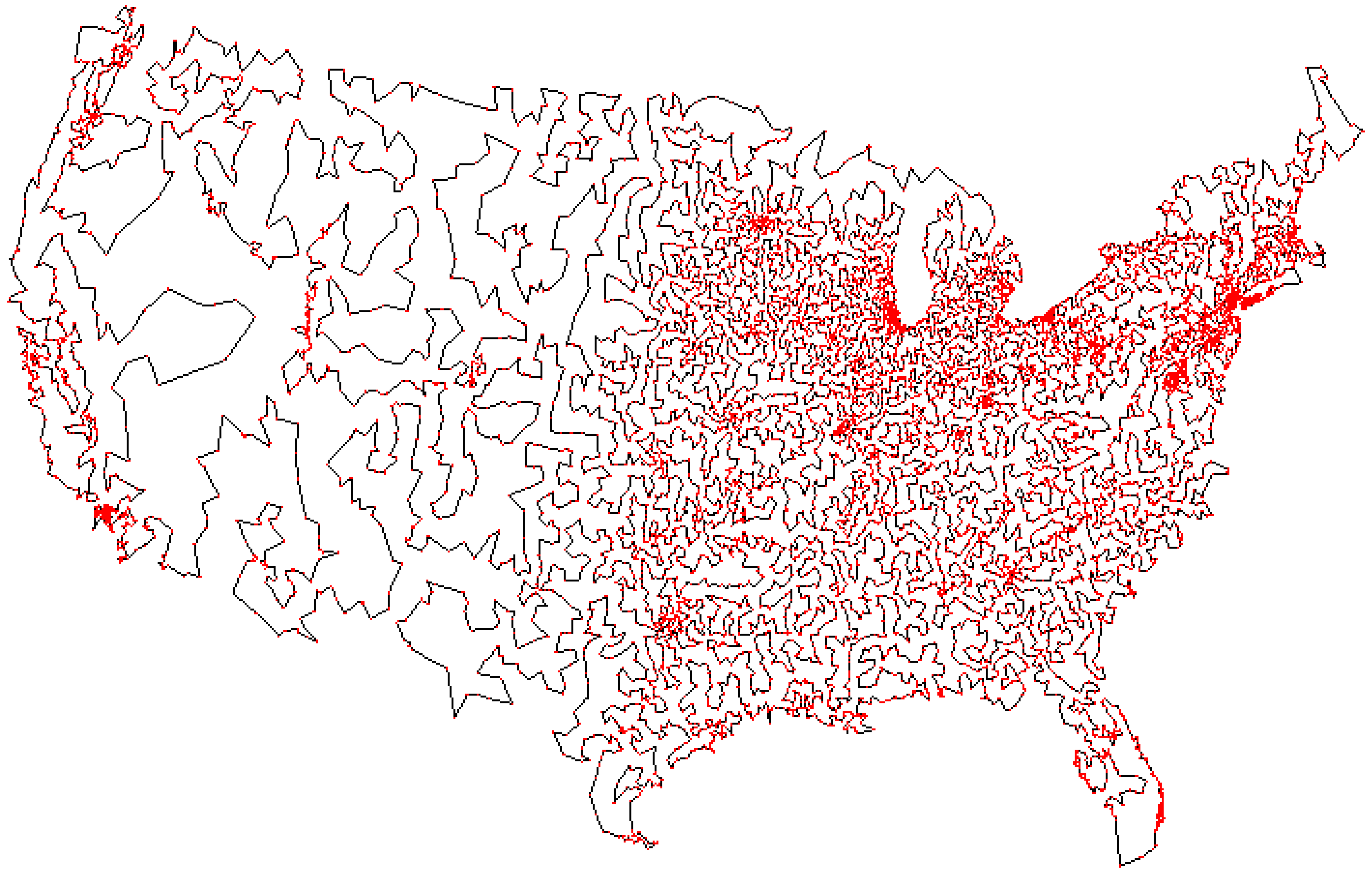
PTAS for ETSP

Polynomial-time approximation scheme for the Euclidean Traveling Salesperson Problem

The Problem

Euclidean Traveling Salesperson Problem (ETSP)

Input: set of points P in $\mathbb{R}^2$



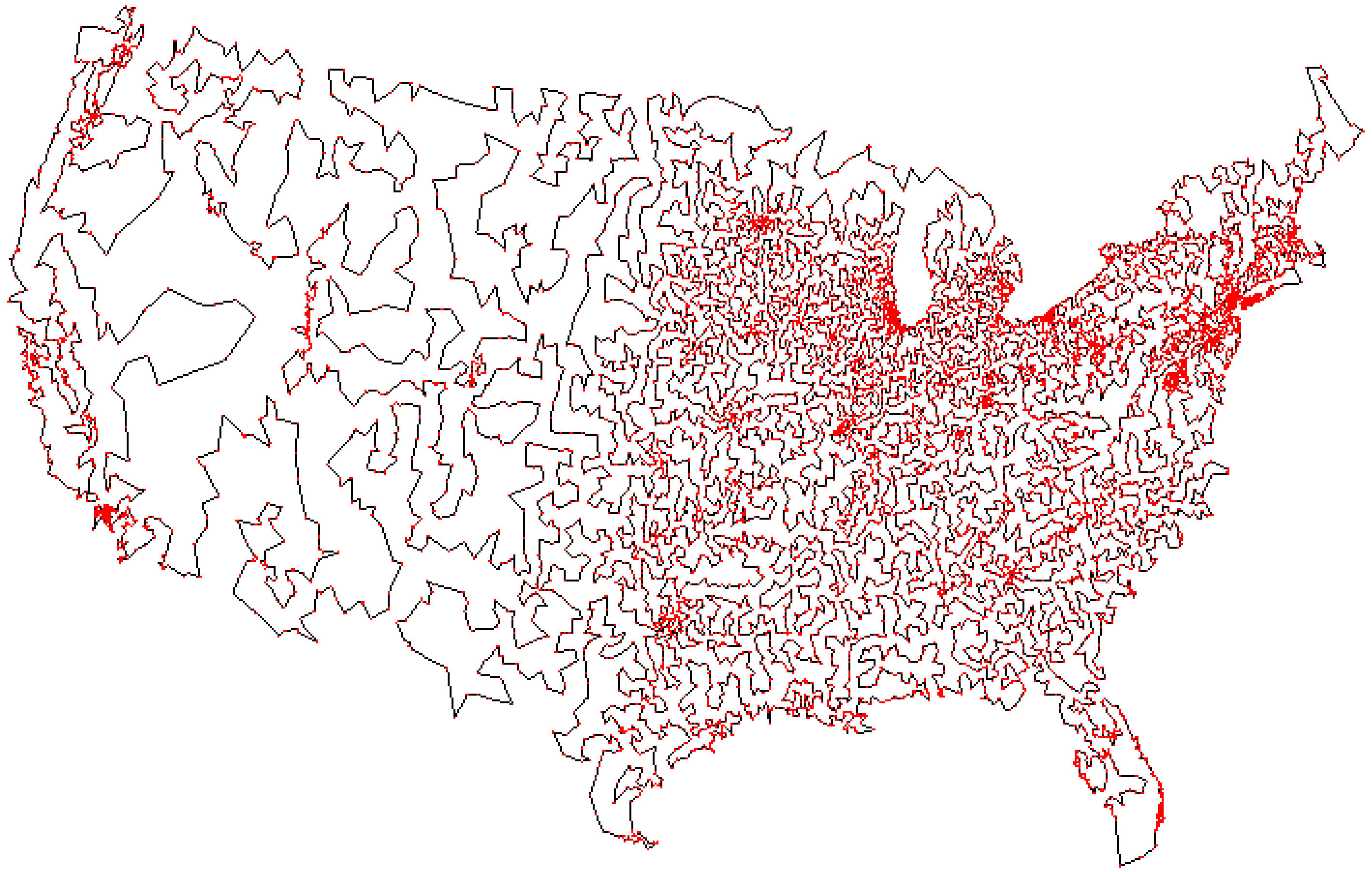
Output: a shortest TSP tour of P

The Problem

Euclidean Traveling Salesperson Problem (ETSP)

Input: set of points P in $\mathbb{R}^2$

Output: a shortest TSP tour of P

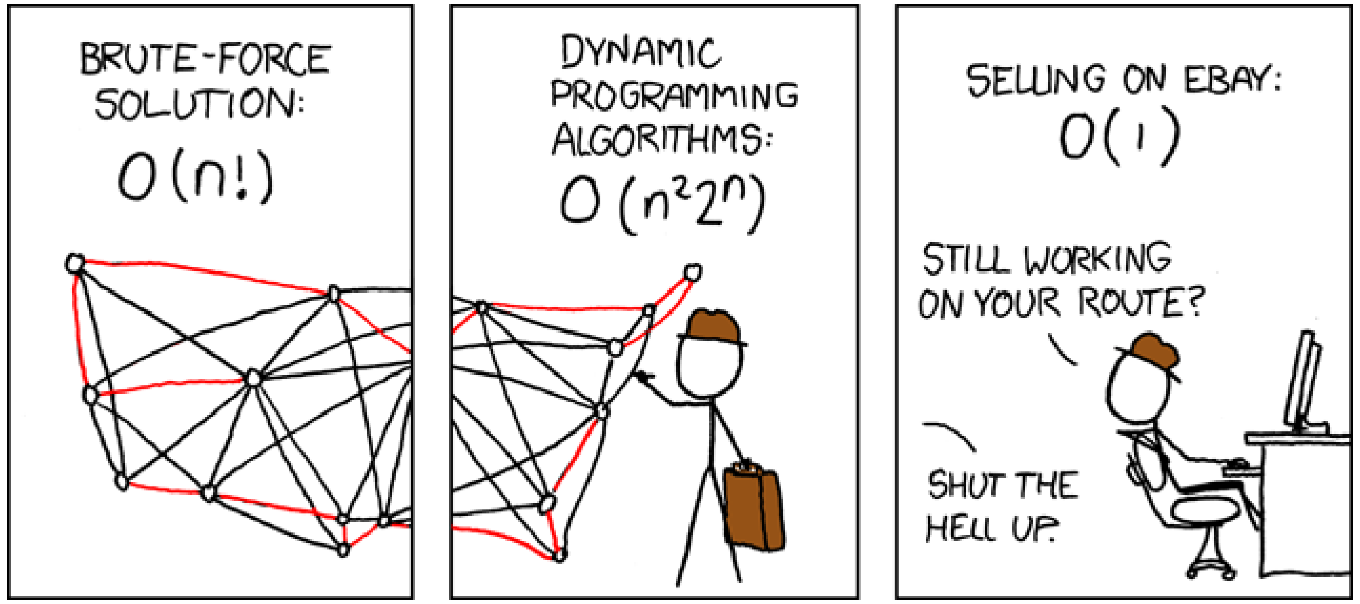


Today: A polynomial-time approximation scheme (PTAS) for the ETSP

Gives for any fixed $\varepsilon > 0$ a polynomial-time $(1+\varepsilon)$-algorithm

Motivation

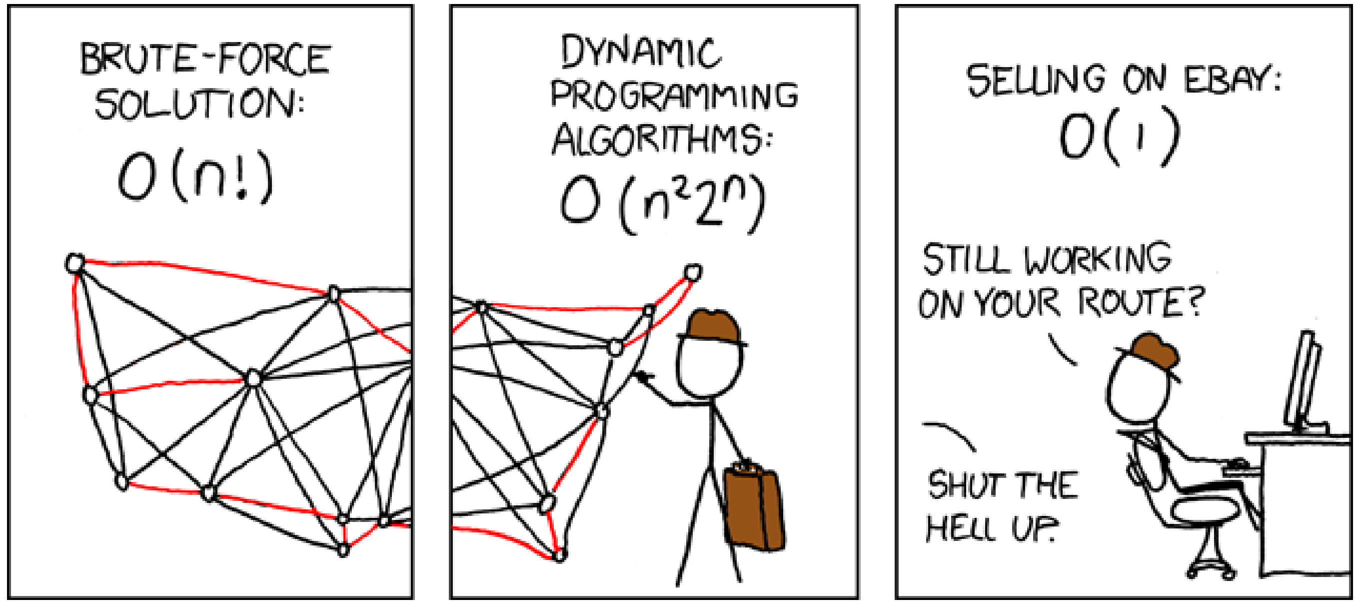
TSP: classic NP-hard optimization problem



https://xkcd.com/399/

Motivation

TSP: classic NP-hard optimization problem



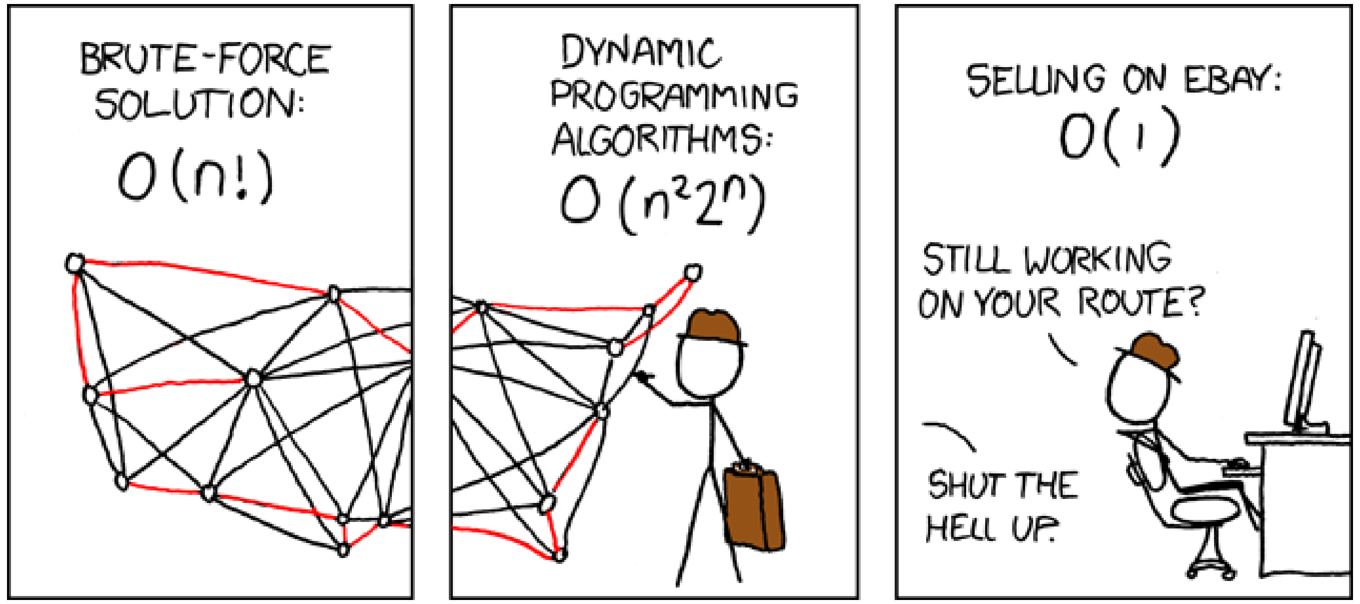
https://xkcd.com/399/

Polynomial time approximation algorithms for TSP

- for general TSP: not possible

Motivation

TSP: classic NP-hard optimization problem



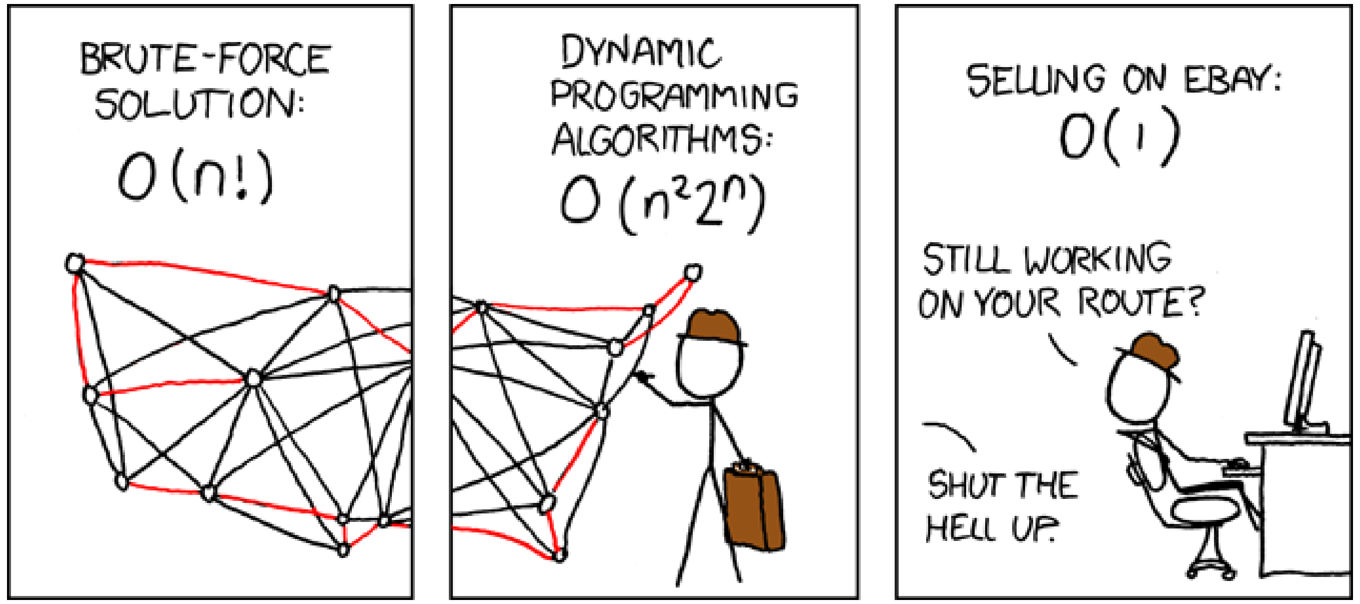
https://xkcd.com/399/

Polynomial time approximation algorithms for TSP

- for general TSP: not possible
- for metric TSP: not possible with approximation factor $< 123/122 \approx 1.008$, Christofides: 1.5-approximation, first improved in 2020: $1.5 - 10^{-36}$

