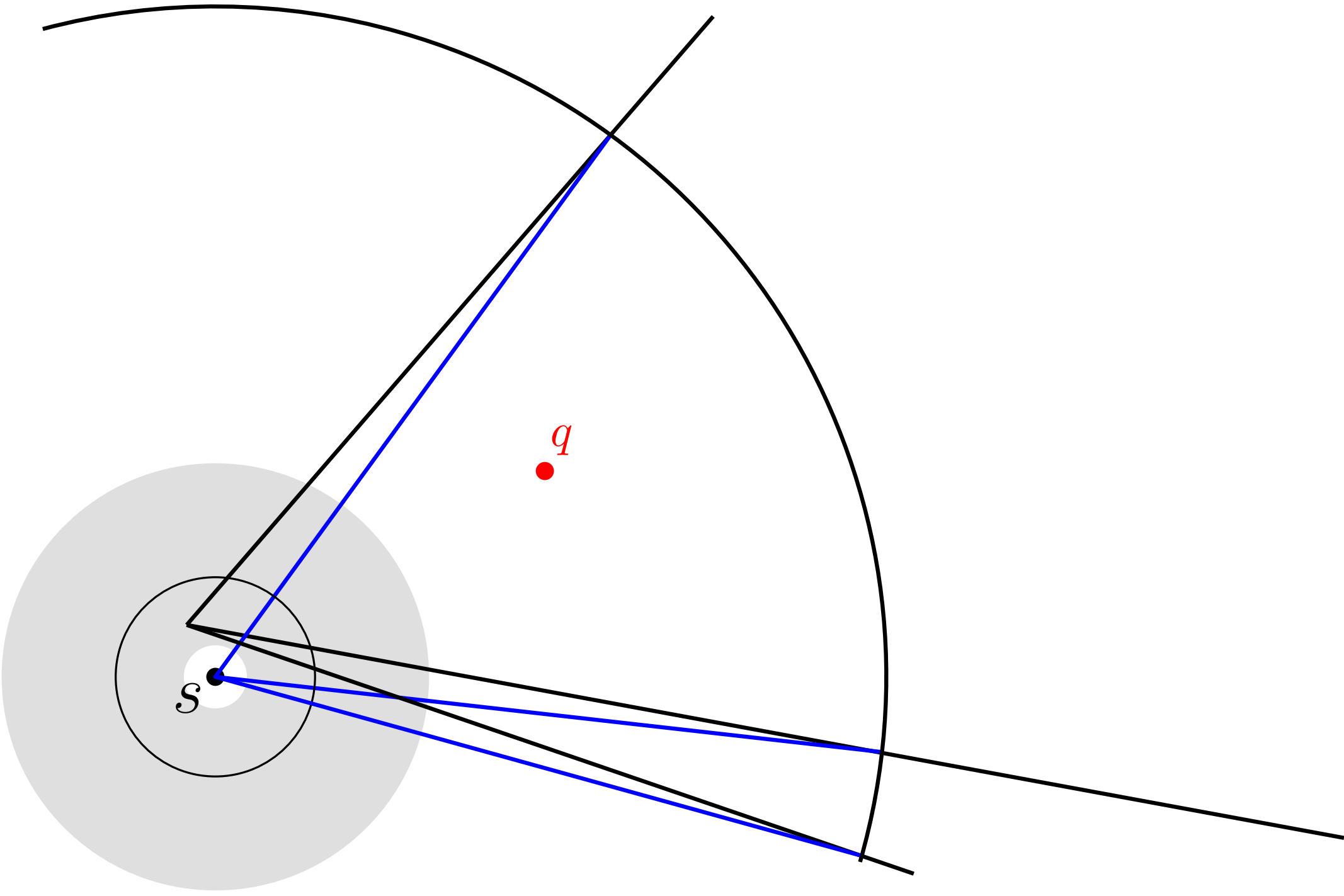
Close range queries



