Wolfgang Mulzer
Institut für Informatik

Freie Universität Berlin

Computational Geometry with Limited Work-Space: State of the Art and Challenges

Algorithms with Limited Work-Space

Today's software is versatile, fast, and BIG.

BUT: Small devices need to compute with little memory.

Flash-Memory has slow write speed.

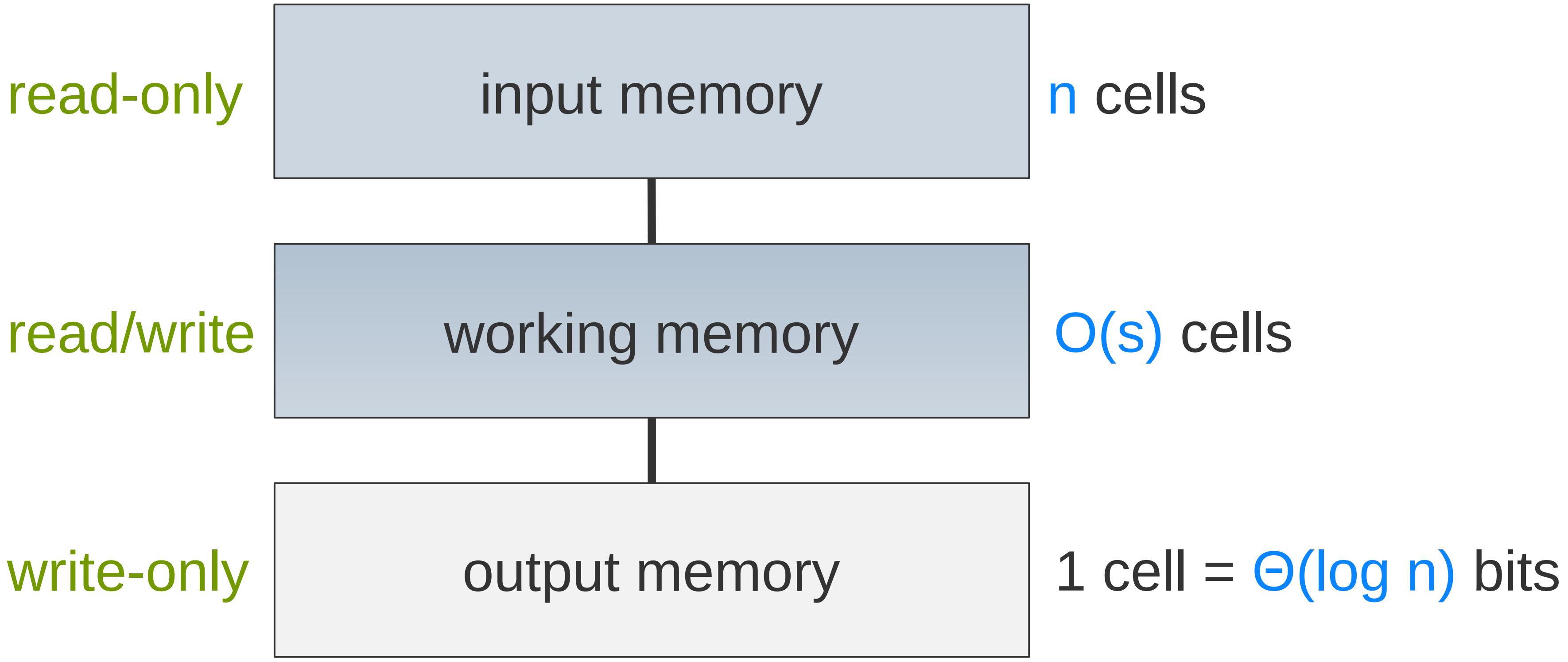
Algorithms with Limited Work-Space

Traditional algorithm design is mostly focused on running time.

We would like algorithms that are sensitive to space usage (while still achieving good running time).

The Model

Parameter: $s \in \{1, \ldots, n\}$.

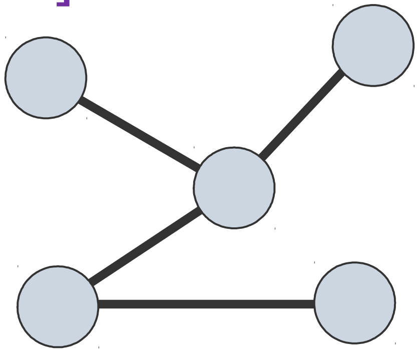


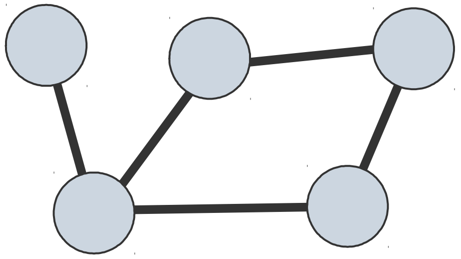
Goal: best possible running time with s cells of work-space.

Context – Complexity Theory

The case $s = 1$ is well known in computational complexity theory: the complexity class LOGSPACE.

Many results. Most prominently: Reingold's LOGSPACE algorithm for st-connectivity in undirected graphs ("SL=L") [2005].





But: little attention to efficient running times.

Context – Related Models

Streaming:

Same:	O(s) cells of work-space
Different:	input is read-once

In-Place:

Same:	O(1) cells of work-space in addition to input
Different:	input is read/write

Succinct:

Same:	aim for efficiency with little space
Different:	account for exact number of bits

Algorithms for Constant Work-Space

Initial focus was on $s = O(1)$.

Some classic results, e.g. Munro and Paterson’s selection algorithms [1978].

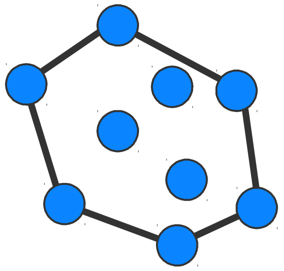
Asano, M, Rote, Wang [2011]: several constant work-space algorithms for problems in computational geometry.