Motivation

TSP: classic NP-hard optimization problem



https://xkcd.com/399/

Polynomial time approximation algorithms for TSP

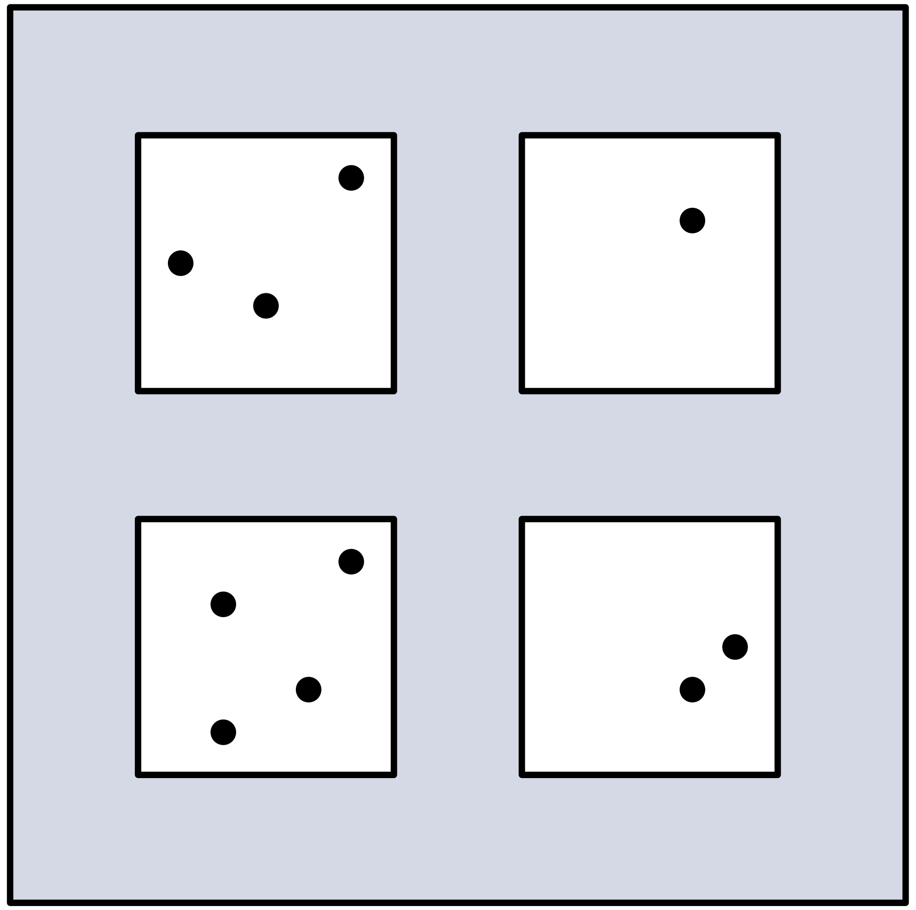
- for general TSP: not possible
- for metric TSP: not possible with approximation factor $< 123/122 \approx 1.008$, Christofides: 1.5-approximation, first improved in 2020: $1.5 - 10^{-36}$
- for ETSP: PTASs by Sanjeev Arora [1998] and Joe Mitchell [1999] $\rightarrow$ Gödel prize in 2010
  today: Arora's algorithm

Overview

1. Intuition
2. Subproblems
3. Algorithm
4. Running time
5. Quality of approximation

Intuition - Approach

Dynamic programming on quadtrees

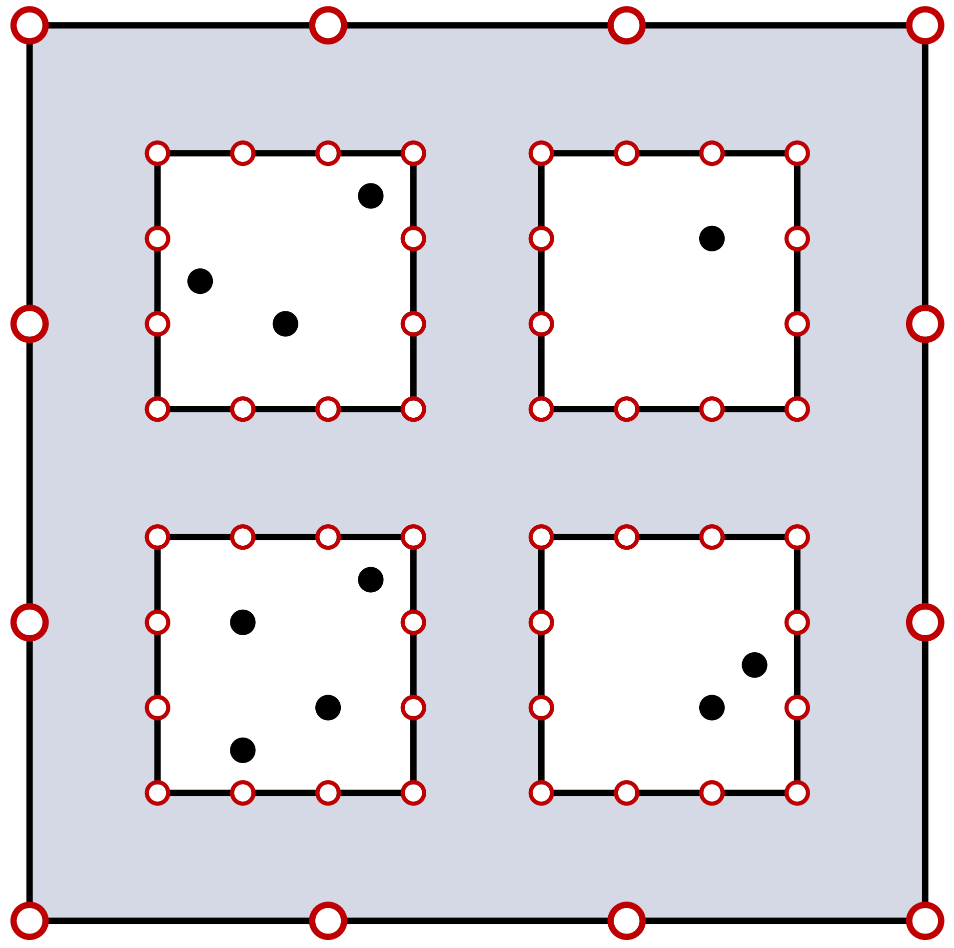


Intuition - Approach

Dynamic programming on quadtrees

Portals on the boundaries

Evenly placed and on each corner



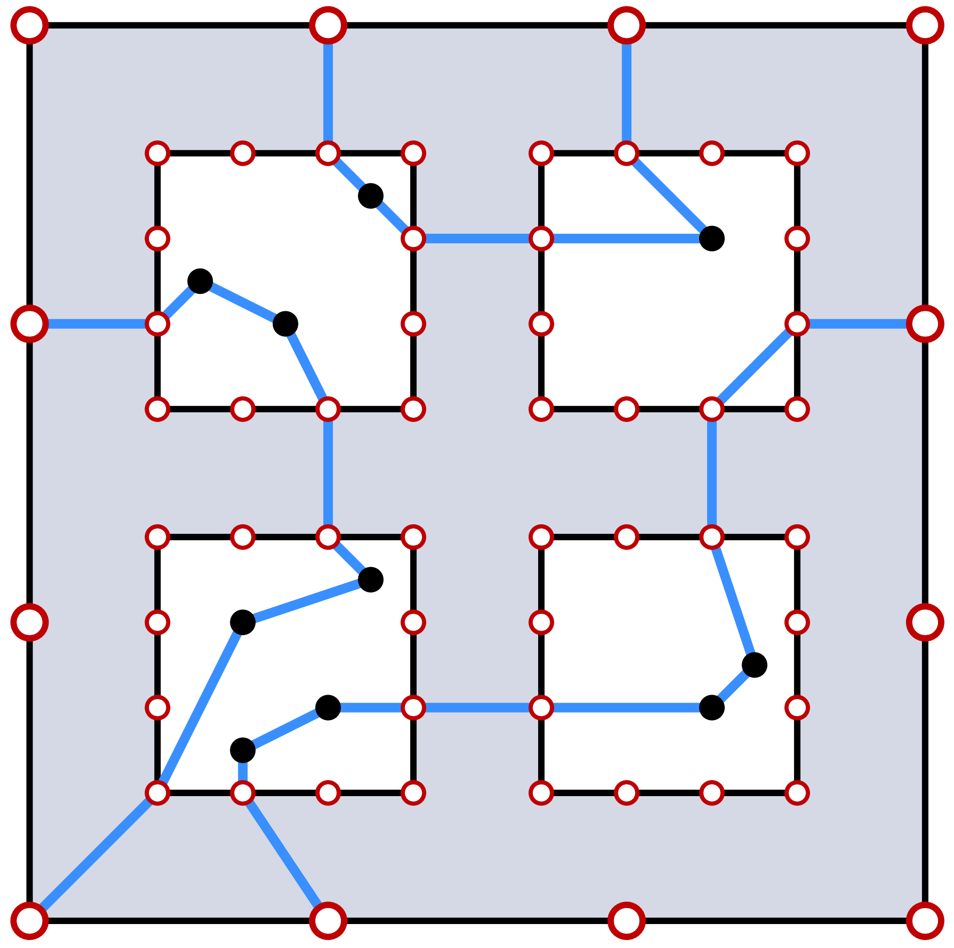
Intuition - Approach

Dynamic programming on quadtrees

Portals on the boundaries

Evenly placed and on each corner

Move between squares only through portals



Intuition - Approach

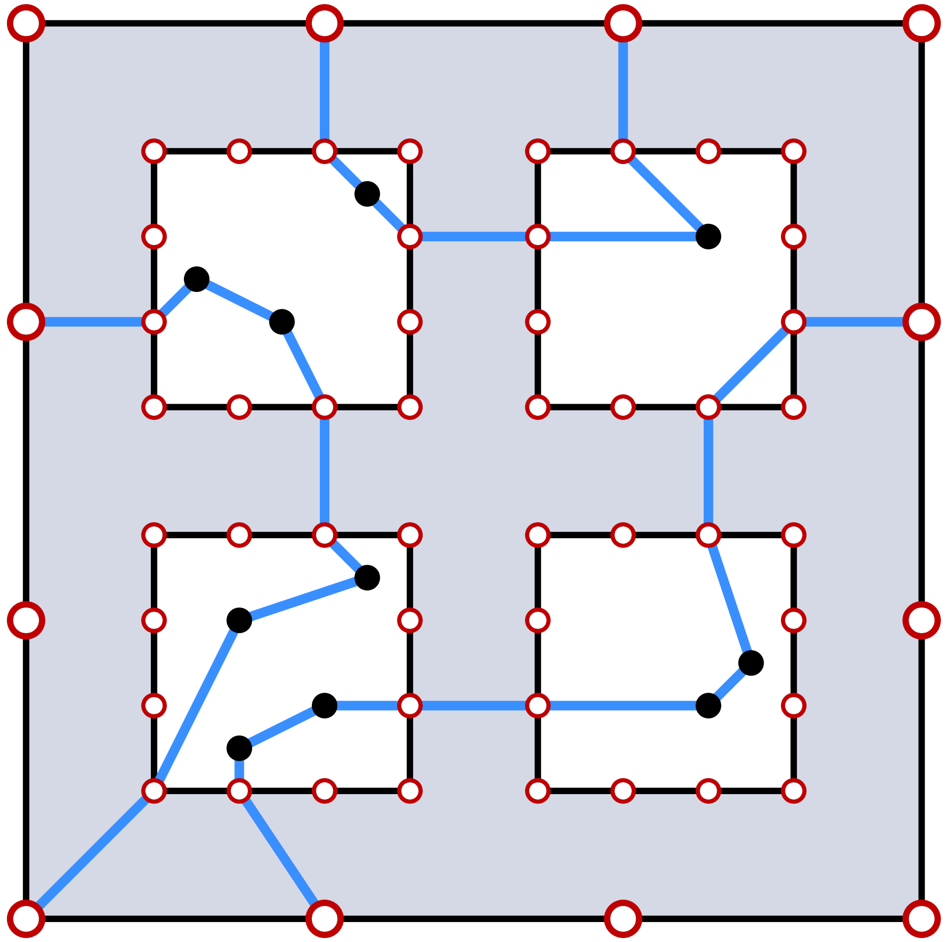
Dynamic programming on quadtrees

Portals on the boundaries

Evenly placed and on each corner

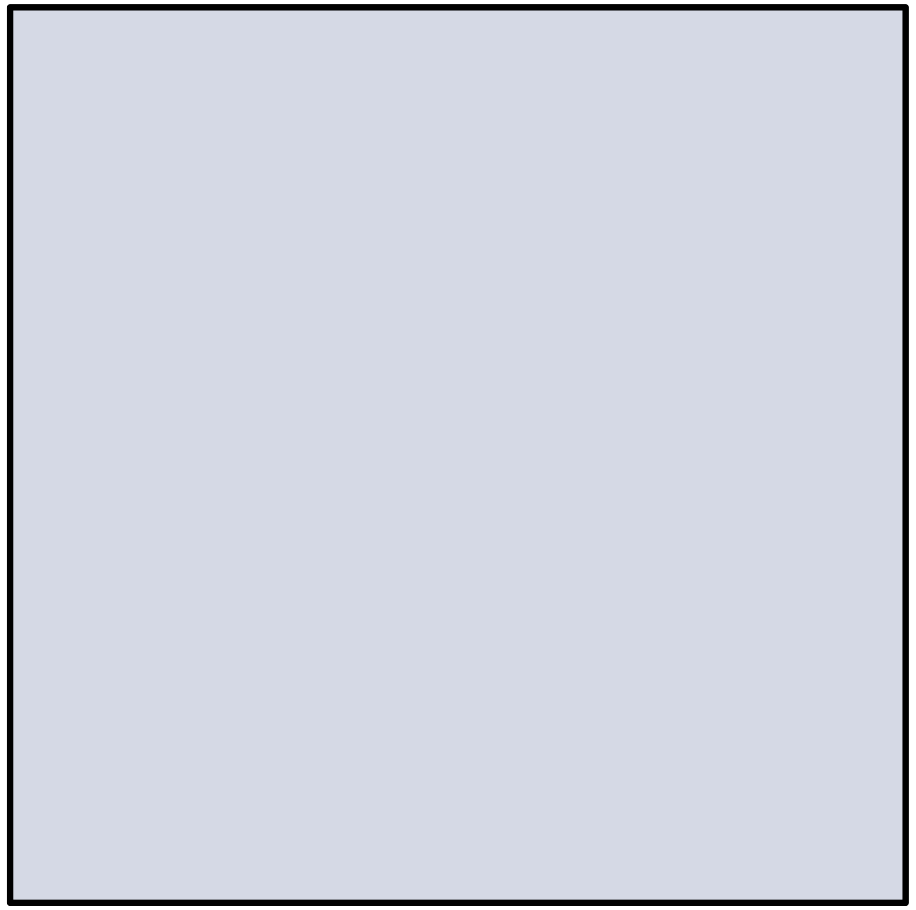
Move between squares only through portals

What defines a subproblem?