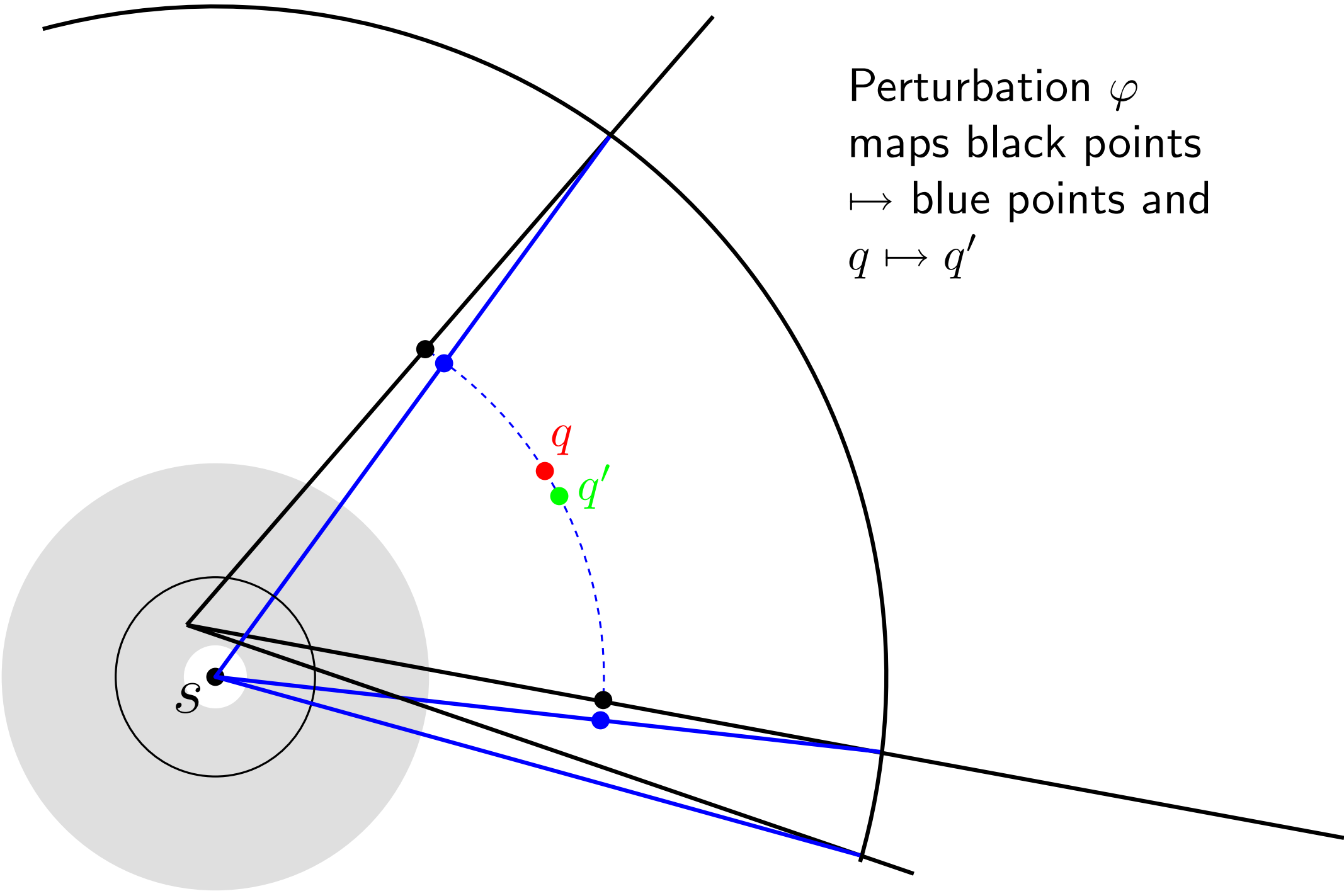
Between two annuli



Between two annuli