Since then: countless additional results and techniques.

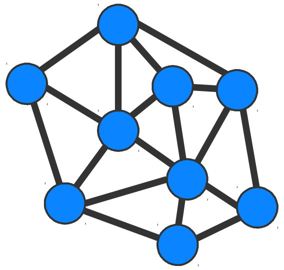
Algorithms for Constant Work-Space

Examples:

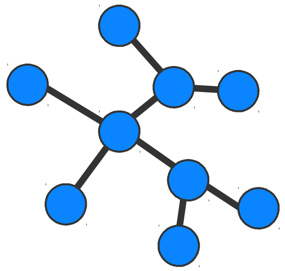
Convex Hull in $O(n^2)$ time
AMRW
[2011]



Delaunay triangulation in $O(n^2)$ time
AMRW
[2011]



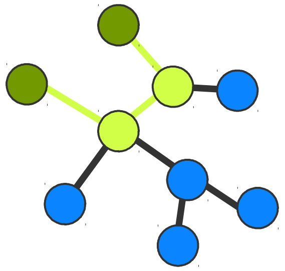
Euclidean minimum spanning tree in $O(n^3)$ time
AMRW
[2011]



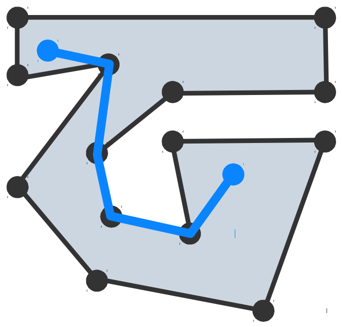
Algorithms for Constant Work-Space

Examples:

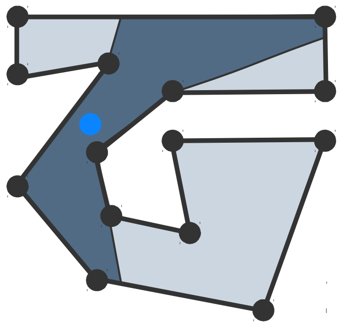
Shortest path in a tree in O(n) time
AMW [2011]



Shortest path in a polygon in O(n^2) time
AMW, AMRW [2011]



Visibility region in a polygon in O(nr) time
BKLS [2014]

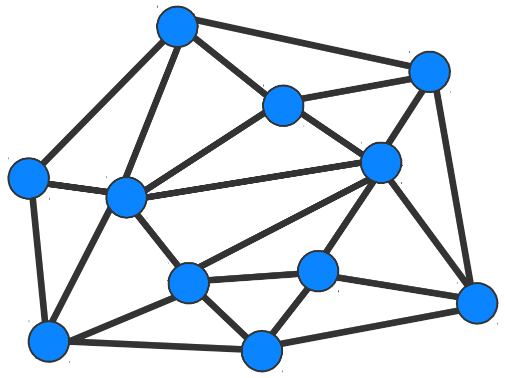


And many more…

Example I: Delaunay Triangulations

Given: A set S of n sites in the plane

Want: The Delaunay triangulation of S



We may use only a constant amount of work-space.

Idea AMRW [2011]: Report edges for each site, in ccw order.

Example I: Delaunay Triangulations

Algorithm:

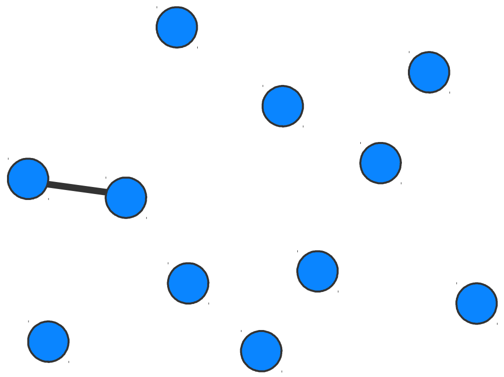
Let s є S

Find nearest neighbor s' of s

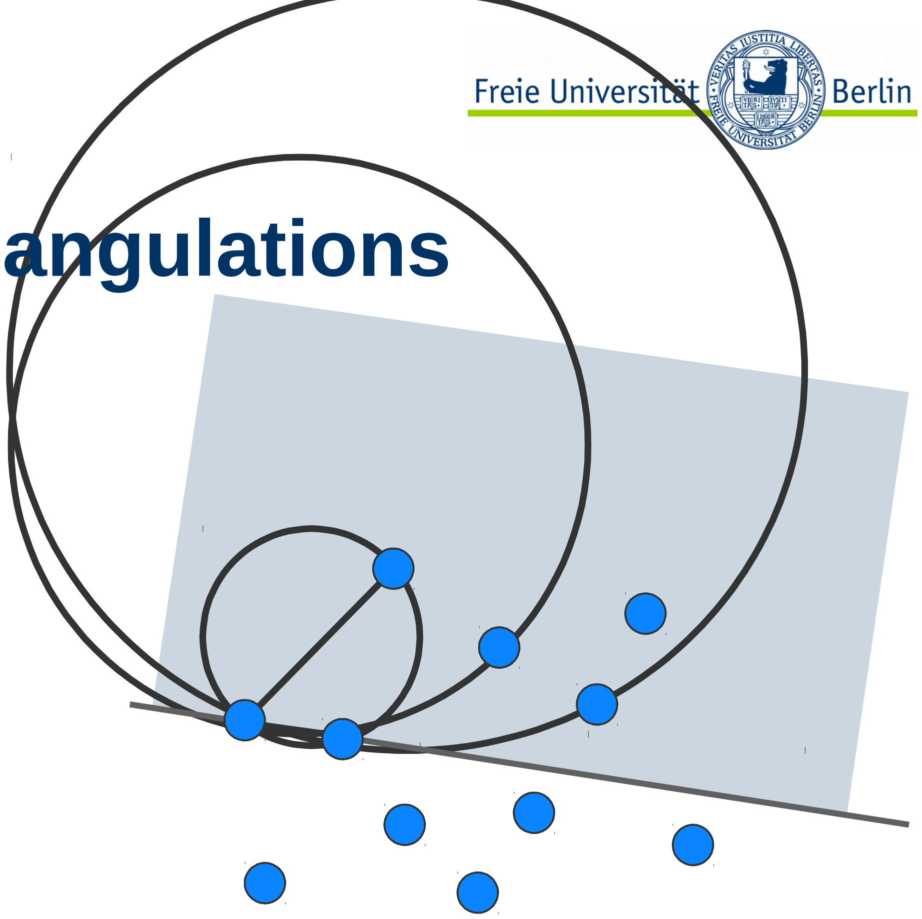
Know: ss' is edge of DT(S)