Subproblems

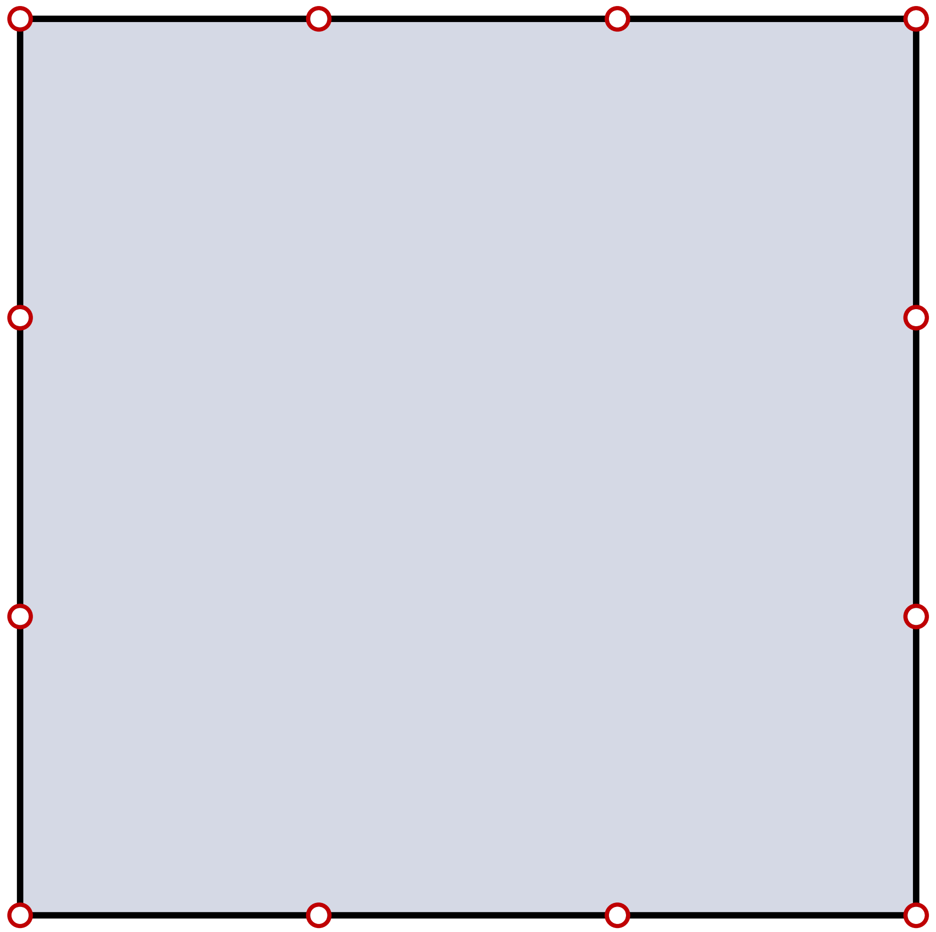
Square S



Subproblems

Square S

The $m + 2$ portals on each edge of S

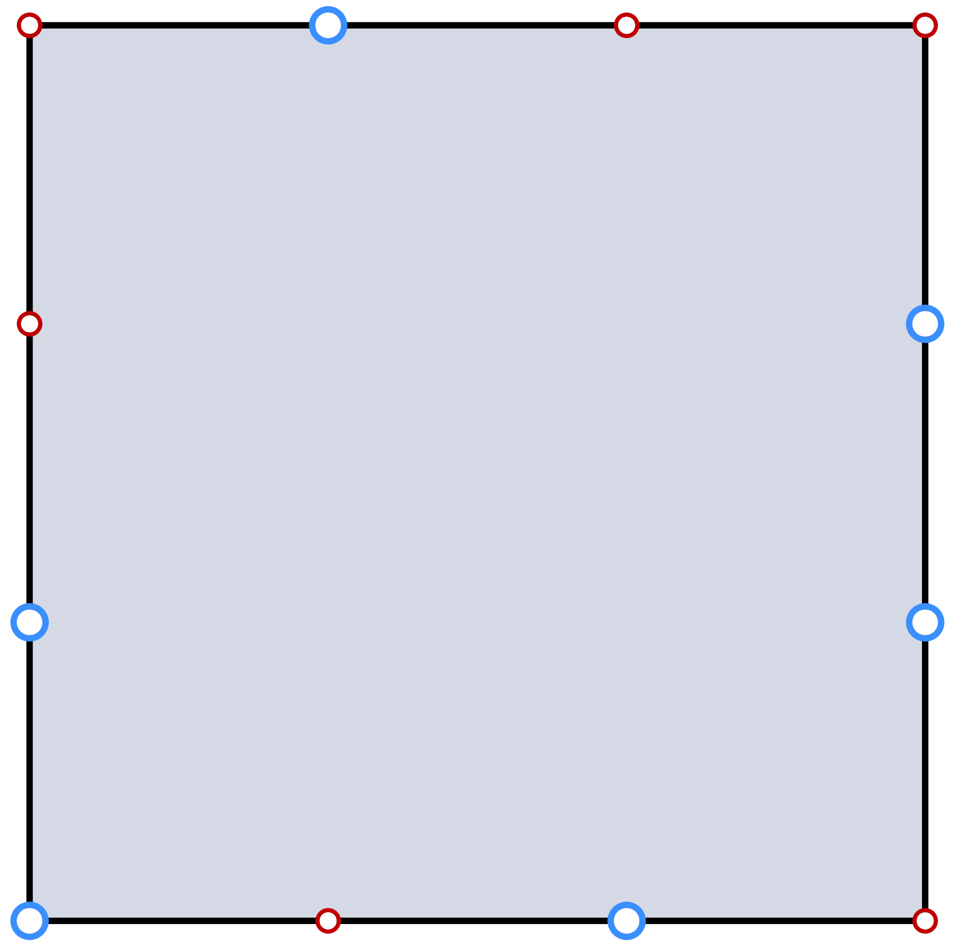


Subproblems

Square S

The $m + 2$ portals on each edge of S

Which portals are used



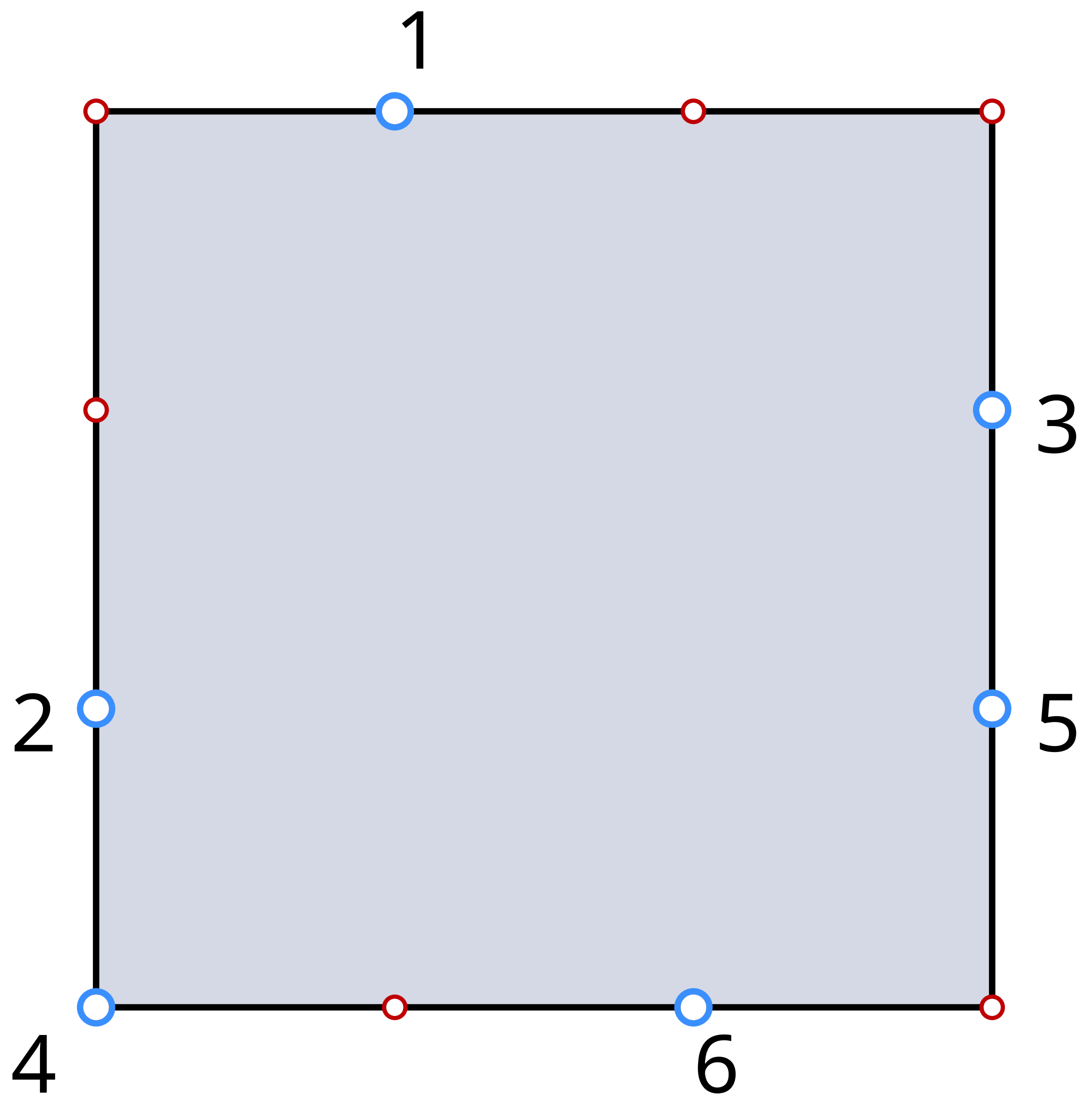
Subproblems

Square S

The $m + 2$ portals on each edge of S

Which portals are used

Order M in which portals are used



Subproblems

Square S

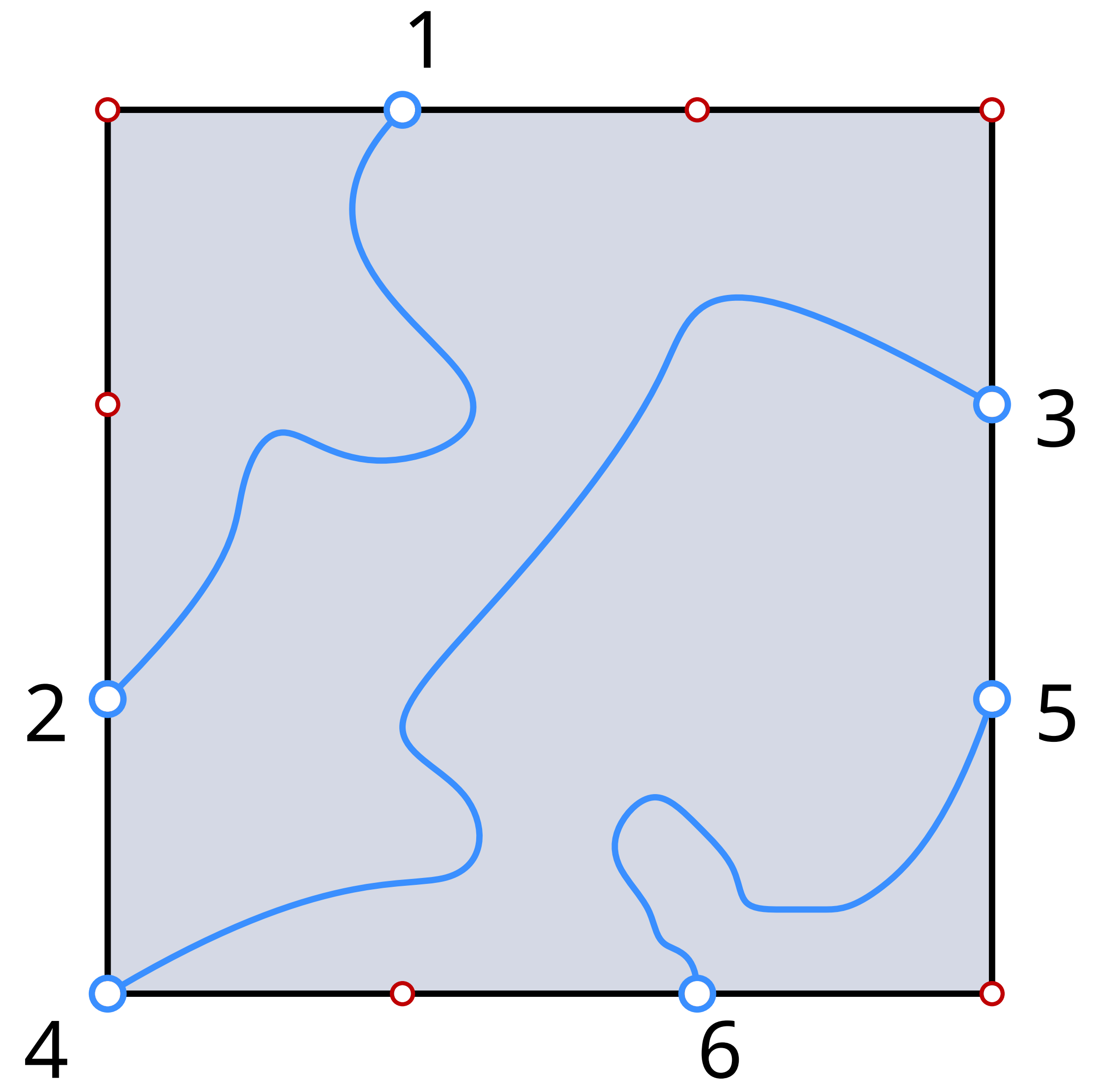
The $m+2$ portals on each edge of S

Which portals are used

Order M in which portals are used

Subproblem denoted (S, M)

Solution: length of the shortest partial tour abiding by M



Subproblems

Square S

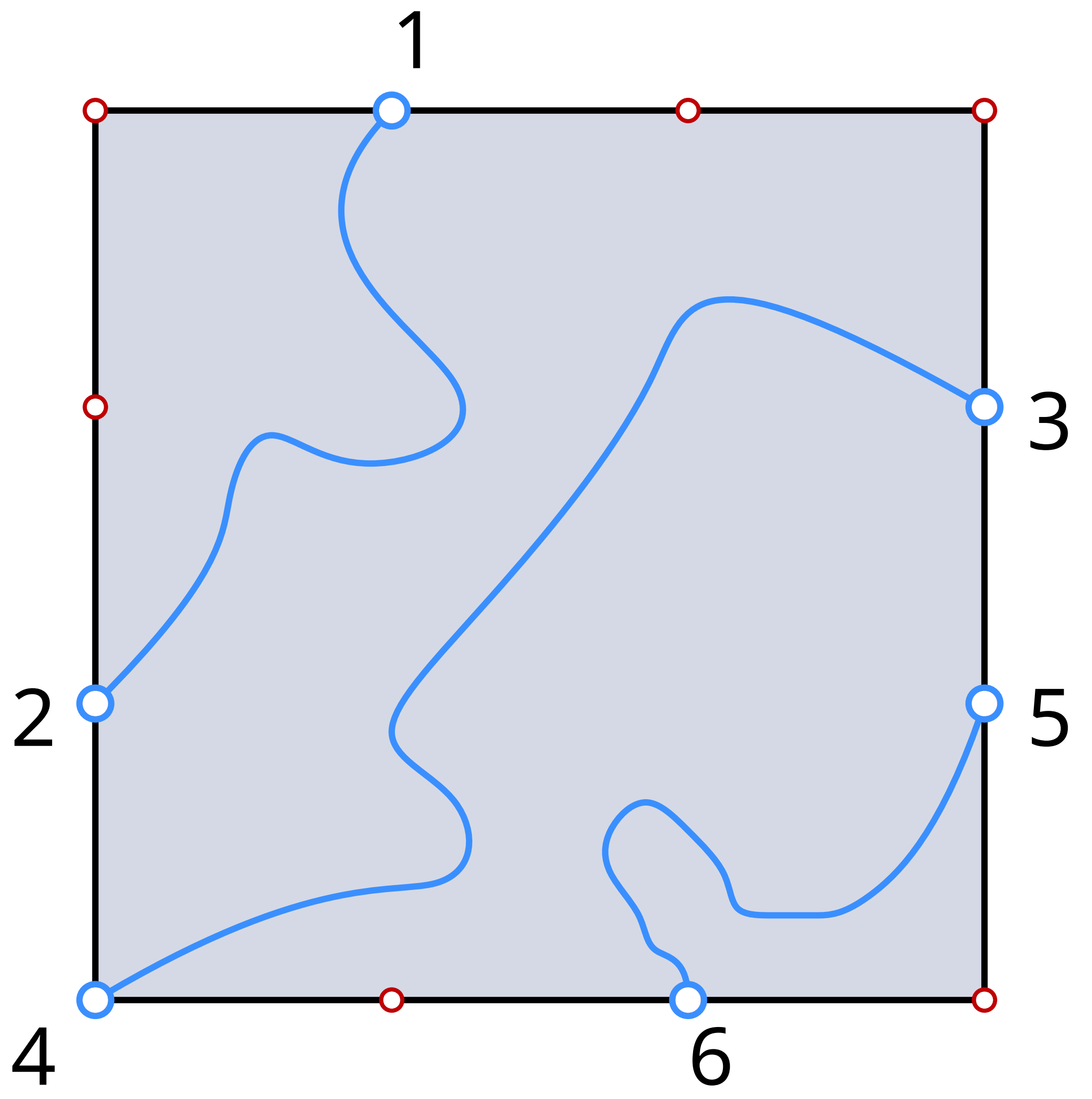
The $m+2$ portals on each edge of S

Which portals are used

Order M in which portals are used

Subproblem denoted (S, M)

Solution: length of the shortest partial tour abiding by M



Restrictions:

The solution takes each portal at most twice (patching lemma $\to$ later)

Per side of S at most k portals can be used

Overview

1. Intuition ✓
2. Subproblems ✓
3. Algorithm
4. Running time
5. Quality of approximation

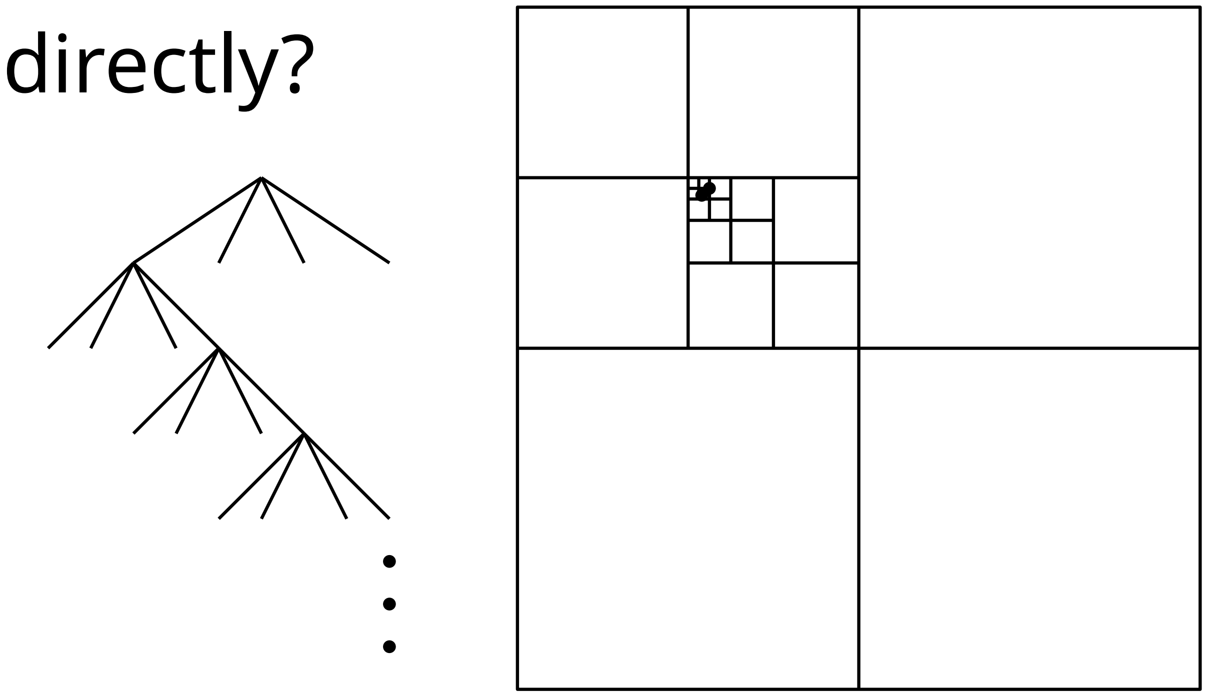
A Good Quadtree

Problems with constructing a quadtree on P directly?

A Good Quadtree

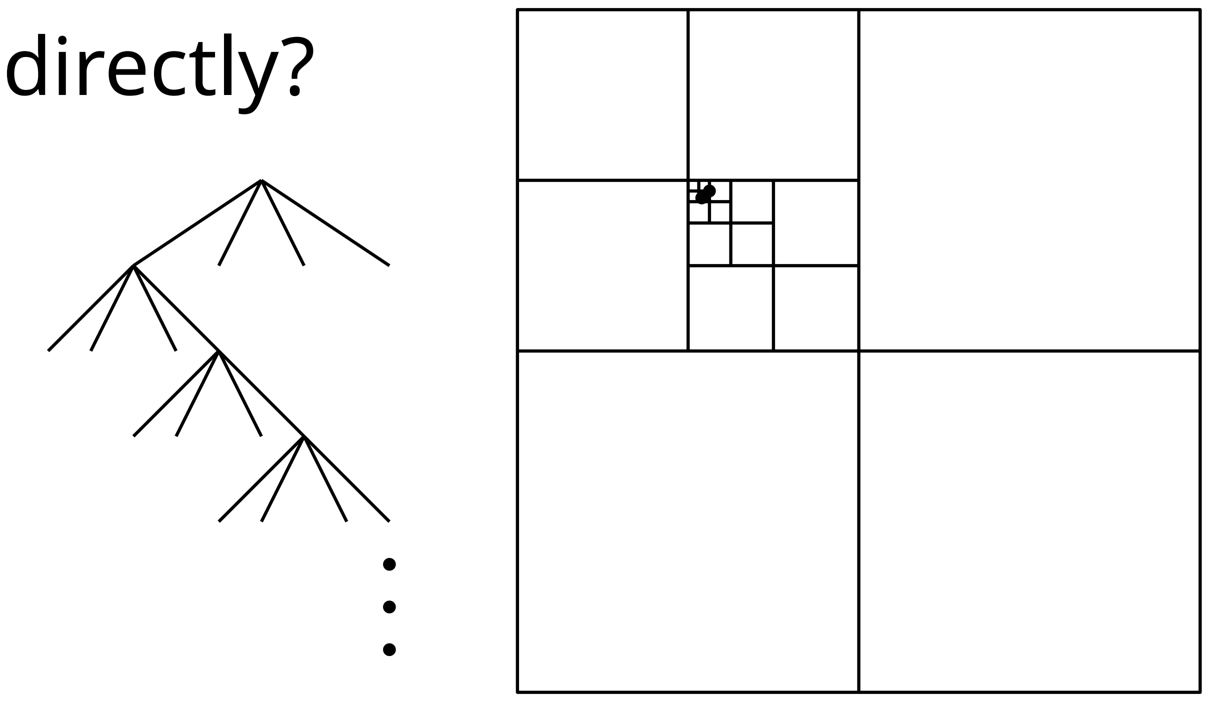
Problems with constructing a quadtree on P directly?