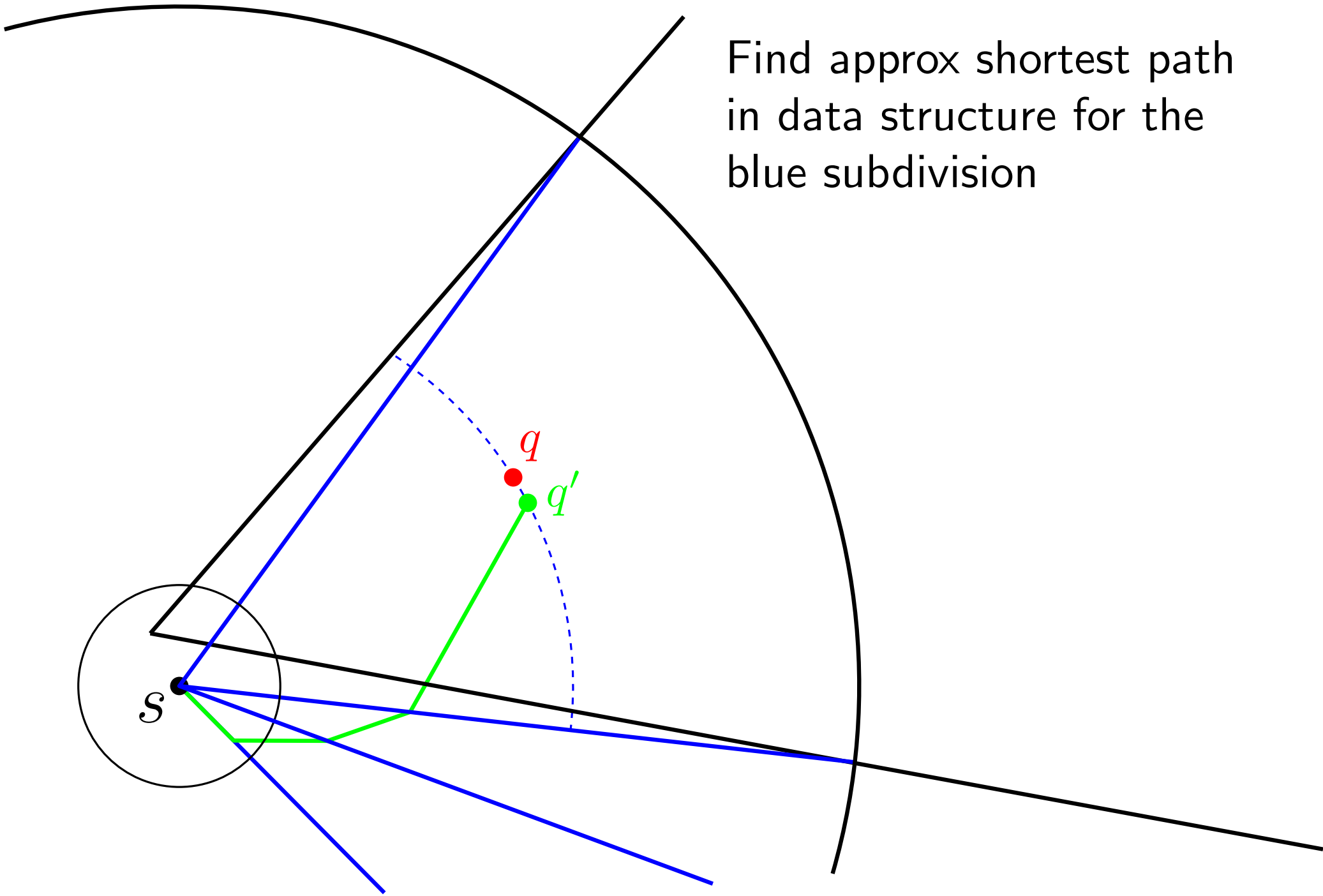


Between two annuli



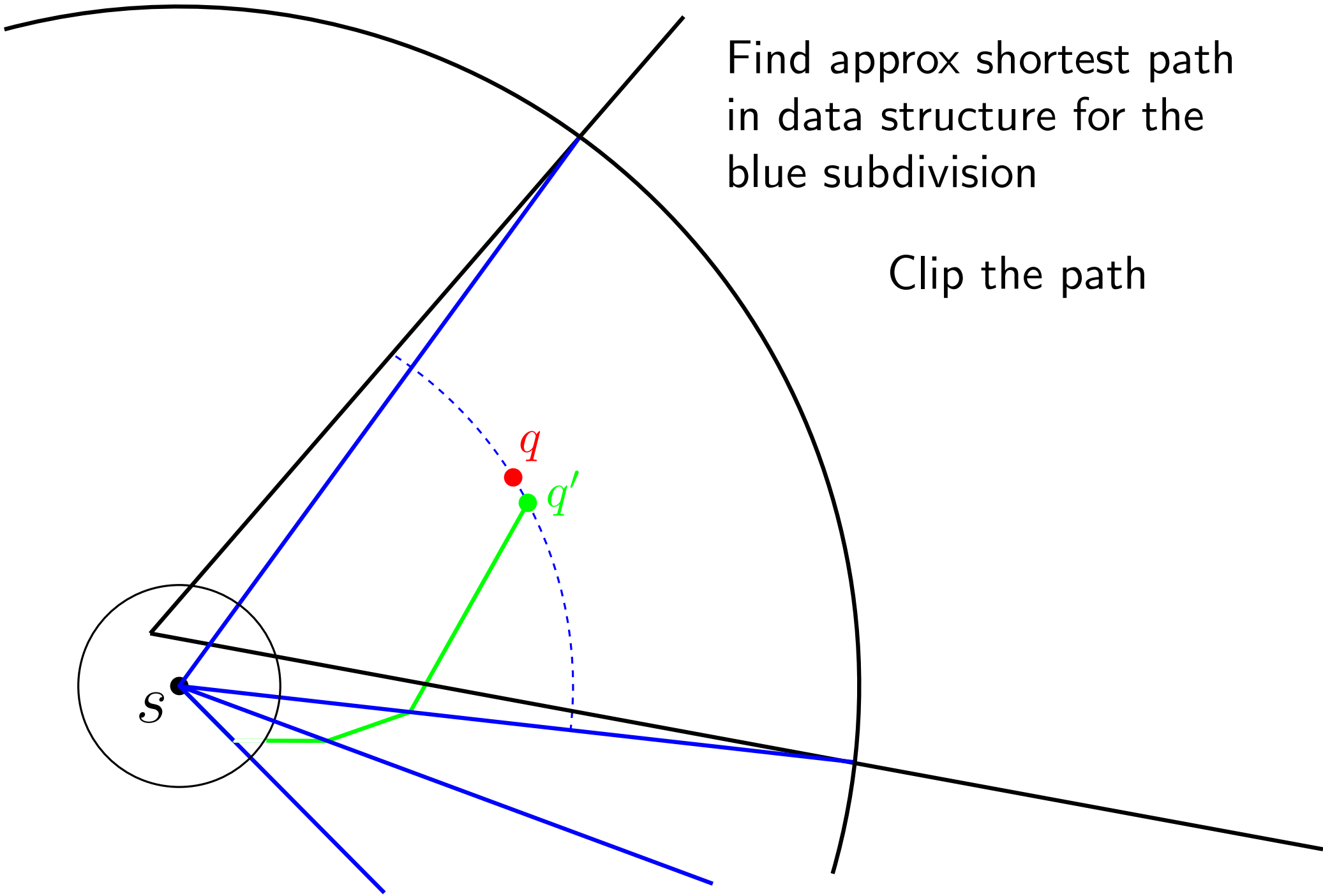
Perturbation φ
maps black points
$\mapsto$ blue points and
$q \mapsto q'$