Set current edge to ss'

Repeatedly find ccw neighbor of current edge, until back to ss'



Example I: Delaunay Triangulations



Algorithm:

H: upper halfplane for current edge

If $H \cap S \neq \emptyset$:

t: first point in $H \cap S$

current disk: given by s, s’, t

go through $H \cap S$ and update current disk and t

next current edge: st

Running Time: $O(n)$

Example I: Delaunay Triangulations

Algorithm:

H: upper halfplane for current edge

If $H \cap S \neq \emptyset$:

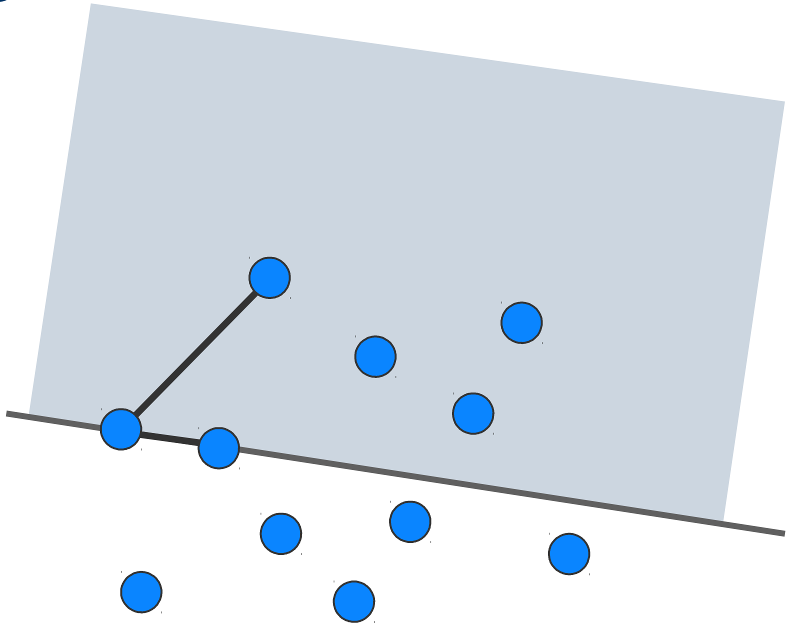
t: first point in $H \cap S$

current disk: given by s, s', t

go through $H \cap S$ and
update current disk and t

next current edge: st

Correctness: Endpoint of next current edge must be in current disk



Example I: Delaunay Triangulations

Algorithm:

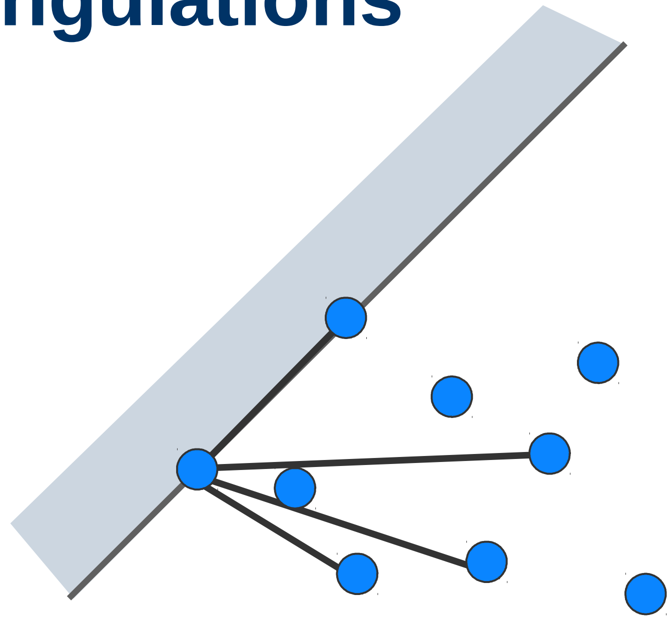
H: upper halfplane for current edge

If $H \cap S = \emptyset$:

do gift wrapping on s

next current edge: st

Running Time: O(n)



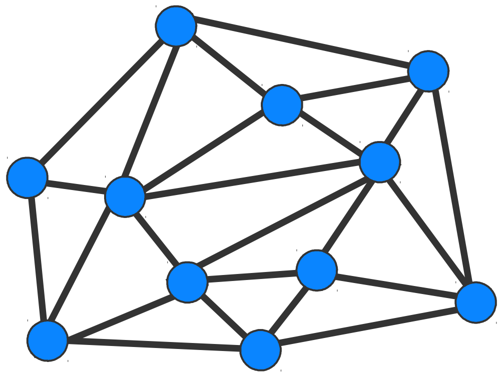
Example I: Delaunay Triangulations

Each step uses only O(1) words of work-space

We need O(n) time per edge.

The total running time is $O(n^2)$

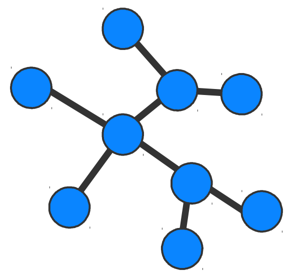
Through appropriate tie-breaking,
we can report each edge only once.



Example II: Paths in Trees

Given: A tree T with n vertices, $s, t \in T$

Want: The path in T from s to t



We may use only a constant amount of work-space.