1. Depth not bounded



A Good Quadtree

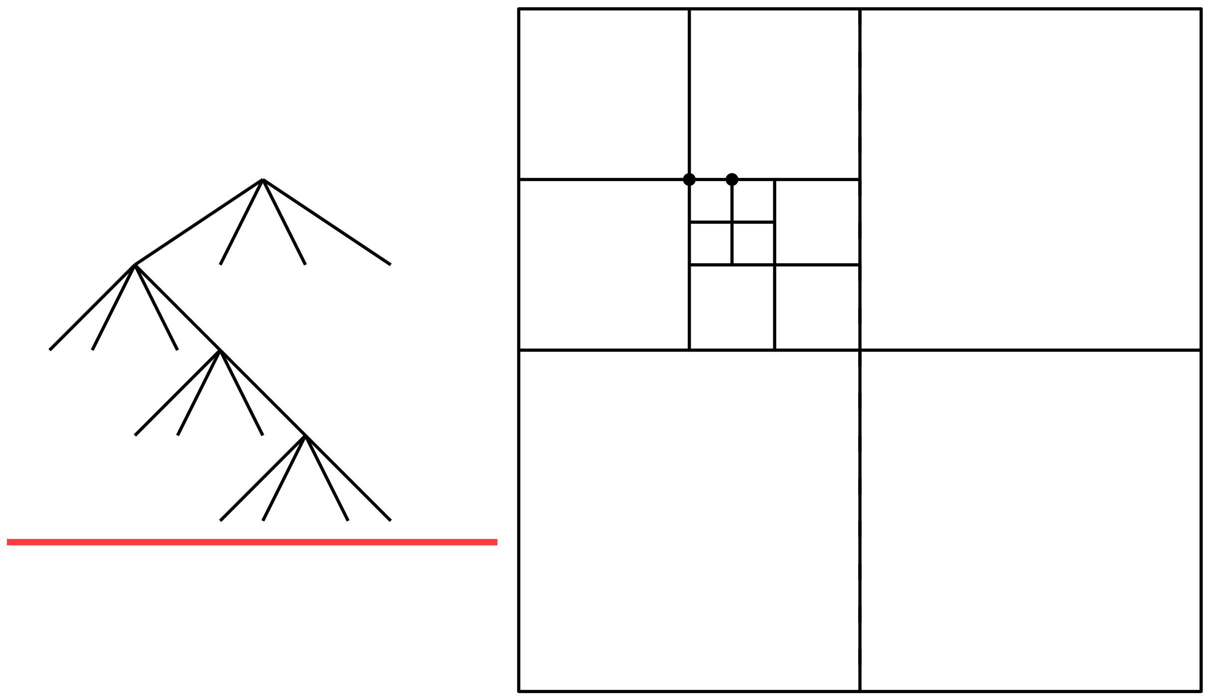
Problems with constructing a quadtree on P directly?



1. Depth not bounded

Solution

1. Snap points to a grid



A Good Quadtree

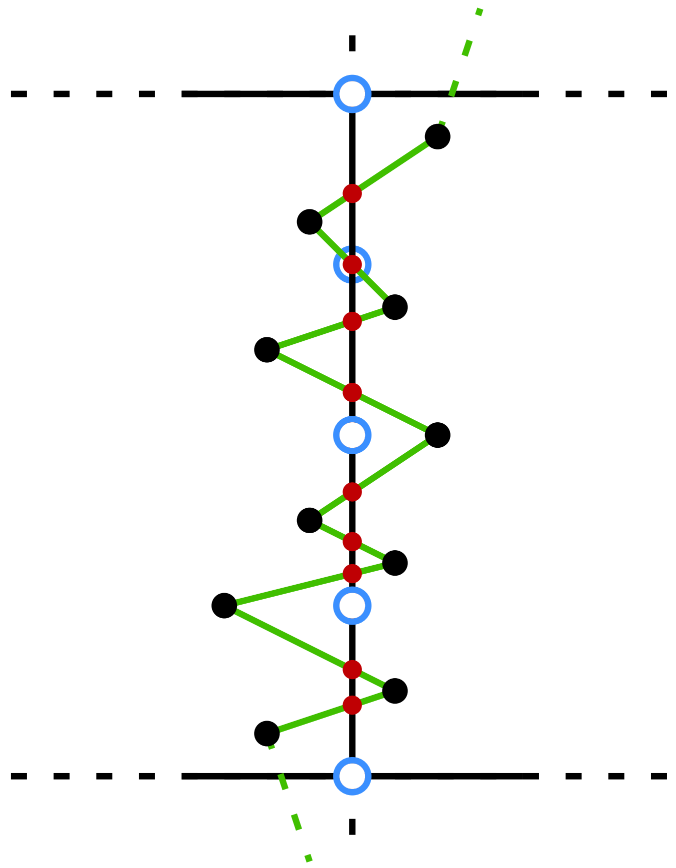
Problems with constructing a quadtree on P directly?

1. Depth not bounded

2. When short tour edges intersect long quadtree edges: next portal may be far away

Solution

1. Snap points to a grid



A Good Quadtree

Problems with constructing a quadtree on P directly?

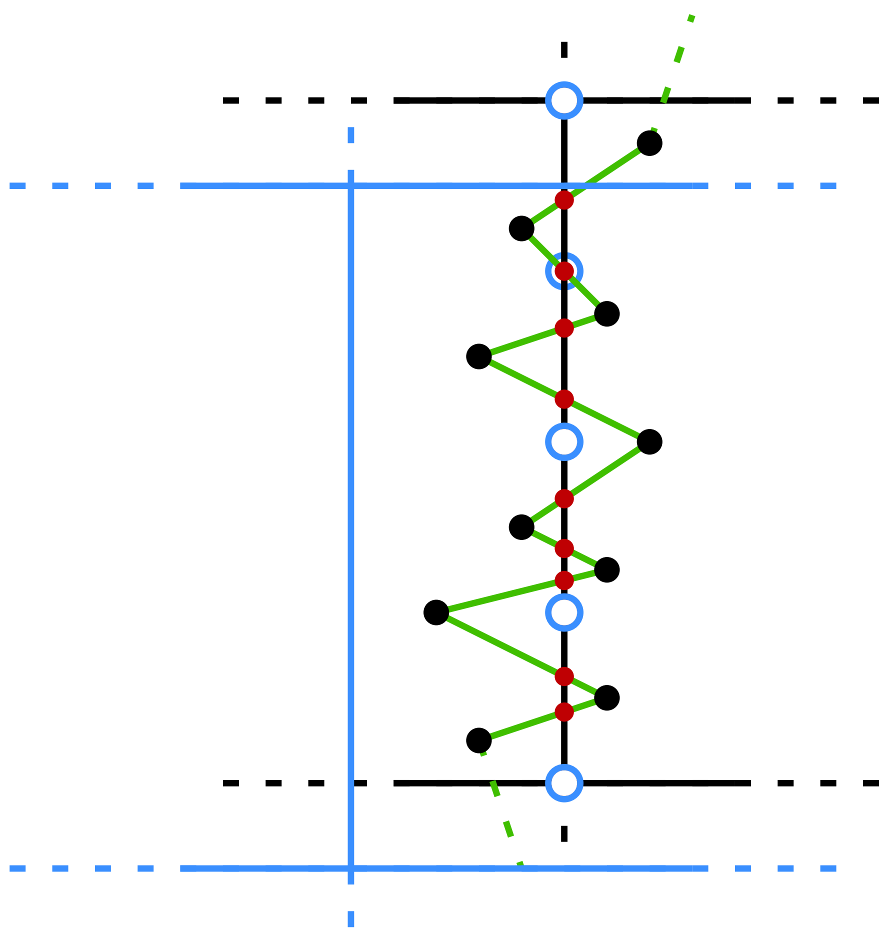
1. Depth not bounded

2. When short tour edges intersect long quadtree edges:
next portal may be far away

Solution

1. Snap points to a grid

2. Randomize the position of the initial square



A Good Quadtree - Grid

Restrictions on the problem

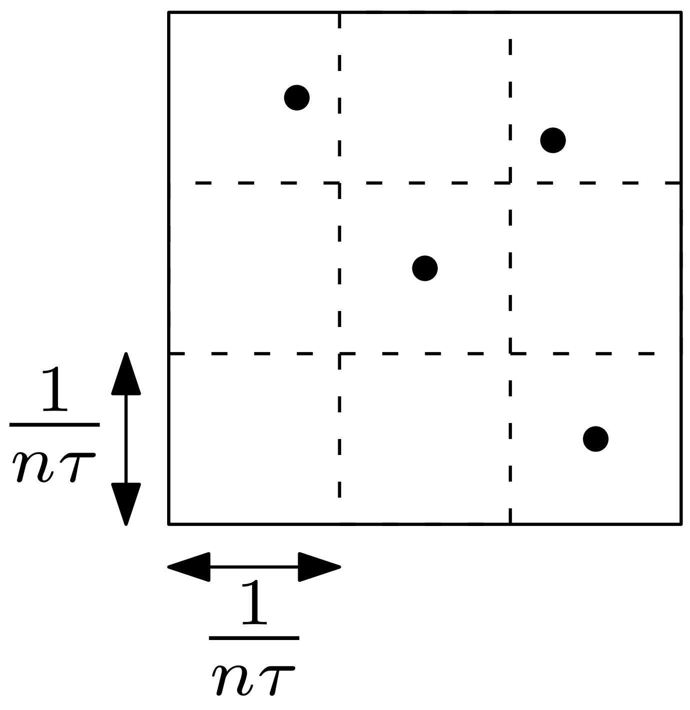
- P is contained in $[\frac{1}{2}, 1]^2$ and has diameter at least $\frac{1}{4}$
- $\frac{1}{n} < \varepsilon < 1$ the approximation factor where $n = |P|$

A Good Quadtree - Grid

Restrictions on the problem

- P is contained in $[\frac{1}{2}, 1]^2$ and has diameter at least $\frac{1}{4}$
- $\frac{1}{n} < \varepsilon < 1$ the approximation factor where $n = |P|$

Grid of cells of width $\frac{1}{n\tau}$ where $\tau = \lceil \frac{32}{\varepsilon} \rceil$



A Good Quadtree - Grid

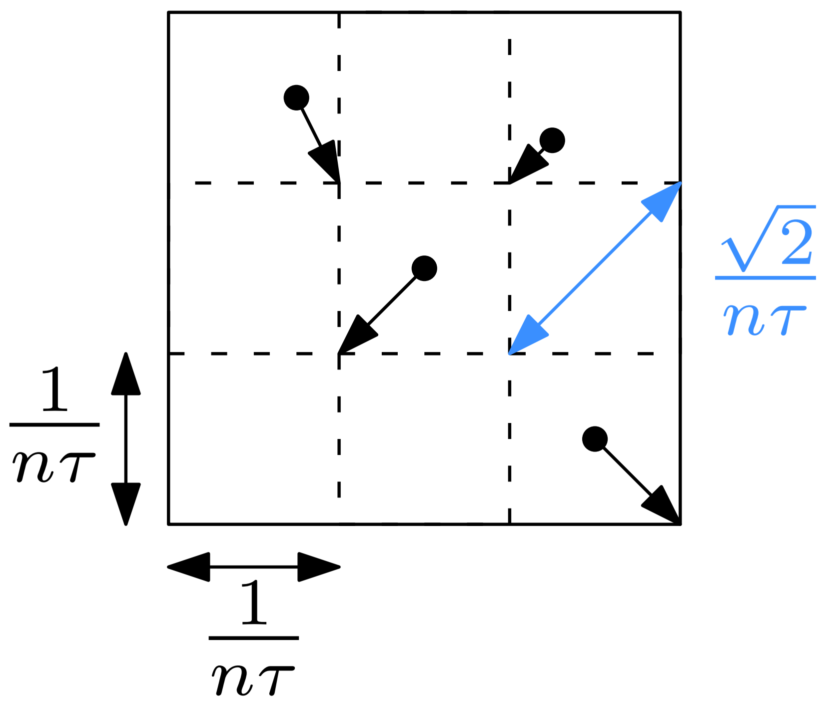
Restrictions on the problem

- P is contained in $[\frac{1}{2}, 1]^2$ and has diameter at least $\frac{1}{4}$
- $\frac{1}{n} < \varepsilon < 1$ the approximation factor where $n = |P|$

Grid of cells of width $\frac{1}{n\tau}$ where $\tau = \left\lceil \frac{32}{\varepsilon} \right\rceil$

Let Q be P with snapped to nearest gridpoints

Each point was moved at most $\frac{\sqrt{2}}{2n\tau}$



A Good Quadtree - Grid

Restrictions on the problem

- P is contained in $[\frac{1}{2}, 1]^2$ and has diameter at least $\frac{1}{4}$
- $\frac{1}{n} < \varepsilon < 1$ the approximation factor where $n = |P|$

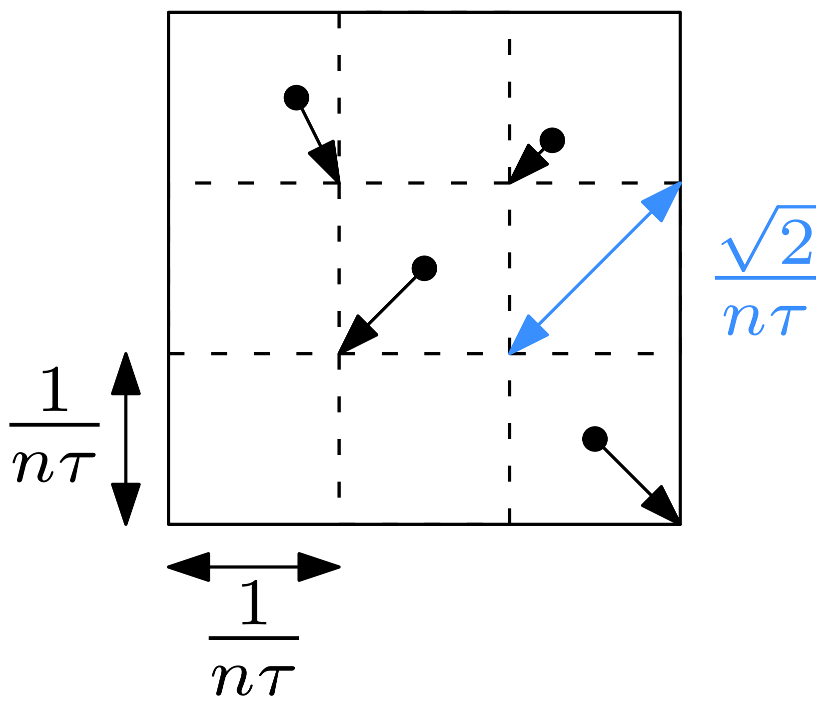
Grid of cells of width $\frac{1}{n\tau}$ where $\tau = \lceil \frac{32}{\varepsilon} \rceil$

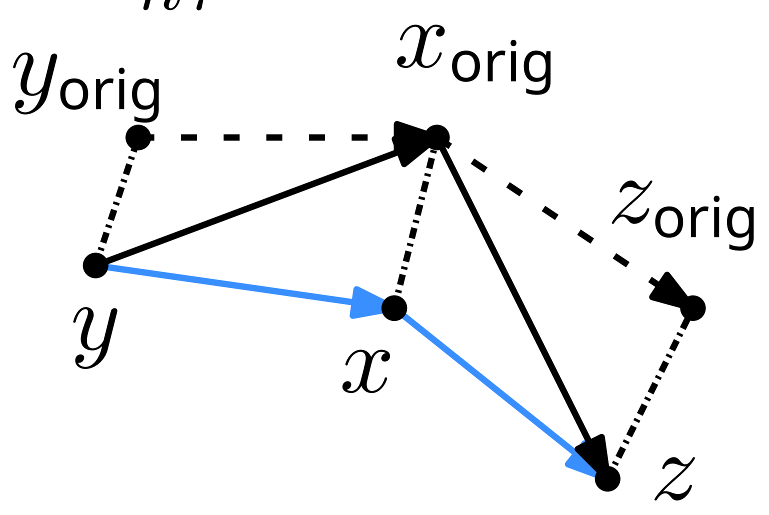
Let Q be P with snapped to nearest gridpoints

Each point was moved at most $\frac{\sqrt{2}}{2n\tau}$

Solution for Q can be converted to solution for P

Additional cost: $\leq 2n \cdot \frac{\sqrt{2}}{2n\tau}$





A Good Quadtree - Grid

Restrictions on the problem

- P is contained in $[\frac{1}{2}, 1]^2$ and has diameter at least $\frac{1}{4}$
- $\frac{1}{n} < \varepsilon < 1$ the approximation factor where $n = |P|$

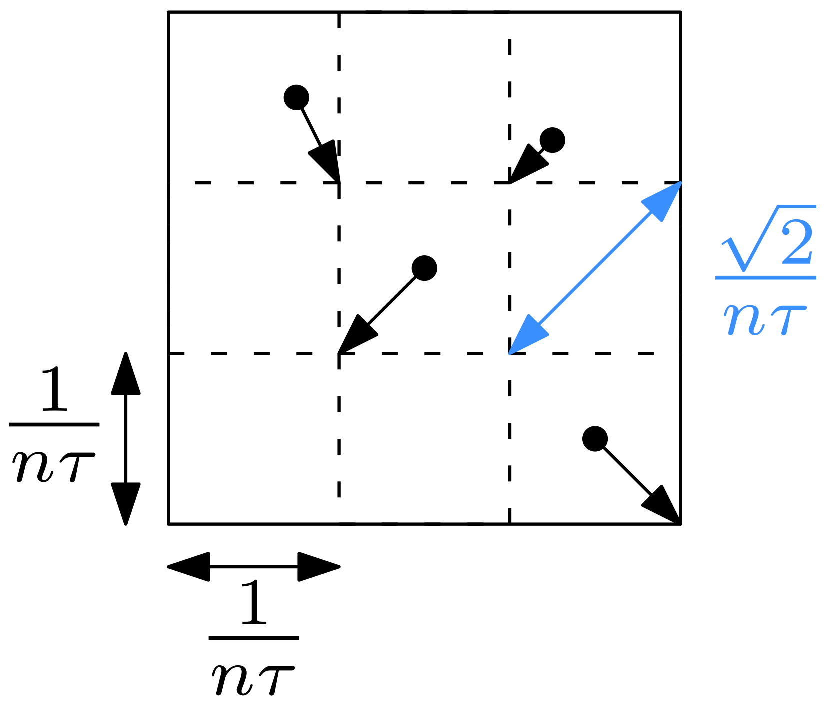
Grid of cells of width $\frac{1}{n\tau}$ where $\tau = \lceil \frac{32}{\varepsilon} \rceil$

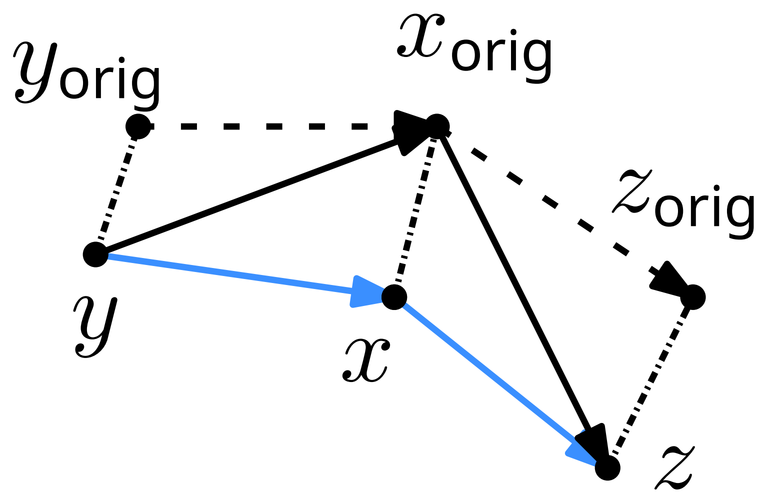
Let Q be P with snapped to nearest gridpoints

Each point was moved at most $\frac{\sqrt{2}}{2n\tau}$

Solution for Q can be converted to solution for P

Additional cost: $\leq 2n \cdot \frac{\sqrt{2}}{2n\tau} \leq \frac{\sqrt{2}}{\tau}$





A Good Quadtree - Grid

Restrictions on the problem

- P is contained in $[\frac{1}{2}, 1]^2$ and has diameter at least $\frac{1}{4}$
- $\frac{1}{n} < \varepsilon < 1$ the approximation factor where $n = |P|$