Between two annuli



Find approx shortest path in data structure for the blue subdivision

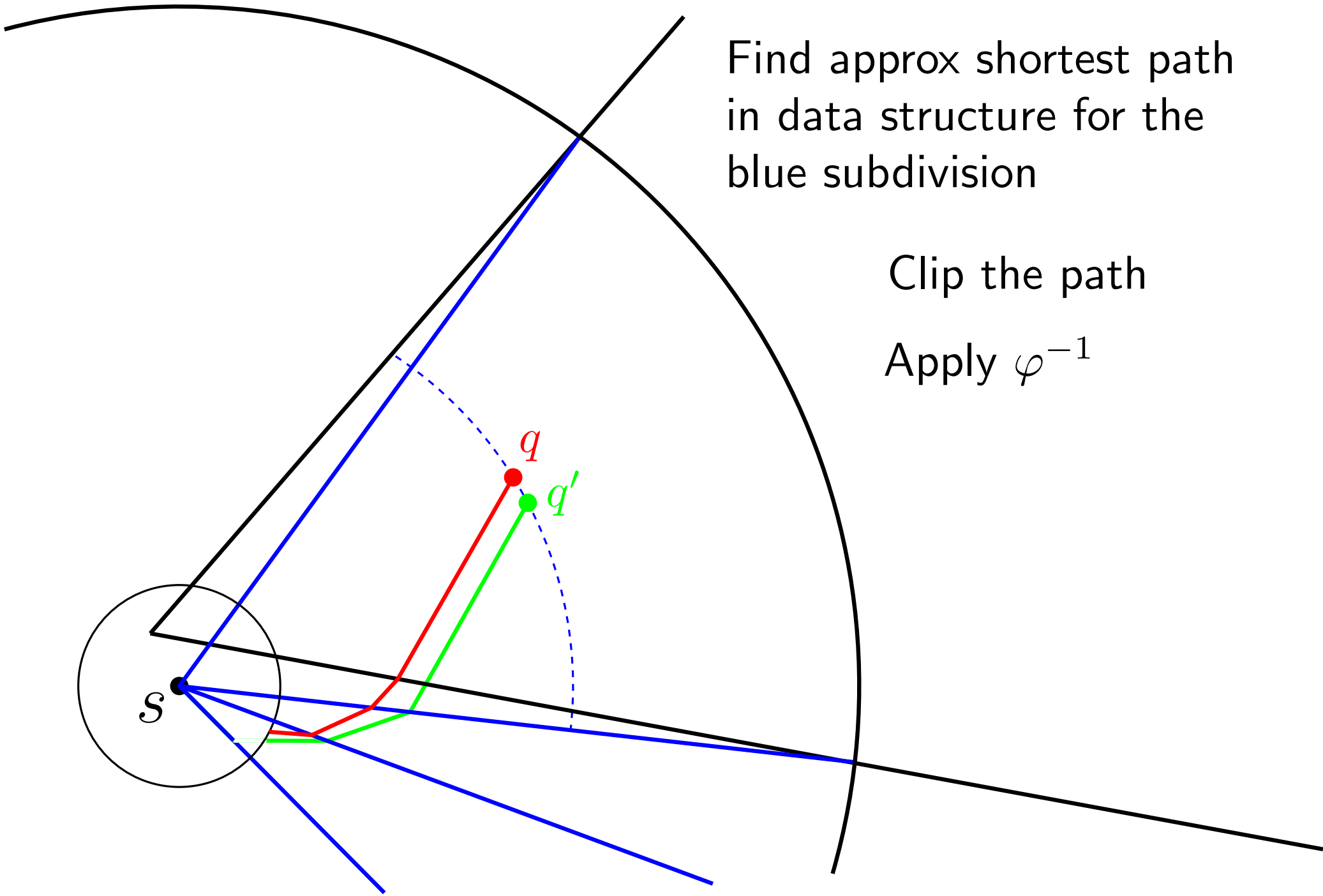
Between two annuli



Find approx shortest path in data structure for the blue subdivision

Clip the path

Between two annuli

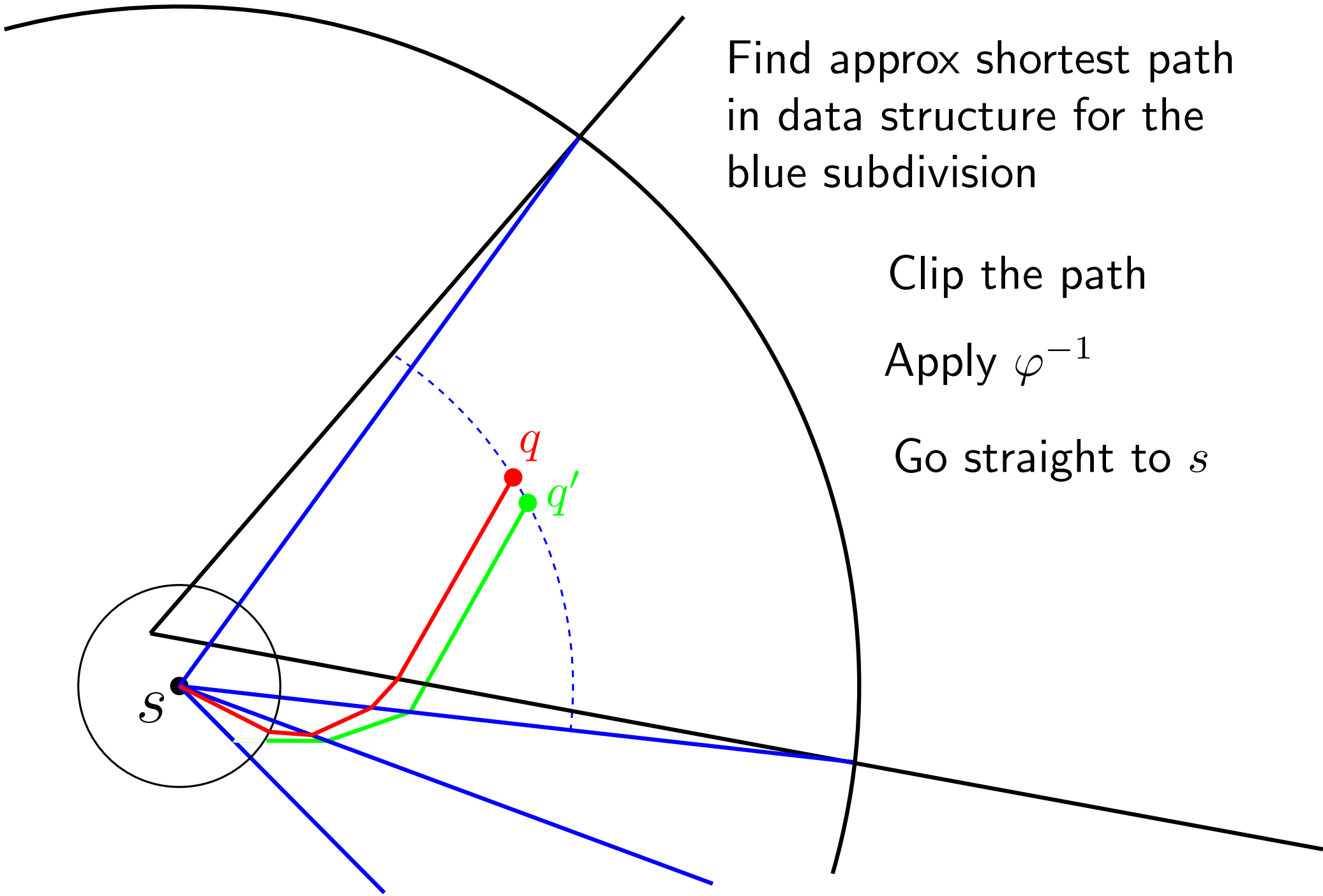


Find approx shortest path in data structure for the blue subdivision

Clip the path

Apply φ^{-1}

Between two annuli



Find approx shortest path in data structure for the blue subdivision

Clip the path

Apply φ^{-1}

Go straight to s

Conclusion

We can also handle obstacles

Conclusion

We can also handle obstacles

2 point queries?

Conclusion

We can also handle obstacles

2 point queries?