Idea AMW [2011]: Find the edges one by one

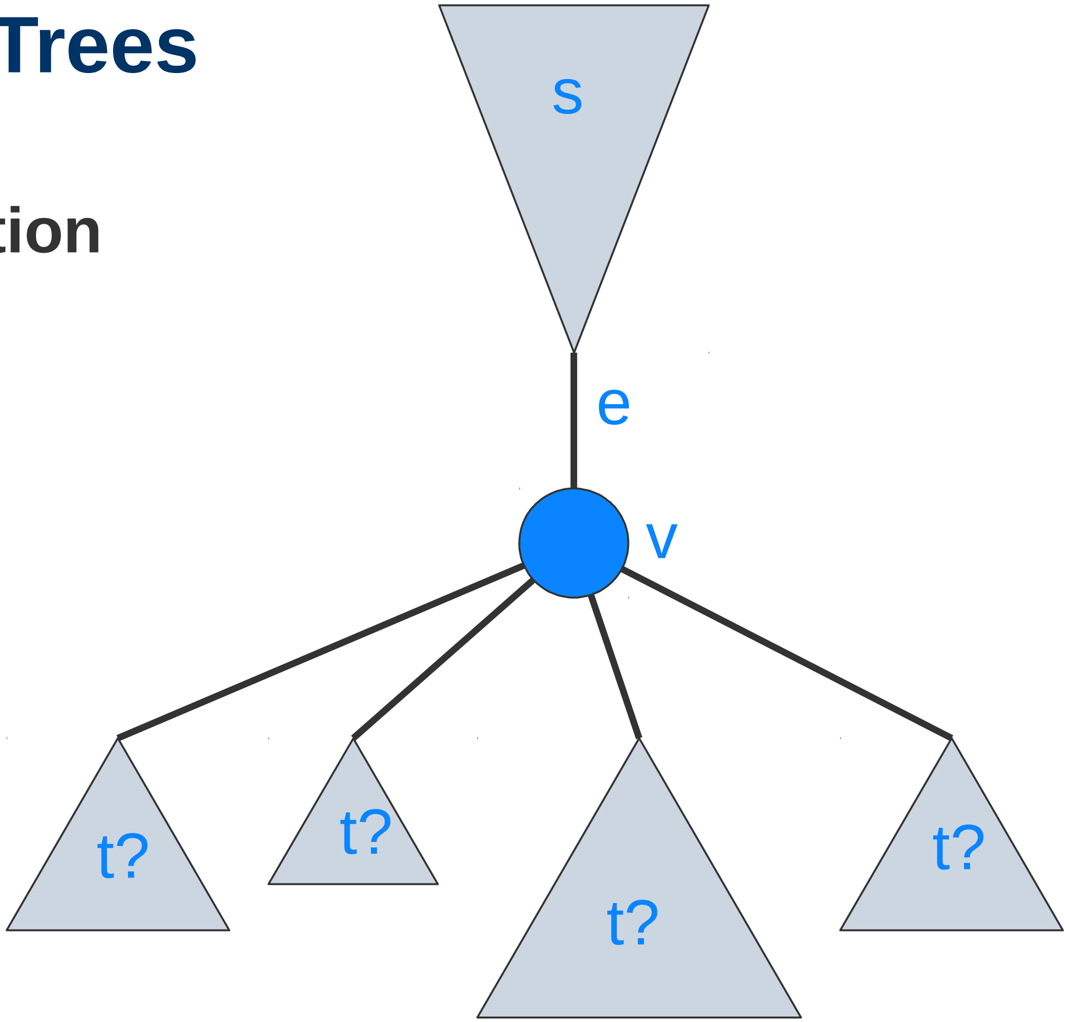
Example II: Paths in Trees

Algorithm – general situation

Are at a node v of T

Know the edge e where we entered

Goal: Find subtree of v that contains t



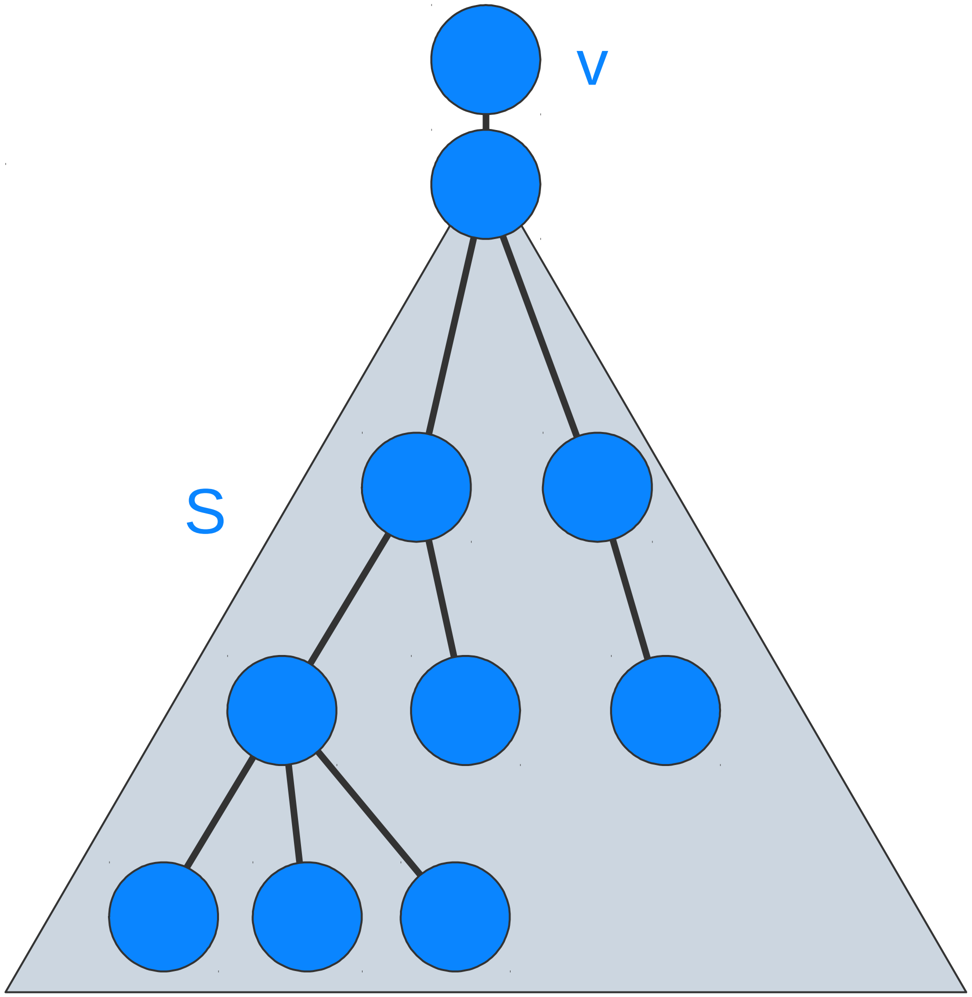
By iterating, we can find the path from s to t