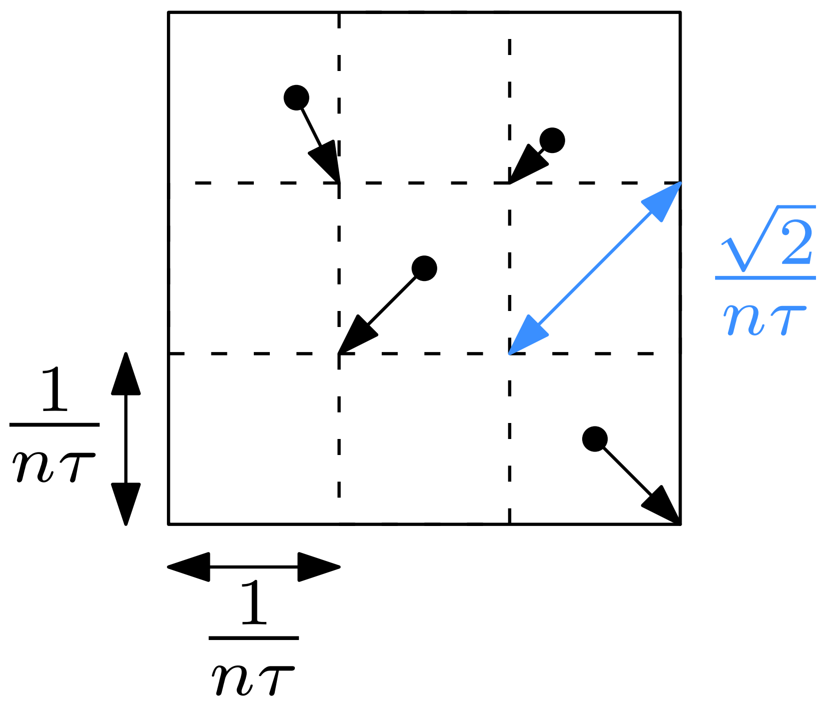
Grid of cells of width $\frac{1}{n\tau}$ where $\tau = \lceil \frac{32}{\varepsilon} \rceil$

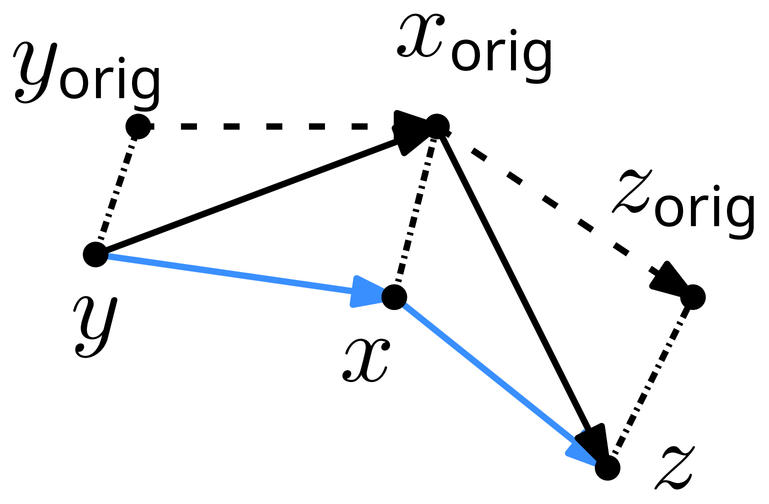
Let Q be P with snapped to nearest gridpoints

Each point was moved at most $\frac{\sqrt{2}}{2n\tau}$

Solution for Q can be converted to solution for P

Additional cost: $\leq 2n \cdot \frac{\sqrt{2}}{2n\tau} \leq \frac{\sqrt{2}}{\tau} \leq \frac{4\varepsilon}{32} \leq \frac{\varepsilon}{2} \cdot \frac{1}{4}$





A Good Quadtree - Grid

Restrictions on the problem

- P is contained in $[\frac{1}{2}, 1]^2$ and has diameter at least $\frac{1}{4}$
- $\frac{1}{n} < \varepsilon < 1$ the approximation factor where $n = |P|$

Grid of cells of width $\frac{1}{n\tau}$ where $\tau = \left\lceil \frac{32}{\varepsilon} \right\rceil$

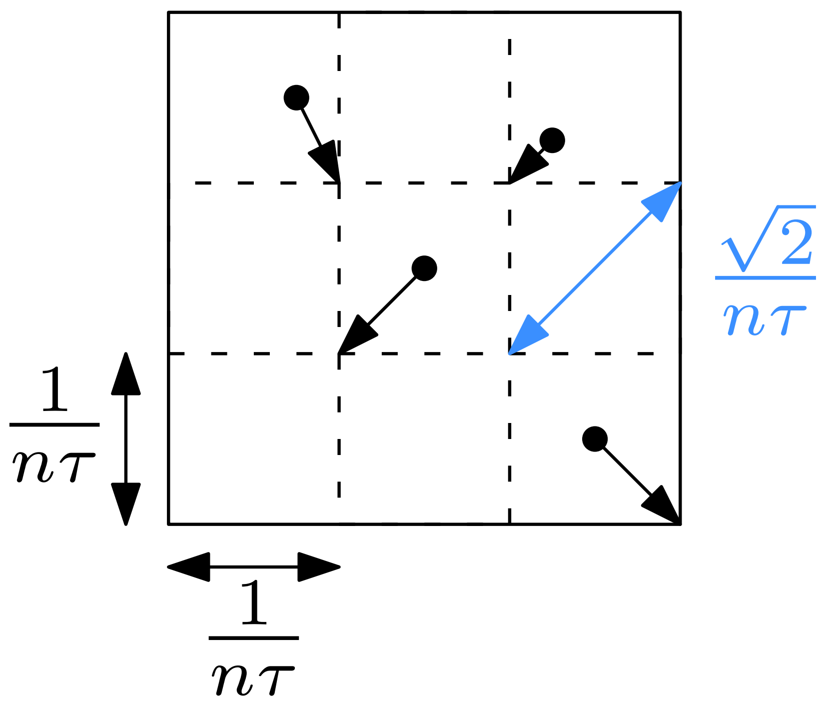
Let Q be P with snapped to nearest gridpoints

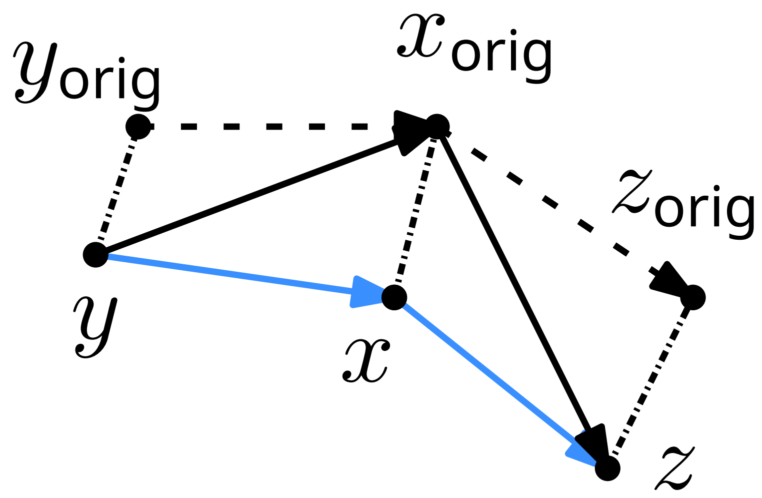
Each point was moved at most $\frac{\sqrt{2}}{2n\tau}$

Solution for Q can be converted to solution for P

Additional cost: $\leq 2n \cdot \frac{\sqrt{2}}{2n\tau} \leq \frac{\sqrt{2}}{\tau} \leq \frac{4\varepsilon}{32} \leq \frac{\varepsilon}{2} \cdot \frac{1}{4} \leq \frac{\varepsilon}{2}||\pi_{\text{opt}}||$

Diameter of P is at least $\frac{1}{4}$, so an optimal solution must be at least as large





A Good Quadtree - Height

Recall: Q contained in $[\frac{1}{2}, 1]^2$ and on grid of width $\frac{1}{n\tau}$

spread: $\Phi(Q) = \frac{\max_{p,q \in Q} ||p-q||}{\min_{p,q \in Q} ||p-q||} = \frac{\sqrt{2}/2}{1/(n\tau)} = \frac{n\tau\sqrt{2}}{2}$

A Good Quadtree - Height

Recall: Q contained in $[\frac{1}{2}, 1]^2$ and on grid of width $\frac{1}{n\tau}$

spread: $\Phi(Q) = \frac{\max_{p,q\in Q} ||p-q||}{\min_{p,q\in Q} ||p-q||} = \frac{\sqrt{2}/2}{1/(n\tau)} = \frac{n\tau\sqrt{2}}{2}$

Let Q a set of n points in the unit square such that its diameter is at least $\frac{1}{2}$. Let $\mathcal{T}$ the quadtree of Q constructed over the unit square. The depth of $\mathcal{T}$ is bounded by $\mathcal{O}(\log \Phi(Q))$, can be constructed in $\mathcal{O}(n \log \Phi(Q))$ time, and is of total size $\mathcal{O}(n \log \Phi(Q))$.

A Good Quadtree - Height

Recall: Q contained in $[\frac{1}{2}, 1]^2$ and on grid of width $\frac{1}{n\tau}$

spread: $\Phi(Q) = \frac{\max_{p,q \in Q} ||p-q||}{\min_{p,q \in Q} ||p-q||} = \frac{\sqrt{2}/2}{1/(n\tau)} = \frac{n\tau\sqrt{2}}{2}$

Let Q a set of n points in the unit square such that its diameter is at least $\frac{1}{2}$. Let $\mathcal{T}$ the quadtree of Q constructed over the unit square. The depth of $\mathcal{T}$ is bounded by $\mathcal{O}(\log \Phi(Q))$, can be constructed in $\mathcal{O}(n \log \Phi(Q))$ time, and is of total size $\mathcal{O}(n \log \Phi(Q))$.

Diameter of Q is at least $\frac{1}{4}$, so we may need one extra level

A Good Quadtree - Height

Recall: Q contained in $[\frac{1}{2}, 1]^2$ and on grid of width $\frac{1}{n\tau}$

spread: $\Phi(Q) = \frac{\max_{p,q \in Q} ||p-q||}{\min_{p,q \in Q} ||p-q||} = \frac{\sqrt{2}/2}{1/(n\tau)} = \frac{n\tau\sqrt{2}}{2}$

Let Q a set of n points in the unit square such that its diameter is at least $\frac{1}{2}$. Let $\mathcal{T}$ the quadtree of Q constructed over the unit square. The depth of $\mathcal{T}$ is bounded by $\mathcal{O}(\log \Phi(Q))$, can be constructed in $\mathcal{O}(n \log \Phi(Q))$ time, and is of total size $\mathcal{O}(n \log \Phi(Q))$.

Diameter of Q is at least $\frac{1}{4}$, so we may need one extra level

Height $H = \mathcal{O}\left(\log \frac{n\tau\sqrt{2}}{2}\right) = \mathcal{O}\left(\log \frac{n}{\varepsilon}\right) = \mathcal{O}(\log n)$

Similarly running time and storage of $\mathcal{O}(n \log n)$ follow

Algorithm

`Initialization`(Q):

Construct quadtree $\mathcal{T}$ over Q with height H

Let $k = \frac{90}{\varepsilon} = \mathcal{O}(\frac{1}{\varepsilon})$, $\qquad m \geq \frac{20H}{\varepsilon} = \mathcal{O}(\varepsilon^{-1} \log n)$

`Recursive`(S, M):

1. **if** $|Q_S| = \mathcal{O}(\frac{1}{\varepsilon})$ **then return** `BruteForce`(S, M)
2. $\min_{\text{length}} \leftarrow \infty$
3. **for each** combination $C = [C_1, C_2, C_3, C_4]$ of subproblems of children at S:
4.   **if** C is valid **then**:
5.   cost $\leftarrow$ ParentConnect$(C, S, M) + \sum_{i=1}^{4}$ `Recursive`(C_i)
6. $\min_{\text{length}} \leftarrow \min(\min_{\text{length}}, \text{cost})$
7. **return** $\min_{\text{length}}$

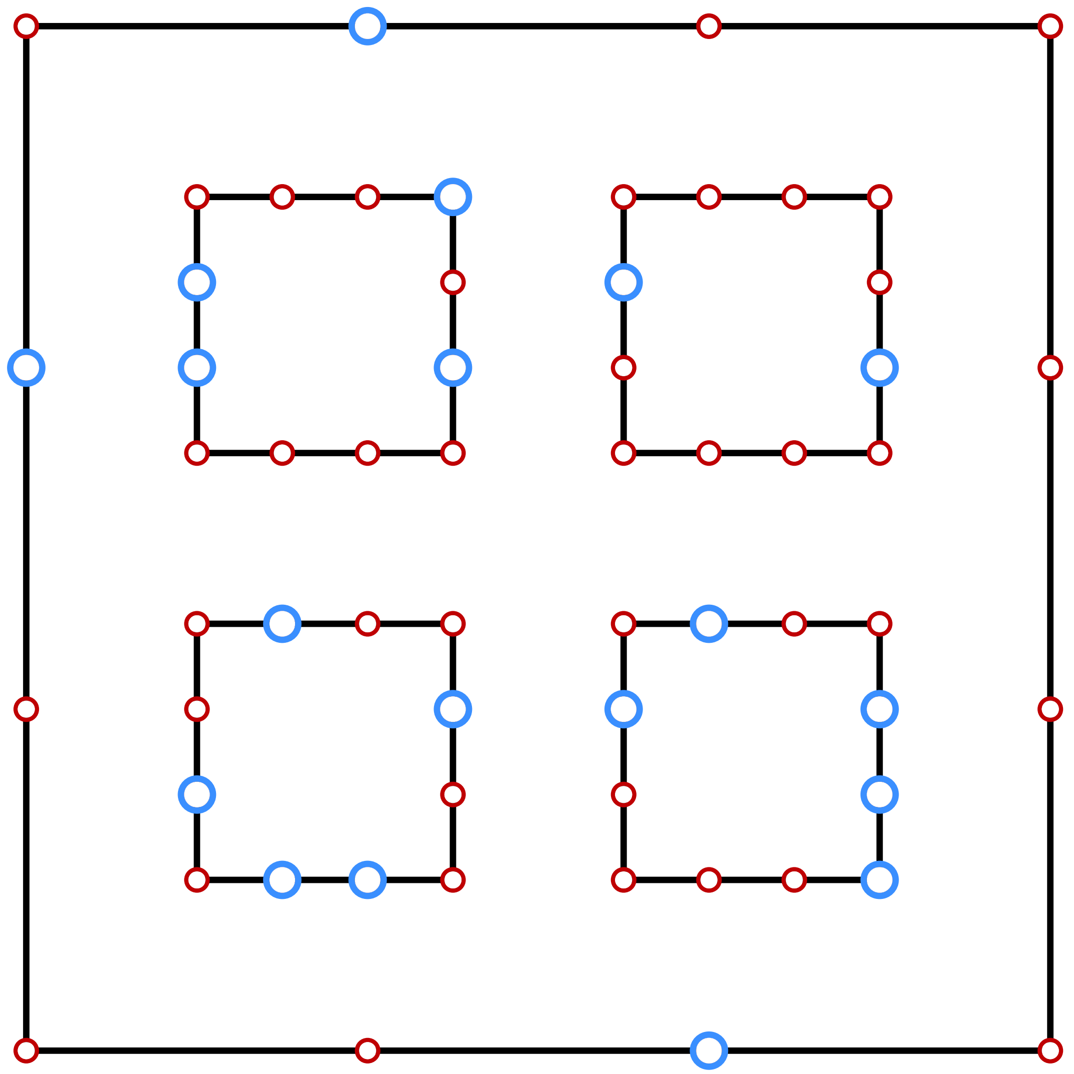
Algorithm

`Recursive`(S, M):

3. **for each** combination $C = [C_1, C_2, C_3, C_4]$ of subproblems of children at S:

C_i is an arbitrary subproblem for child i

C contains one subproblem for each child



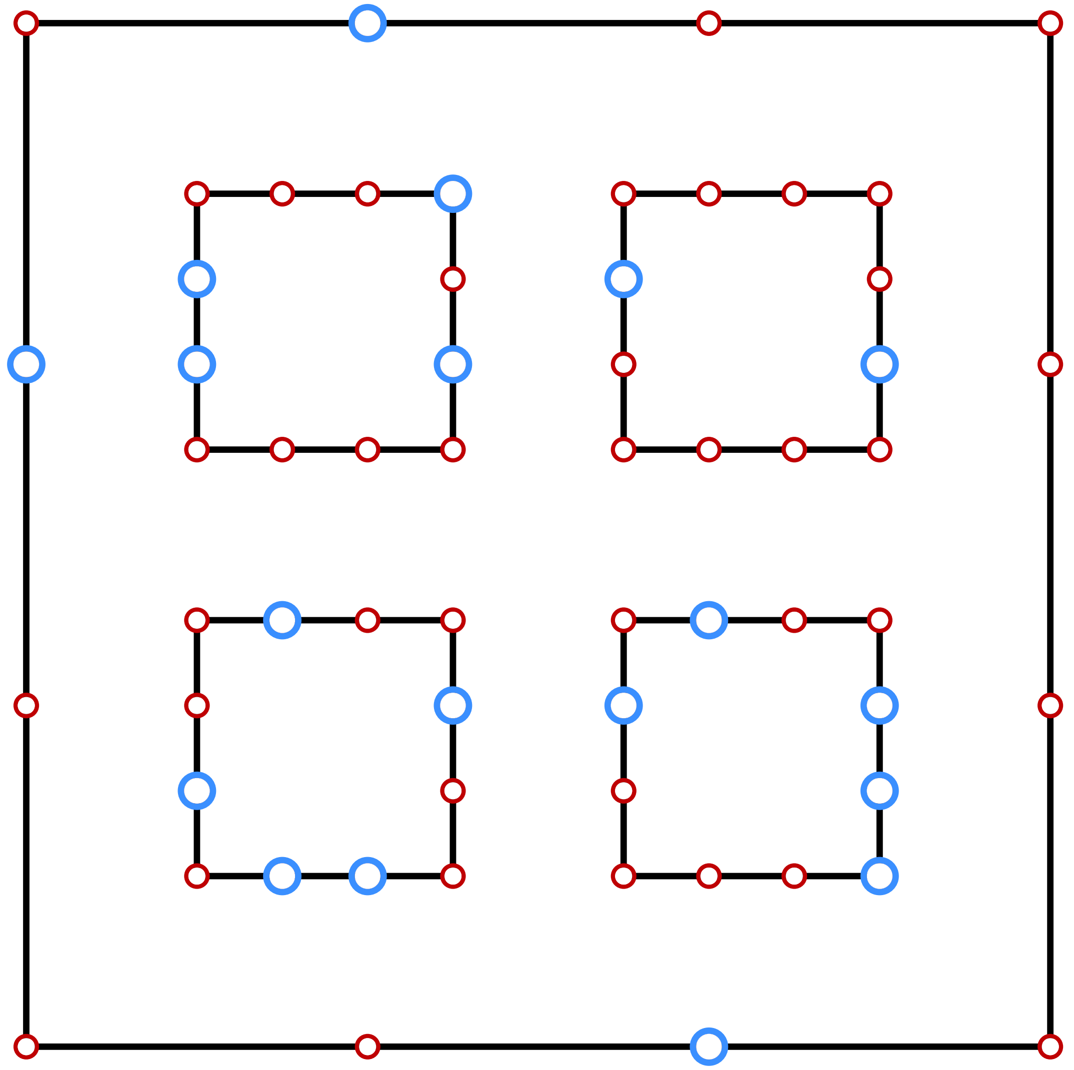
Algorithm

`Recursive`(S, M)`:`

3. **for each** combination $C = [C_1, C_2, C_3, C_4]$ of subproblems of children at S:
4.   **if** C is valid **then**:

C_i is an arbitrary subproblem for child i

C contains one subproblem for each child



Algorithm

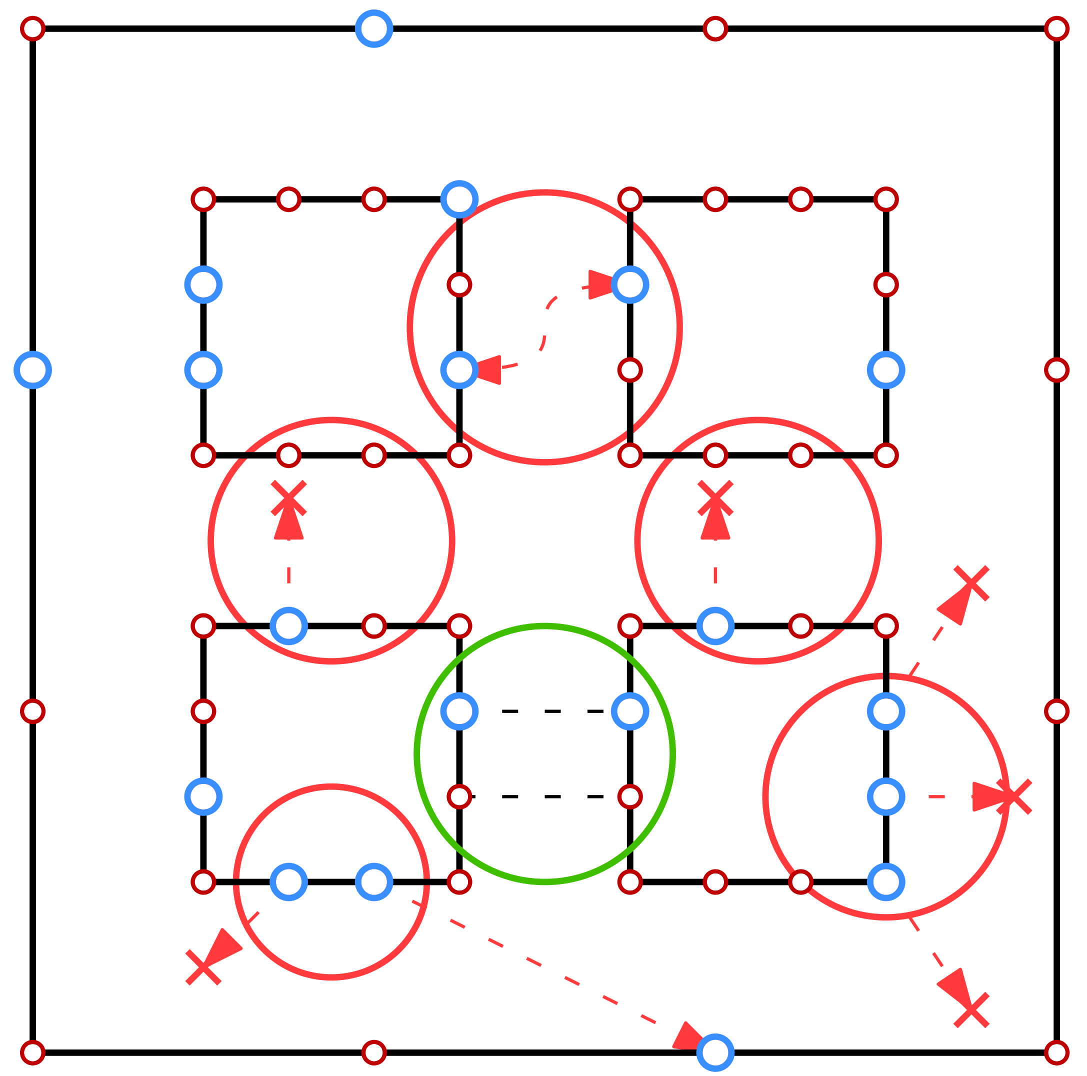
`Recursive`(S, M)`:`

3. **for each** combination $C = [C_1, C_2, C_3, C_4]$ of subproblems of children at S:

4. **if** C is valid **then**:

C_i is an arbitrary subproblem for child i

C contains one subproblem for each child



Algorithm

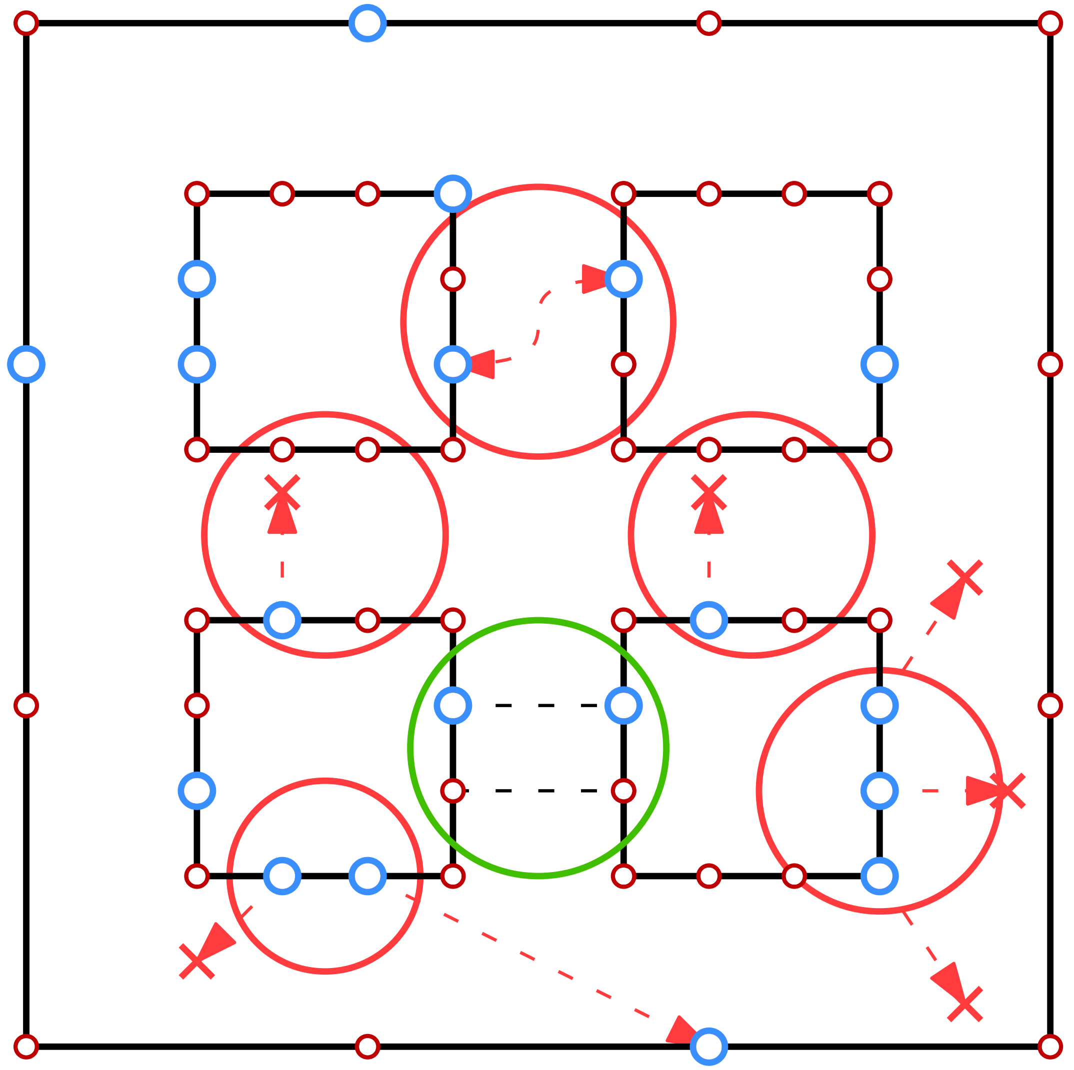
`Recursive`(S, M)`:`

3. **for each** combination $C = [C_1, C_2, C_3, C_4]$ of subproblems of children at S:
4. **if** C is valid **then**:

C_i is an arbitrary subproblem for child i

C contains one subproblem for each child

Valid: even number of portals per child. Portals between children match and number of portals on outside match with parent portals.



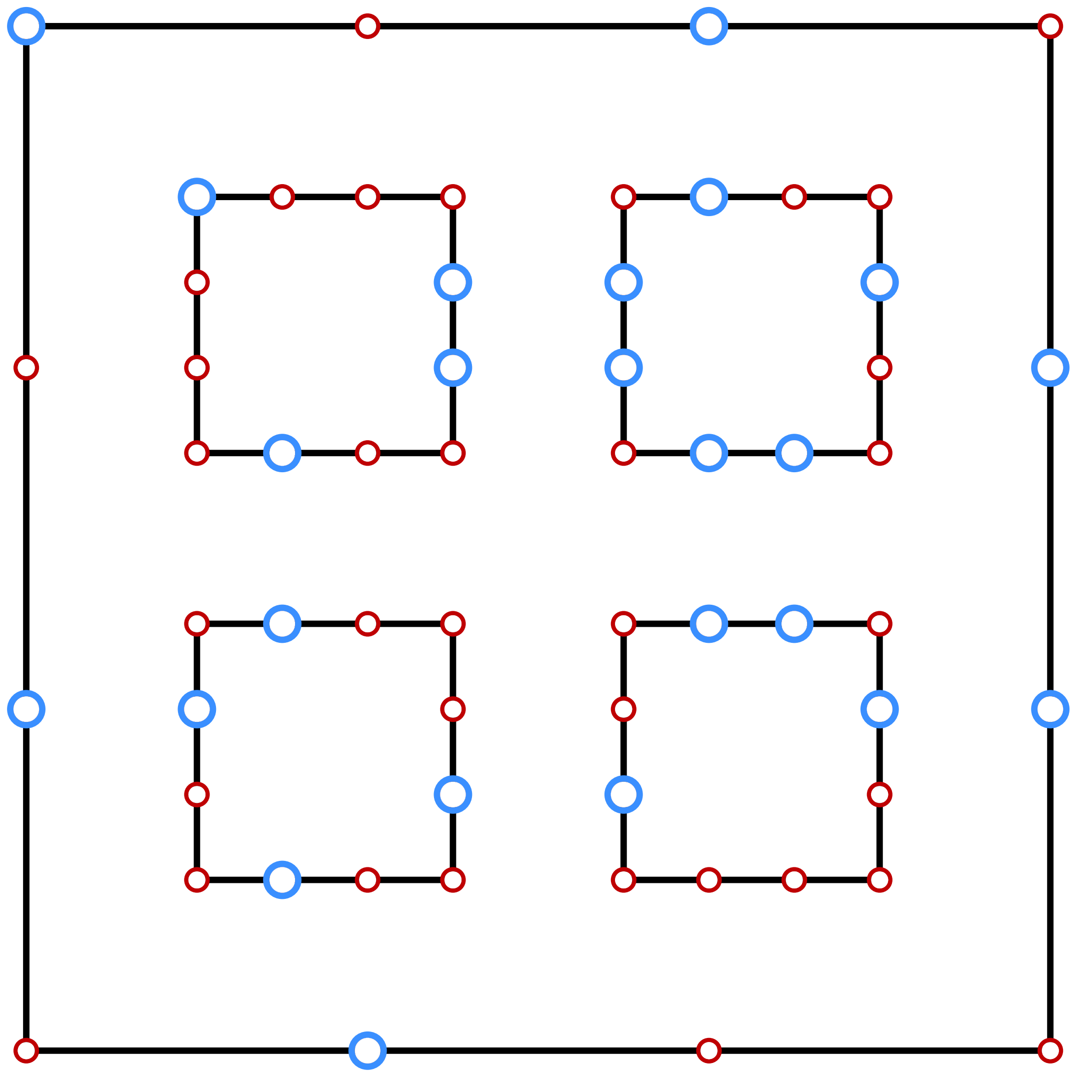
Algorithm

`Recursive`(S, M):
3. **for each** combination $C = [C_1, C_2, C_3, C_4]$ of subproblems of children at S:
4. **if** C is valid **then**:

C_i is an arbitrary subproblem for child i

C contains one subproblem for each child

Valid: even number of portals per child. Portals between children match and number of portals on outside match with parent portals.



Algorithm

Recursive(S, M):

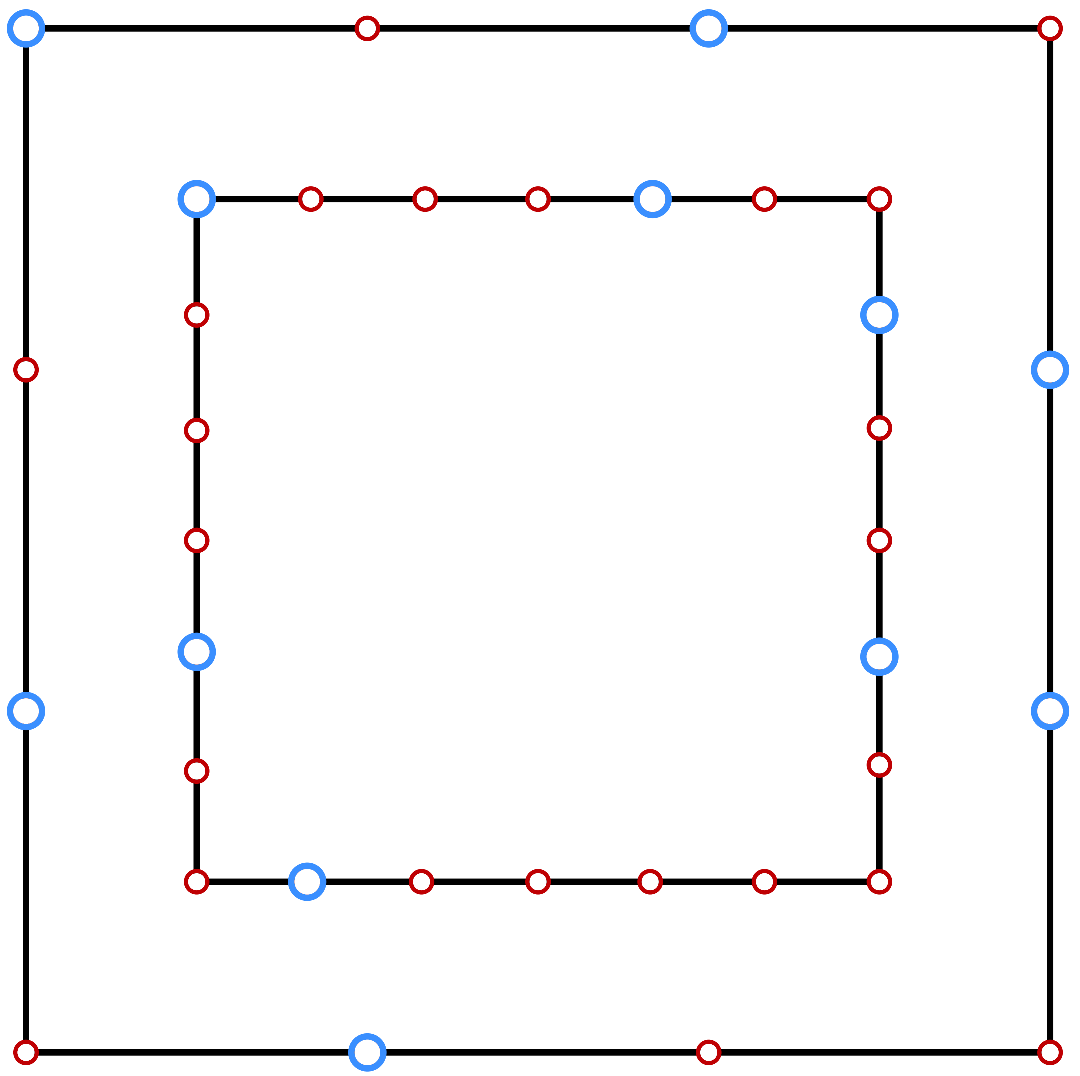
3. **for each** combination $C = [C_1, C_2, C_3, C_4]$ of subproblems of children at S:

4. **if** C is valid **then**:

C_i is an arbitrary subproblem for child i

C contains one subproblem for each child

Valid: even number of portals per child. Portals between children match and number of portals on outside match with parent portals.



Algorithm

`Recursive`(S, M)`:`

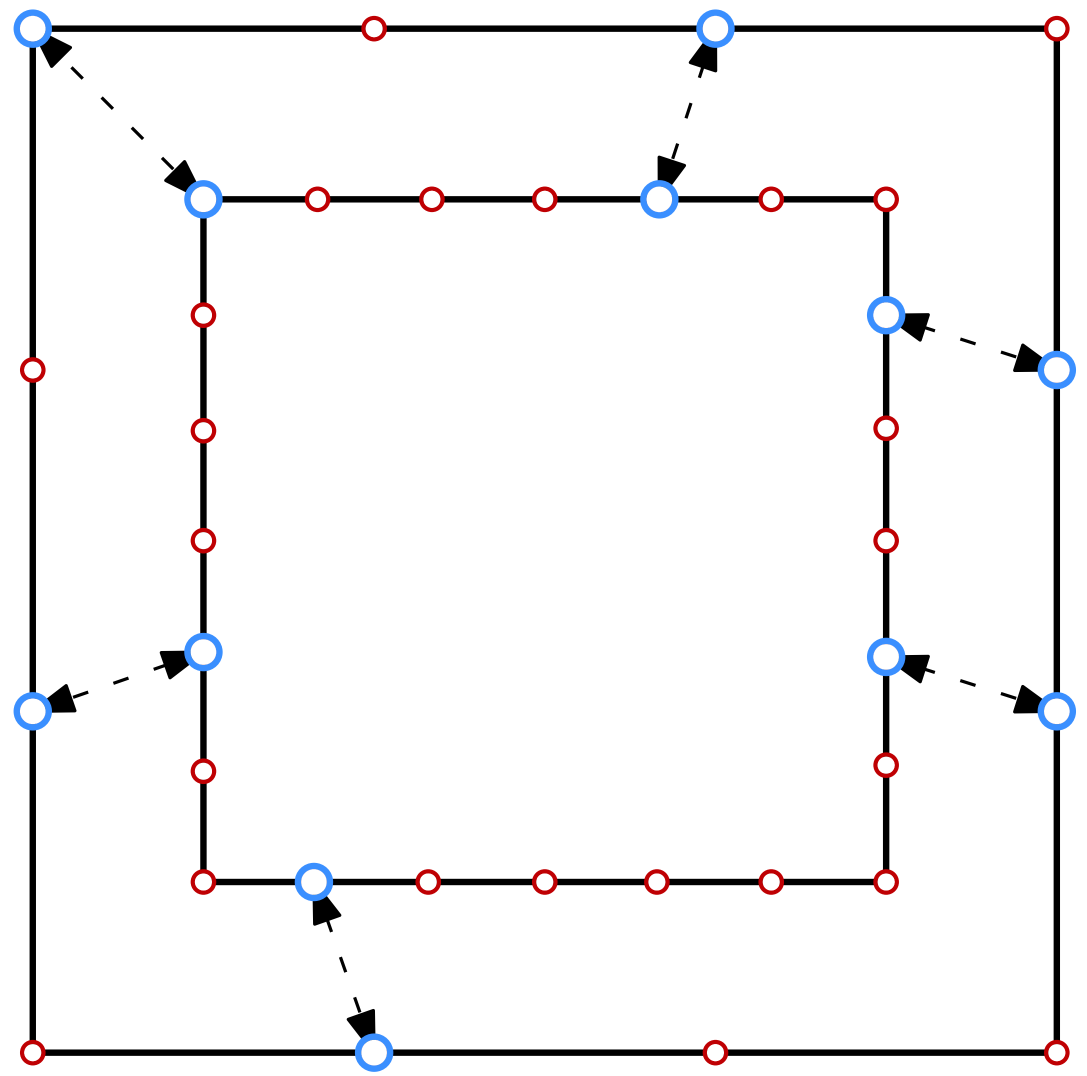
3. **for each** combination $C = [C_1, C_2, C_3, C_4]$ of subproblems of children at S:

4.   **if** C is valid **then**:

C_i is an arbitrary subproblem for child i

C contains one subproblem for each child

Valid: even number of portals per child. Portals between children match and number of portals on outside match with parent portals.