Example II: Paths in Trees

Observation:

For a subtree S of v, we can check if t ∈ S using an Euler Tour

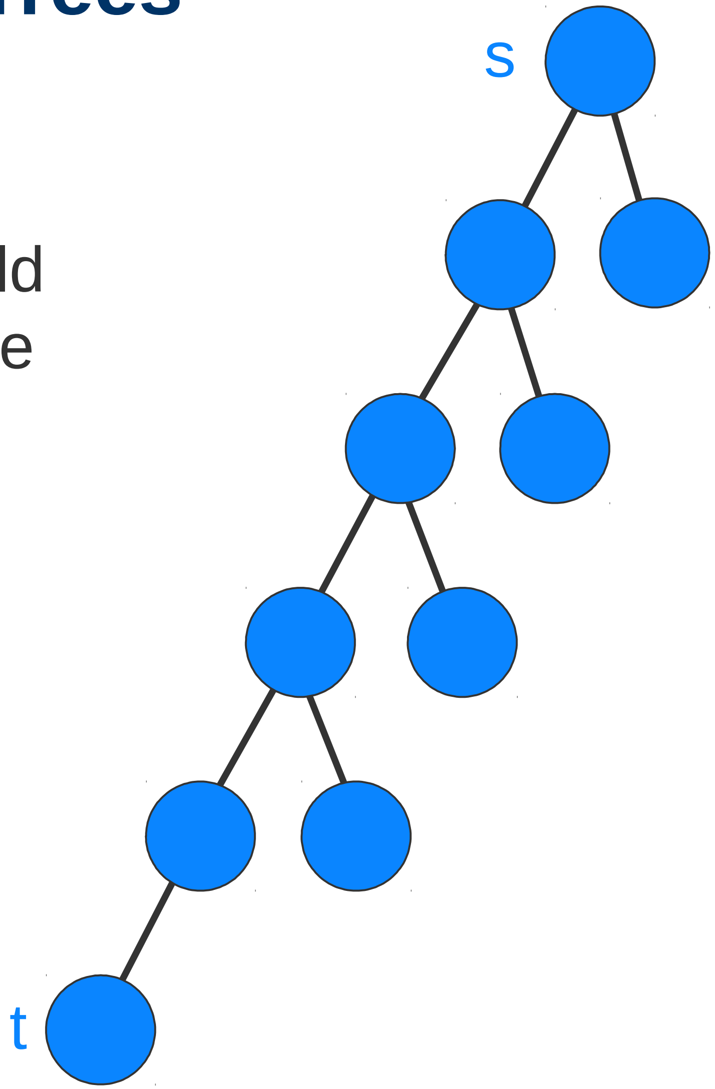
This needs O(|S|) time and O(1) words of work-space



Example II: Paths in Trees

However:

Repeated Euler-Tours could yield quadratic running time



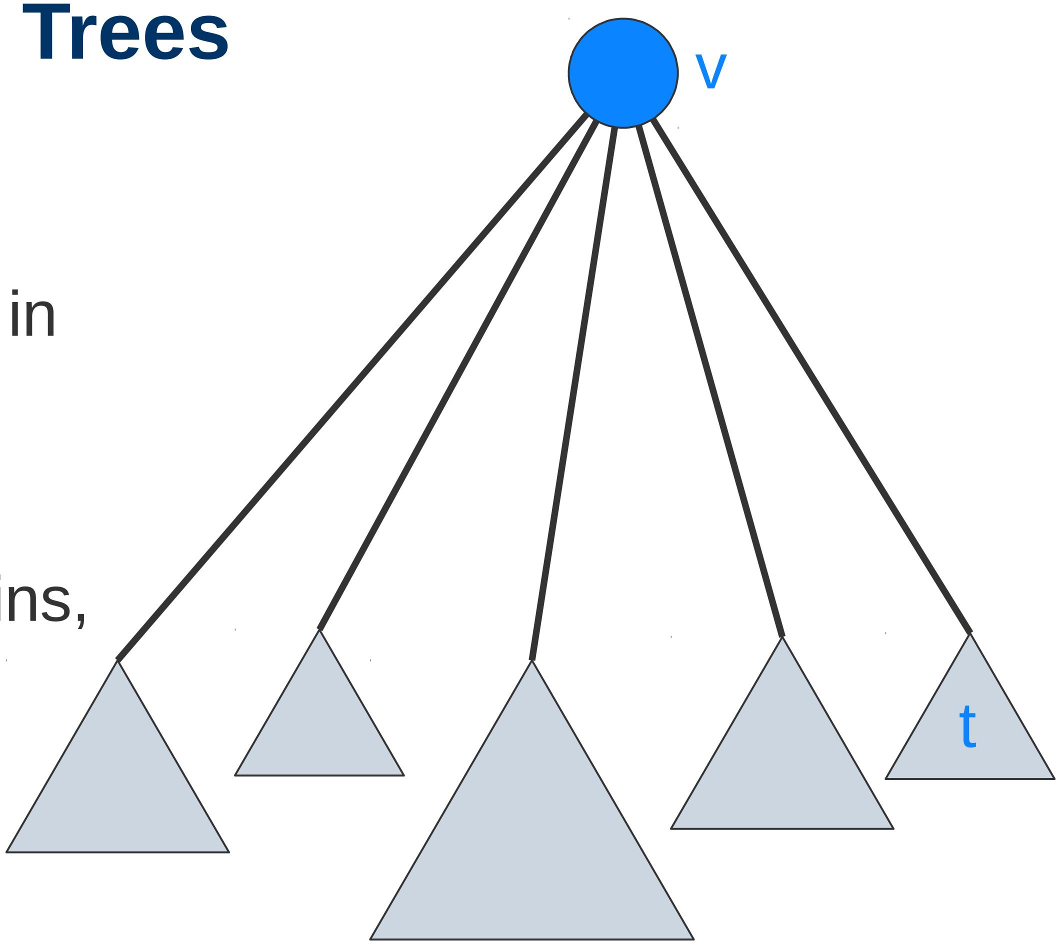
Example II: Paths in Trees

Additional trick:

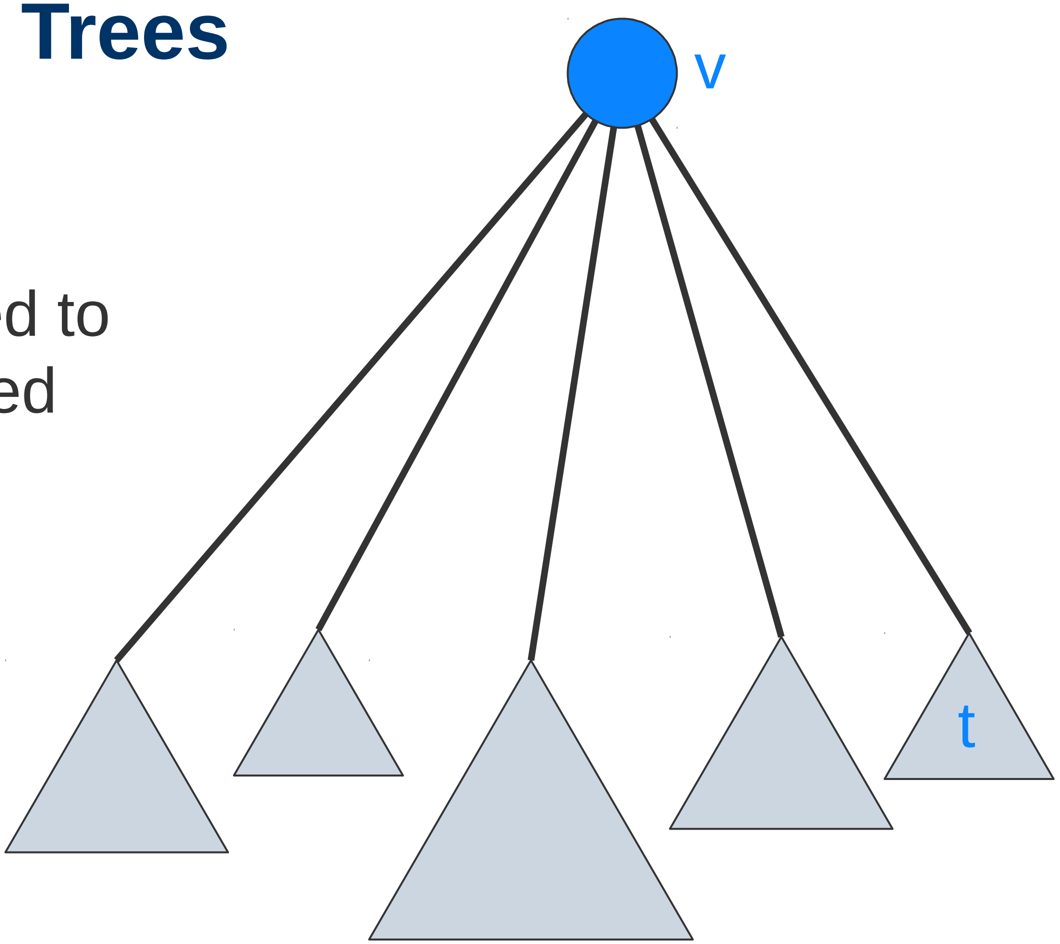
Perform two Euler-Tours in parallel

Interleave steps

If only one subtree remains, it must contain t



Example II: Paths in Trees



Running Time:

Each step can be charged to a node that is never visited again.

There are n nodes.

Total running time O(n).

We only need O(1) words of work-space.

Algorithms for Constant Work-Space

Summary:

Many geometric problems admit constant work-space algorithms.

These algorithms are often simple and effective, and lead to new perspectives on the problems.

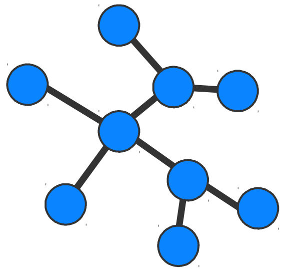
Interesting new techniques can be found.

Algorithms for Constant Work-Space

Open Problems:

Given: set S of n sites in the plane.

Question: Can we find EMST(S) in $o(n^3)$ time and constant work-space?

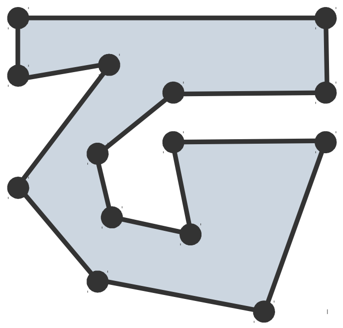


Algorithms for Constant Work-Space

Open Problems:

Given: simple polygon P with n vertices

Question: Can we find a balanced separating diagonal for P in $o(n^2)$ time?