Algorithm

`Recursive`(S, M):

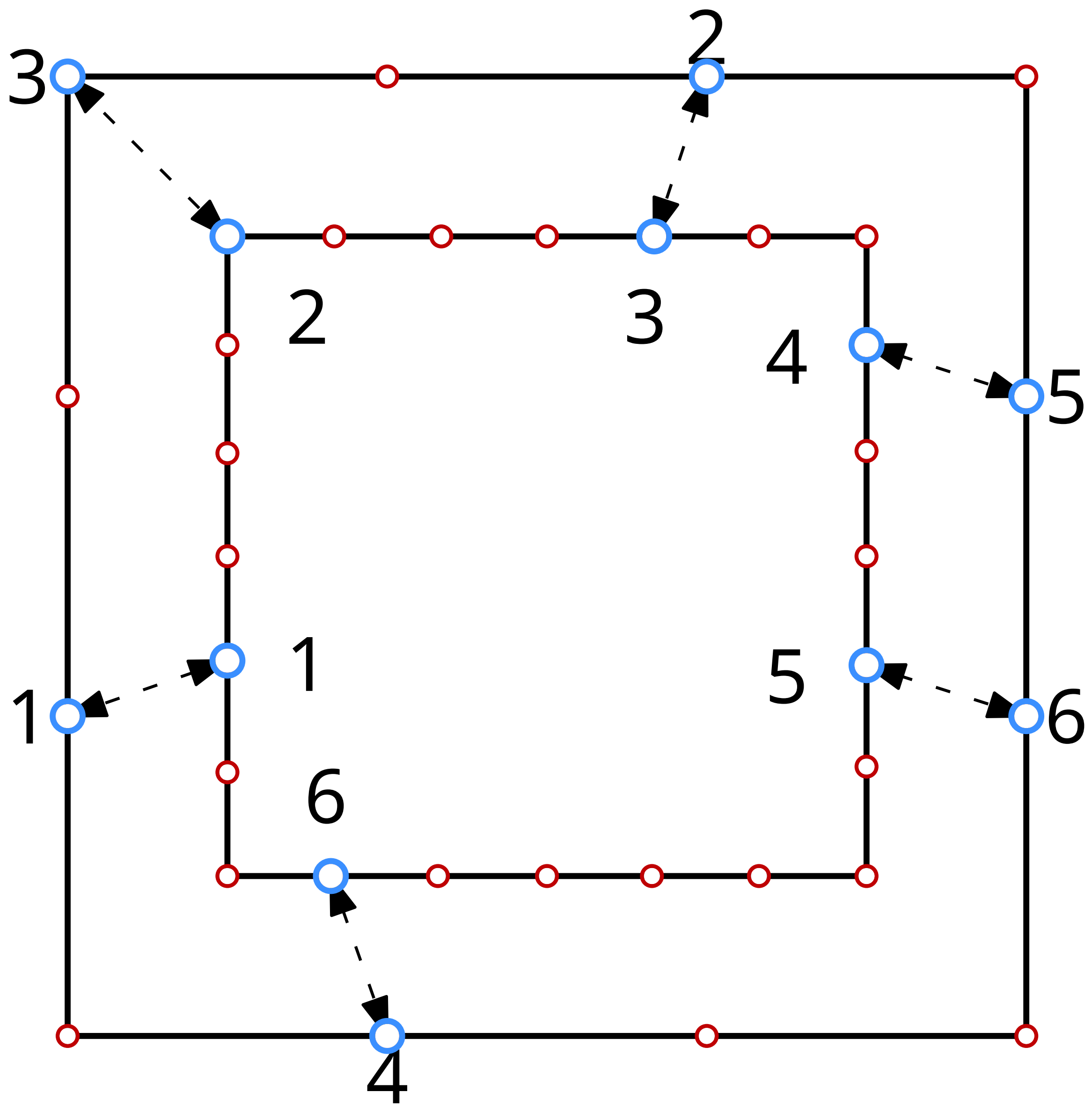
3. **for each** combination $C = [C_1, C_2, C_3, C_4]$ of subproblems of children at S:

4. **if** C is valid **then**:

C_i is an arbitrary subproblem for child i

C contains one subproblem for each child

Valid: even number of portals per child. Portals between children match and number of portals on outside match with parent portals.



Algorithm

`Recursive`(S, M):

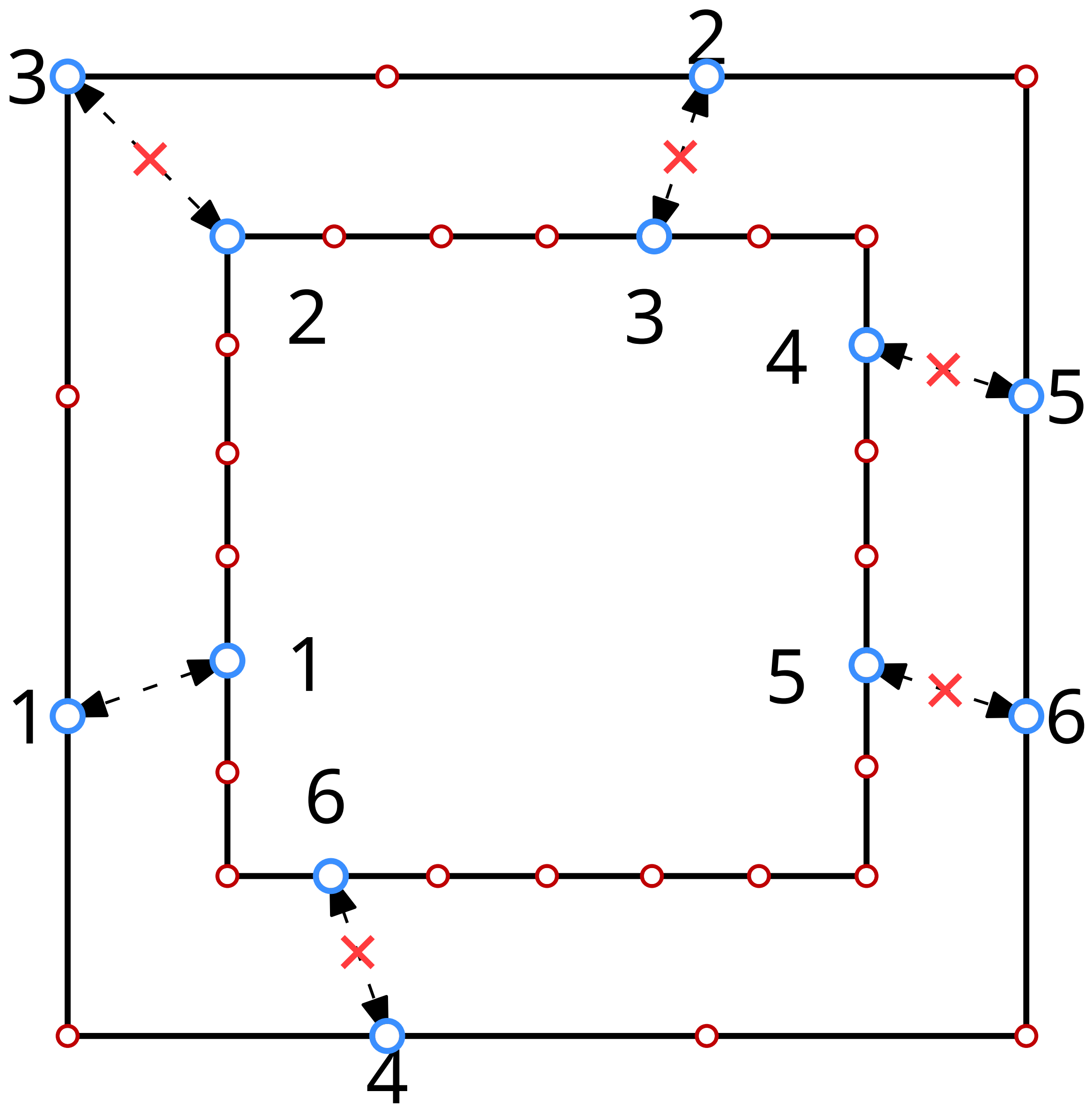
3. **for each** combination $C = [C_1, C_2, C_3, C_4]$ of subproblems of children at S:

4. **if** C is valid **then**:

C_i is an arbitrary subproblem for child i

C contains one subproblem for each child

Valid: even number of portals per child. Portals between children match and number of portals on outside match with parent portals.



Algorithm

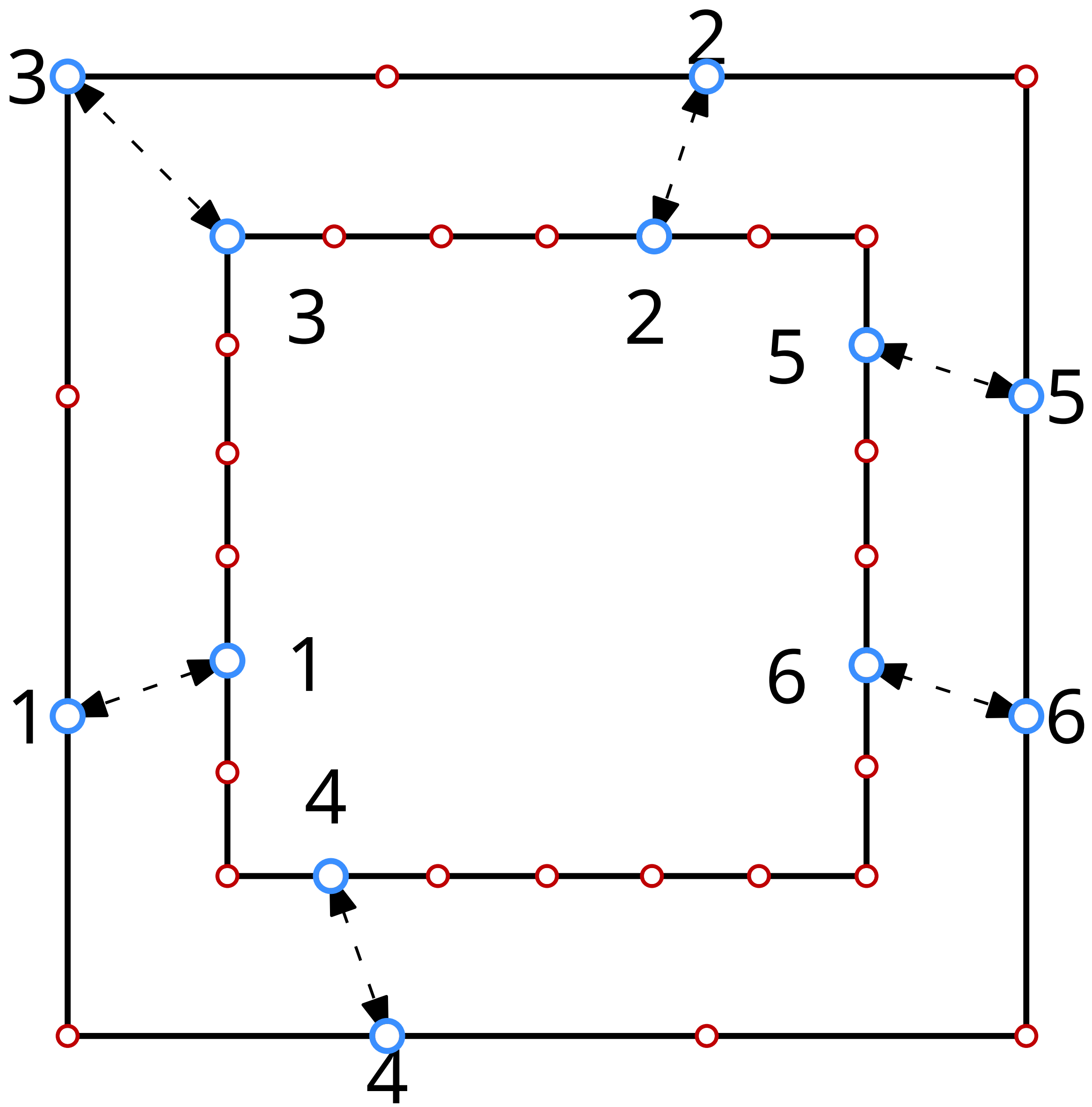
$\texttt{Recursive}(S, M)\texttt{:}$

3. **for each** combination $C = [C_1, C_2, C_3, C_4]$ of subproblems of children at S:
4. **if** C is valid **then**:

C_i is an arbitrary subproblem for child i

C contains one subproblem for each child

Valid: even number of portals per child. Portals between children match and number of portals on outside match with parent portals.
Outside portals have same order as parent



Algorithm

`Recursive`(S, M):

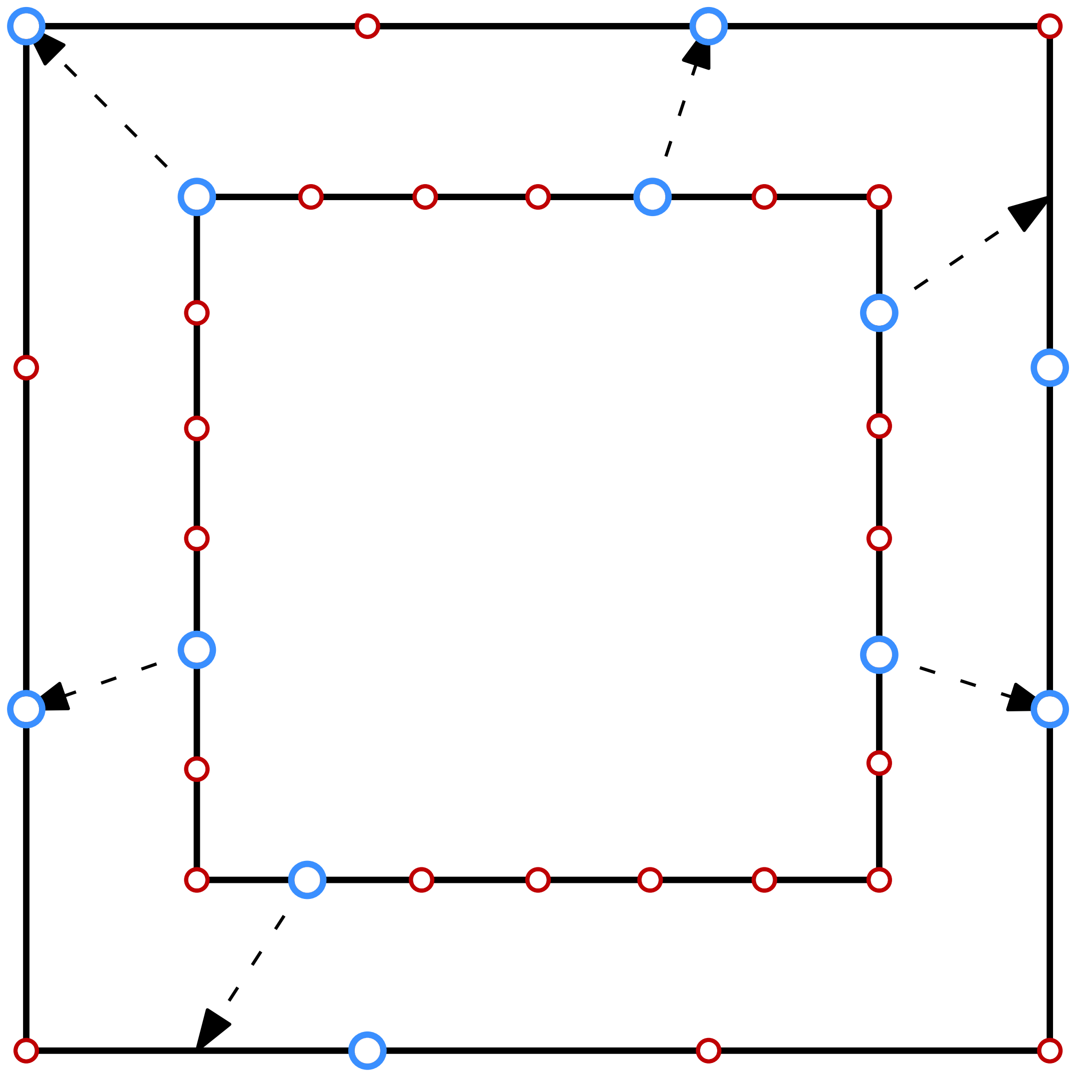
3. **for each** combination $C = [C_1, C_2, C_3, C_4]$ of subproblems of children at S:
4. **if** C is valid **then**:
5. cost $\leftarrow$ ParentConnect$(C, S, M) + \sum_{i=1}^{4}$ `Recursive`(C_i)

C_i is an arbitrary subproblem for child i

C contains one subproblem for each child

Valid: even number of portals per child. Portals between children match and number of portals on outside match with parent portals.