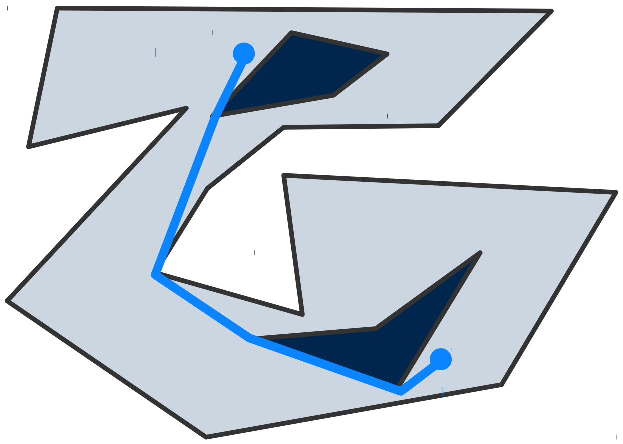


Algorithms for Constant Work-Space

Open Problems:

Given: polygon P with n vertices and h holes, points $s, t \in P$.

Question: How much space is needed to find a shortest path from s to t?



Time-Space Trade-Offs

Now we let s vary.

Goal: fastest possible algorithm for a given space budget s

Several classic results, e.g., for sorting, we can get $T \cdot s = \Theta(n^2/\log n)$, which is optimal. Beame [1991].

Asano, Buchin, Buchin, M, Rote, Schulz [2013]: some initial time-space trade-offs for geometric problems.

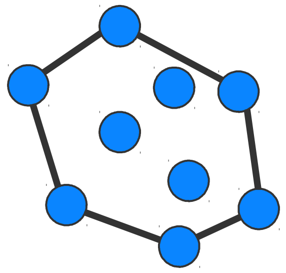
Since then: several general approaches and results.

Time-Space Trade-Offs

Examples: Convex Hulls

Given a set P of n points in the plane, and s words of work-space, we can find $\mathrm{conv}(P)$ in time $O(n^2/(s \log n) + n \log s)$.

Darwish, Elmasry [2014]



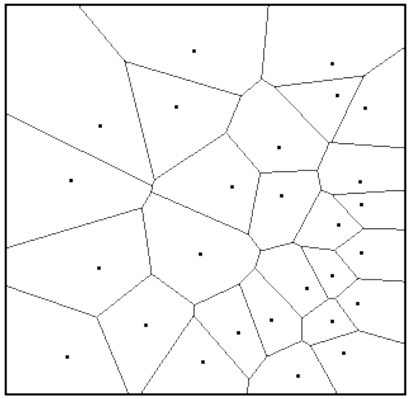
Matches the sorting bound.

Tool: Efficient implicit heap data structures

Time-Space Trade-Offs

Examples: Voronoi Diagrams

Given a set P of n points in the plane, and s words of work-space, we can find VD(P) in expected time $O((n^2/s)\log s + n \log s \log^* s)$.



Korman, M, van Renssen,Roeloffzen, Seiferth, Stein [2015]

Has now been improved.

Tool: Space-efficient implementation of the Clarkson-Shor sampling technique.

Time-Space Trade-Offs

Examples: Stack-based Algorithms

Let A be a stack-based algorithm that runs in $O(n)$ time and needs $O(n)$ words of work-space.
Then, A can be converted in an algorithm that uses s words of work-space and runs in time $O(n^2 \log n/2^s)$, for $s = o(\log n)$, and in time $n^{1+O(1/\log s)}$, for $s \geq \log n$.

Barba, Korman, Langerman, Sadakane, Silveira [2015]

Concrete examples: visibility region in a polygon, convex hull of a polygonal chain,…

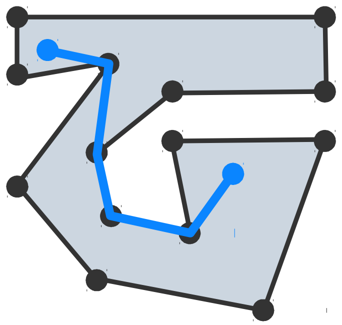
Tool: Balance storage and recomputation

Time-Space Trade-Offs

Examples: Shortest Paths in Polygons

Let P be a simple polygon with n vertices, a, b є P, and s = O(n / log n) words of work-space. We can find the geodesic shortest path from a to b in time $O(n^2/s)$.

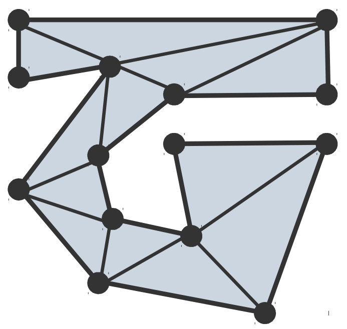
Har-Peled [2016]



Improves several previous results.

Tool: Space-efficient implementation of the violator-space framework & polygon partitioning

Time-Space Trade-Offs



Examples: Polygon Problems

Let P be a simple polygon with n vertices, and suppose we have $s = \Omega(\log n) \cap O(n)$ words of work-space. Then, we can triangulate P in expected time $O(n^2/s + n \log s \log^5(n/s))$.

Aronov, Korman, Pratt, van Renssen, Roeloffzen [2016]

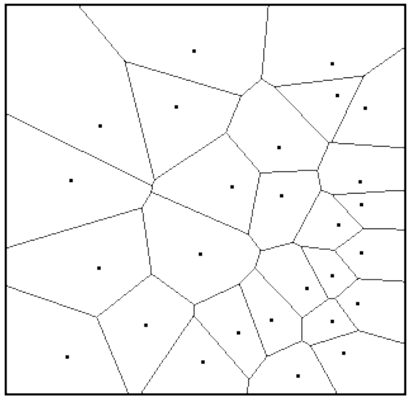
Extends to other problems in polygons.

Tool: Har-Peled's shortest path algorithm and divide & conquer.

Detailed Example: Voronoi Diagrams

Given: set S of n sites in the plane,
s words of work-space

Want: the Voronoi diagram of S



Idea BKMvRRSS [2017]: Batch processing

Detailed Example: Voronoi Diagrams

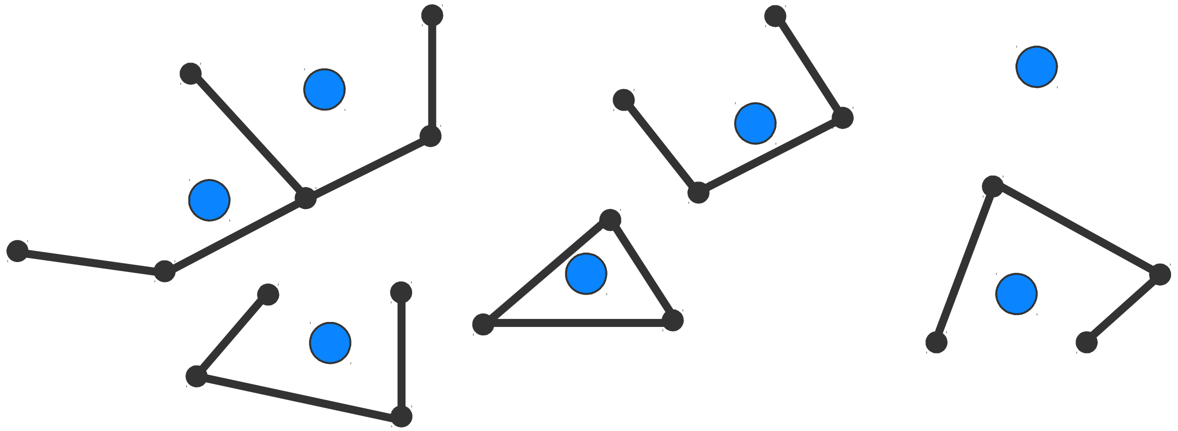
Algorithm – general structure

Algorithm proceeds in rounds

At each point, have a set Q of s active sites

In each phase, for each q ϵ Q, produce a new edge of q's Voronoi cell

If Voronoi cell of q ϵ Q is complete, replace q by a new site from S



Detailed Example: Voronoi Diagrams

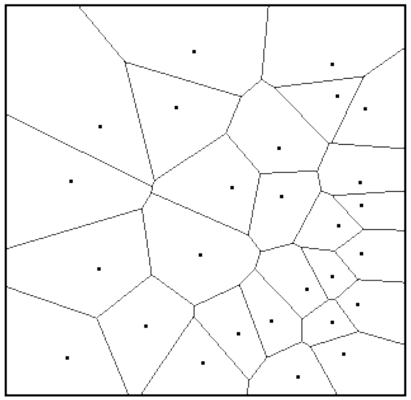
Algorithm – single round

One round can be implemented in $O(n \log s)$ time

Divide S in n/s batches of s sites

For each batch S_i, compute $VD(Q \cup S_i)$ in $O(s \log s)$ time and with $O(s)$ words of work-space

Keep track of current edge for each $q \in Q$.



Detailed Example: Voronoi Diagrams

Algorithm – putting it together

In each round, we produce s new edges, one for each active site.

Would like to stop after n/s rounds.

Problem: sites may have Voronoi cells with more then n/s edges

But: there are only O(s) such big sites.

Detailed Example: Voronoi Diagrams

Algorithm – putting it together

To handle big sites, we perform another round with the big sites, to identify all the Voronoi edges incident to two big sites.

Through appropriate tie breaking, we can avoid reporting an edge multiple times.

Total running time: $O((n^2/s)\log s)$

Space: $O(s)$ words.

By adjusting constants, we can reduce it to s

Generalizes to farthest site and higher order diagrams.

Time-Space Trade-Offs

Summary:

A lot of interesting trade-offs are possible.

Can observe a rich variety of dependencies on s.

Several general methods

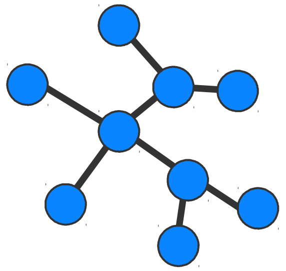
Sophisticated techniques

Time-Space Trade-Offs

Open Problems:

Given: set S of n sites in the plane.

Question: Is there a time-space trade-off for computing EMST(S) ?

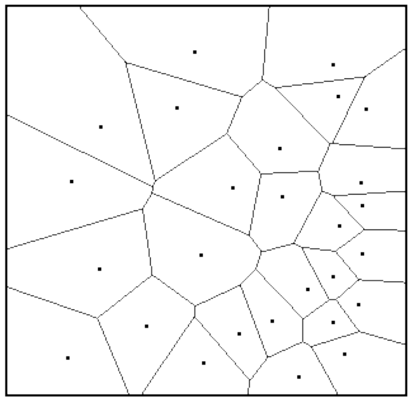


Time-Space Trade-Offs

Open Problems:

Given: set P of n points in the plane.

Question: Can you prove a time-space trade-off lower bound for computing $VD(P)$?

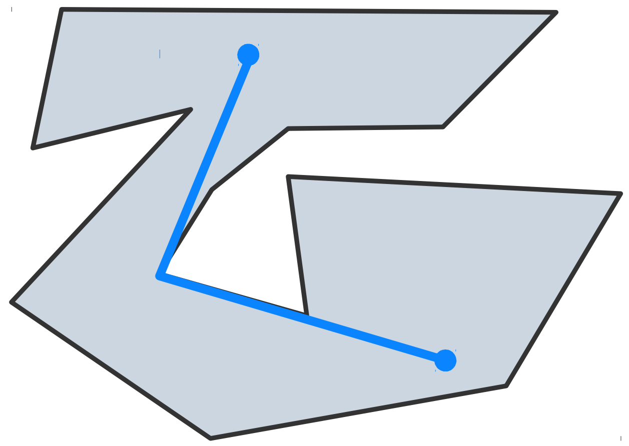


Time-Space Trade-Offs

Open Problems:

Given: simple polygon P with n vertices
points s, t ϵ P.

Question: Is there a simple time-space trade-off for finding a shortest path from s to t?



Additional Challenges

Say more about the role of randomness

A systematic study of lower bounds

Implementations and experiments

Questions?