Outside portals have same order as parent



Algorithm

`Recursive`(S, M):

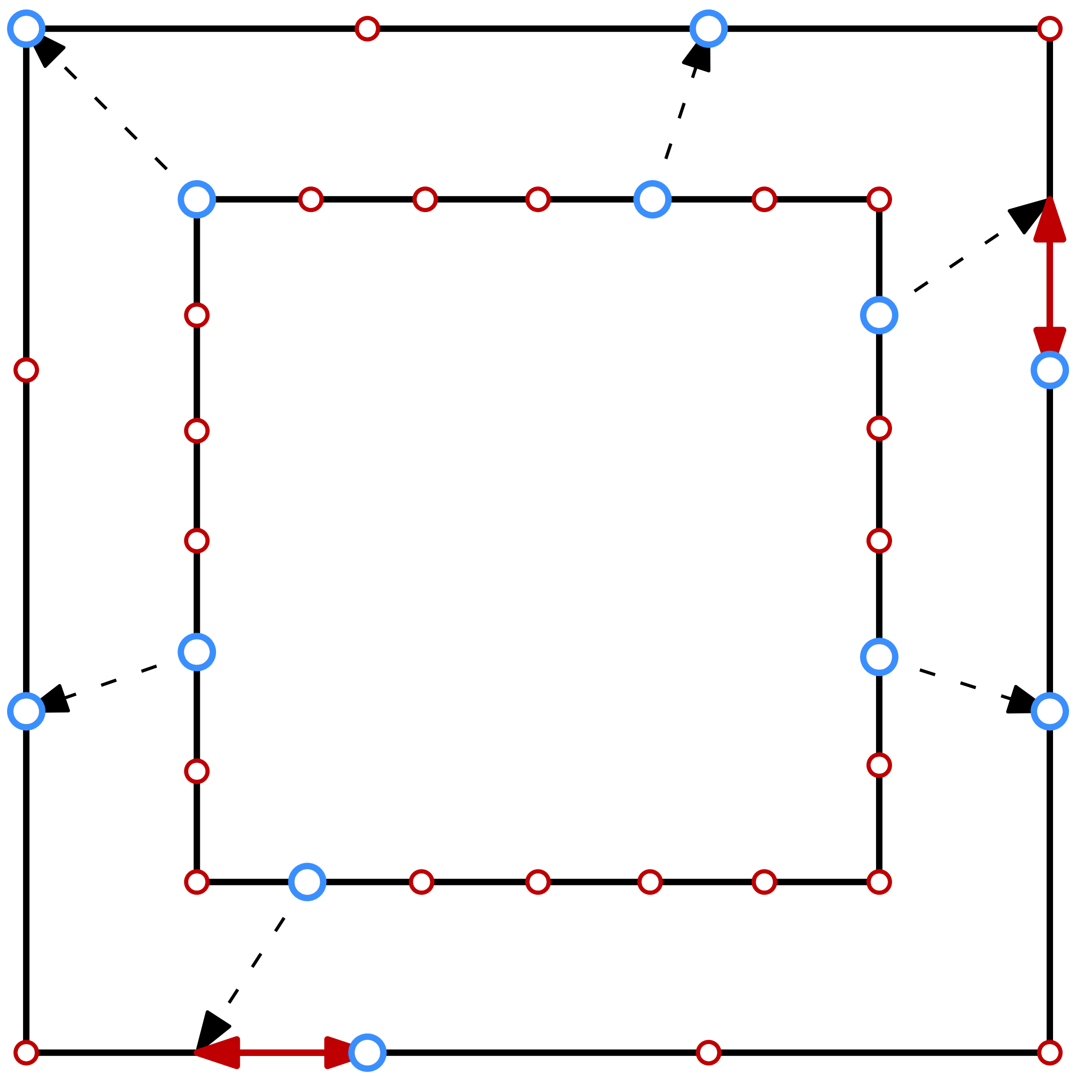
3. **for each** combination $C = [C_1, C_2, C_3, C_4]$ of subproblems of children at S:
4.   **if** C is valid **then**:
5.   cost $\leftarrow$ ParentConnect$(C, S, M) + \sum_{i=1}^{4}$ `Recursive`(C_i)

C_i is an arbitrary subproblem for child i

C contains one subproblem for each child

Valid: even number of portals per child. Portals between children match and number of portals on outside match with parent portals.
Outside portals have same order as parent

ParentConnect(C, S, M): total misalignment between parent and child portals



Algorithm

`Initialization`(Q):

Construct quadtree $\mathcal{T}$ over Q with height H

Let $k = \frac{90}{\varepsilon} = \mathcal{O}(\frac{1}{\varepsilon})$, $\qquad m \geq \frac{20H}{\varepsilon} = \mathcal{O}(\varepsilon^{-1} \log n)$

`Recursive`(S, M):

1. **if** $|Q_S| = \mathcal{O}(\frac{1}{\varepsilon})$ **then return** `BruteForce`(S, M)
2. $\min_{\text{length}} \leftarrow \infty$
3. **for each** combination $C = [C_1, C_2, C_3, C_4]$ of subproblems of children at S:
4.   **if** C is valid **then**:
5.   cost $\leftarrow$ ParentConnect$(C, S, M) + \sum_{i=1}^{4}$ `Recursive`(C_i)
6. $\min_{\text{length}} \leftarrow \min(\min_{\text{length}}, \text{cost})$
7. **return** $\min_{\text{length}}$

Use memoization to make it a DP algorithm

Overview

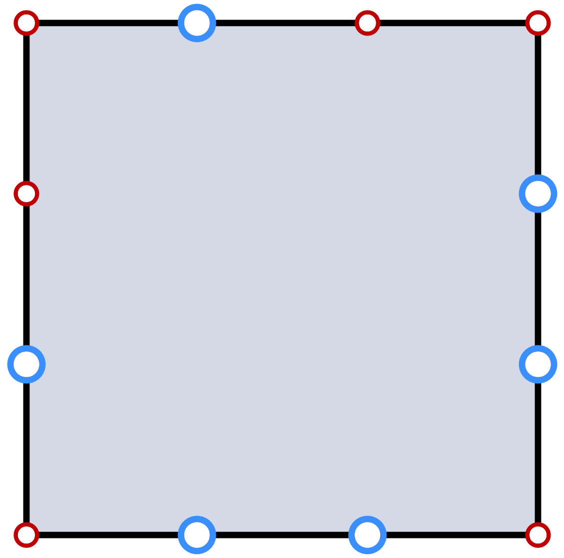
1. Intuition ✓
2. Subproblems ✓
3. Algorithm ✓
4. Running time
5. Quality of approximation

Running Time - Number of Subproblems per Square

Consider a square S and its portals

It has $4m + 4$ portals on its boundary, each of which is used at most twice

At most k portals are used on each side of S



Running Time - Number of Subproblems per Square

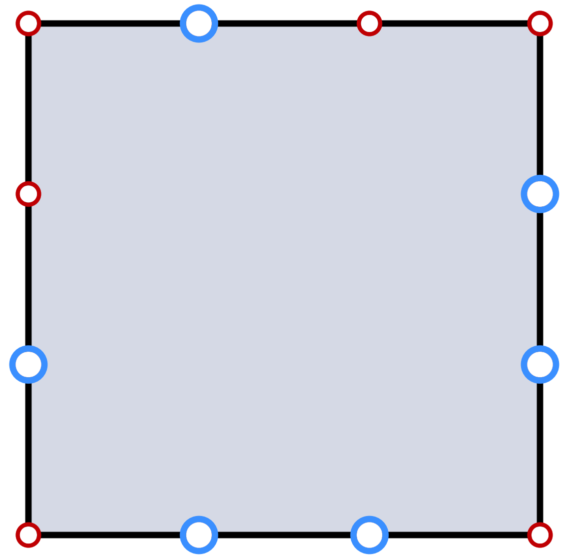
Consider a square S and its portals

It has $4m + 4$ portals on its boundary, each of which is used at most twice

At most k portals are used on each side of S

Let $i \leq 8k$ the total number of portals used

There are $i!$ orderings of these portals



Running Time - Number of Subproblems per Square

Consider a square S and its portals

It has $4m+4$ portals on its boundary, each of which is used at most twice

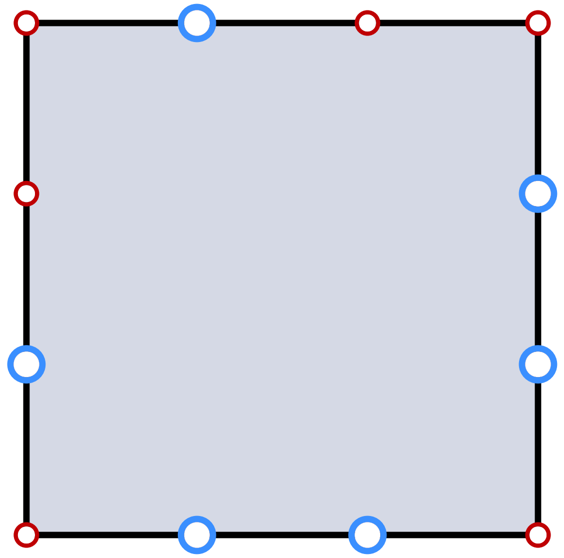
At most k portals are used on each side of S

Let $i \leq 8k$ the total number of portals used

There are $i!$ orderings of these portals

Bound T the number of subproblems with S as square

$$T = \sum_{i=0}^{8k} \binom{8m+8}{i} i!$$



Running Time - Number of Subproblems per Square

Consider a square S and its portals

It has $4m+4$ portals on its boundary, each of which is used at most twice

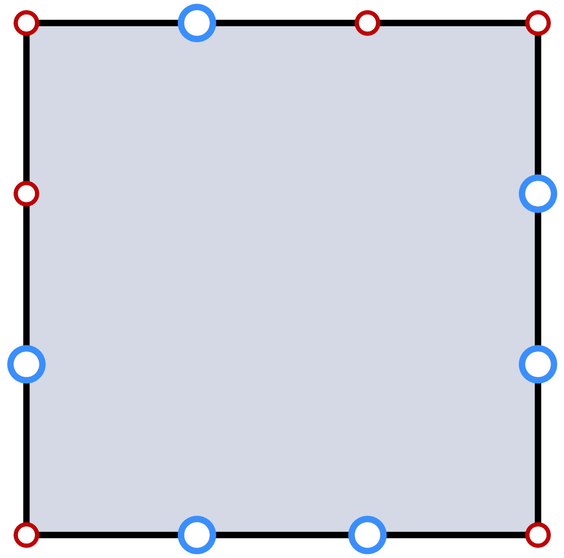
At most k portals are used on each side of S

Let $i \leq 8k$ the total number of portals used

There are $i!$ orderings of these portals

Bound T the number of subproblems with S as square

$$T = \sum_{i=0}^{8k} \binom{8m+8}{i} i! \leq 8k \binom{8m+8}{8k} (8k)!$$



Running Time - Number of Subproblems per Square

Consider a square S and its portals

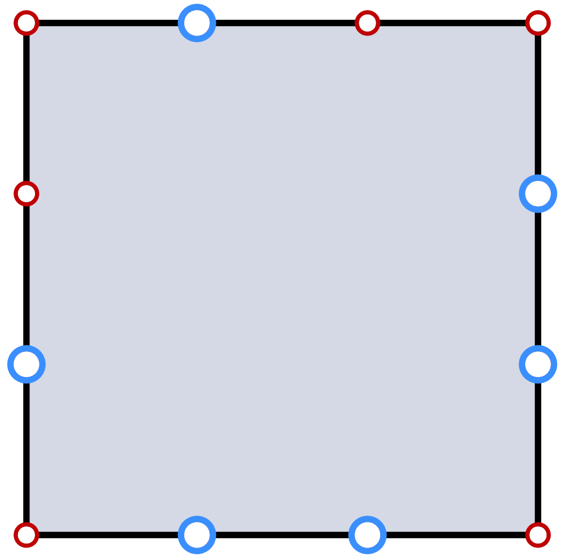
It has $4m + 4$ portals on its boundary, each of which is used at most twice

At most k portals are used on each side of S

Let $i \leq 8k$ the total number of portals used

There are $i!$ orderings of these portals

Bound T the number of subproblems with S as square



$$T = \sum_{i=0}^{8k} \binom{8m+8}{i} i! \leq 8k \binom{8m+8}{8k} (8k)!$$

$$= 8k \frac{(8m+8)!}{(8k)!(8m+8-8k)!} (8k)! = 8k \frac{(8m+8)!}{(8m+8-8k)!}$$

$$\leq 8k(8m+8)^{8k}$$

Running Time - Number of Subproblems per Square

Consider a square S and its portals

It has $4m + 4$ portals on its boundary, each of which is used at most twice

At most k portals are used on each side of S

Let $i \leq 8k$ the total number of portals used

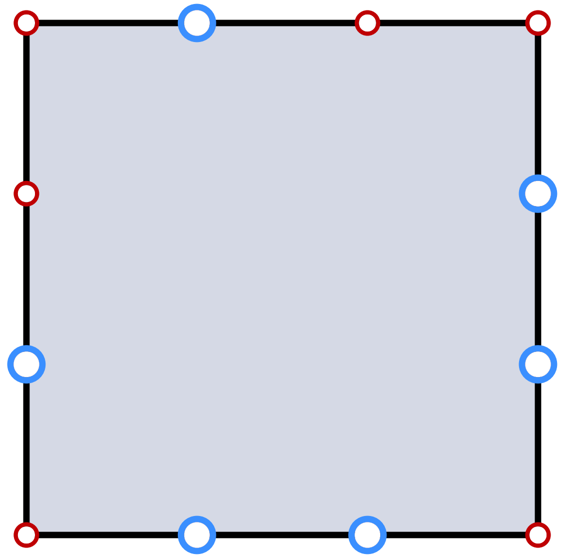
There are $i!$ orderings of these portals

Bound T the number of subproblems with S as square

$$T = \sum_{i=0}^{8k} \binom{8m+8}{i} i! \leq 8k \binom{8m+8}{8k} (8k)!$$

$$= 8k \frac{(8m+8)!}{(8k)!(8m+8-8k)!} (8k)! = 8k \frac{(8m+8)!}{(8m+8-8k)!}$$

$$\leq 8k(8m+8)^{8k} = (\varepsilon^{-1} \log n)^{\mathcal{O}(1/\varepsilon)}$$



Recall

$$k = \mathcal{O}\left(\frac{1}{\varepsilon}\right)$$

$$m = \mathcal{O}\left(\frac{\log n}{\varepsilon}\right)$$

Running Time

Recall: quadtree had $\mathcal{O}(n \log n)$ nodes and one square per node
Claim: brute force on a square with $\mathcal{O}\left(\frac{1}{\varepsilon}\right)$ points (base case) can be done in $T = (\varepsilon^{-1} \log n)^{\mathcal{O}(1/\varepsilon)}$ time

Running Time

Recall: quadtree had $\mathcal{O}(n \log n)$ nodes and one square per node
Claim: brute force on a square with $\mathcal{O}\left(\frac{1}{\varepsilon}\right)$ points (base case) can be done in $T = (\varepsilon^{-1} \log n)^{\mathcal{O}(1/\varepsilon)}$ time

Algorithm considers all combinations of subproblems for the squares of the children for each node

The number of subproblems per child is at most T
The number of combinations for four children is T^4
Computing validity and ParentConnect: upper bounded by $\mathcal{O}(T)$ time

Running Time

Recall: quadtree had $\mathcal{O}(n \log n)$ nodes and one square per node
Claim: brute force on a square with $\mathcal{O}\left(\frac{1}{\varepsilon}\right)$ points (base case) can be done in $T = (\varepsilon^{-1} \log n)^{\mathcal{O}(1/\varepsilon)}$ time

Algorithm considers all combinations of subproblems for the squares of the children for each node

The number of subproblems per child is at most T
The number of combinations for four children is T^4
Computing validity and ParentConnect: upper bounded by $\mathcal{O}(T)$ time

Total running time:
$\mathcal{O}(n \log n + (n \log n)T^5) = \mathcal{O}((n \log n)T^5) = n(\varepsilon^{-1} \log n)^{\mathcal{O}(1/\varepsilon)}$

Overview

1. Intuition ✓
2. Subproblems ✓
3. Algorithm ✓
4. Running time ✓
5. Quality of approximation

Quality of Approximation

Introduced error when:

- snapping to the grid
- bounding the number of intersections at k per side of each square
- requiring the use of portals

Quality of Approximation

Introduced error when:

- snapping to the grid
- bounding the number of intersections at k per side of each square
  uses: patching lemma (+ shifting)
- requiring the use of portals

Quality of Approximation

Introduced error when:

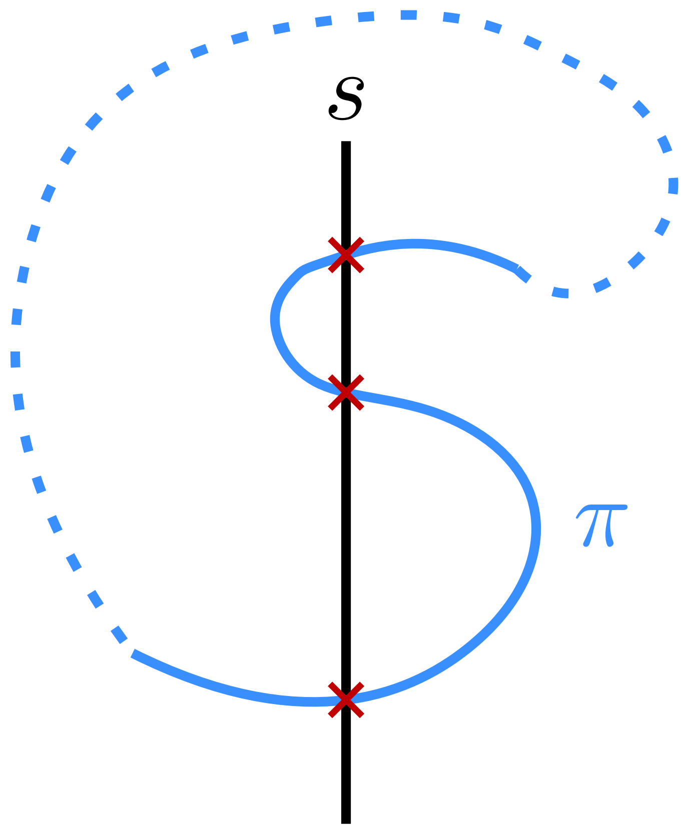
- snapping to the grid
- bounding the number of intersections at k per side of each square

  uses: patching lemma (+ shifting)
- requiring the use of portals

  uses: shifting

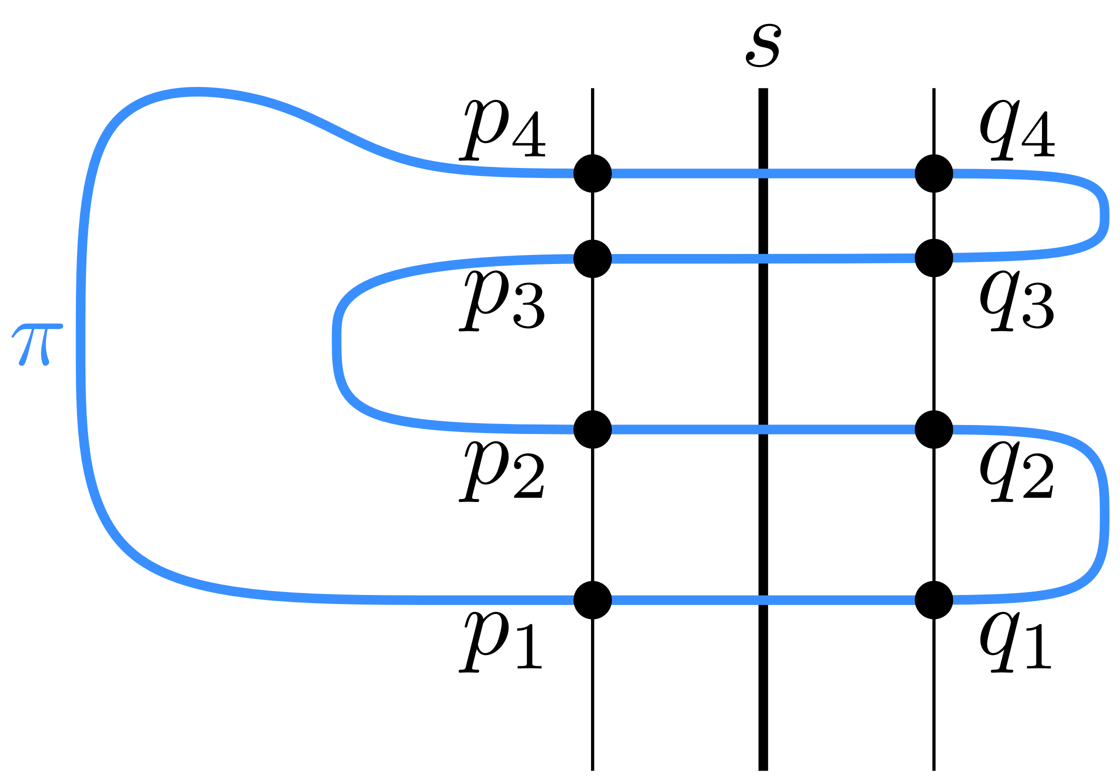
Patching Lemma

Patching lemma: closed curve π crossing segment s at least $k \geq 3$ times can be replaced by closed curve π' crossing s at most twice such that $||\pi'|| \leq ||\pi|| + 4||s||$



Patching Lemma

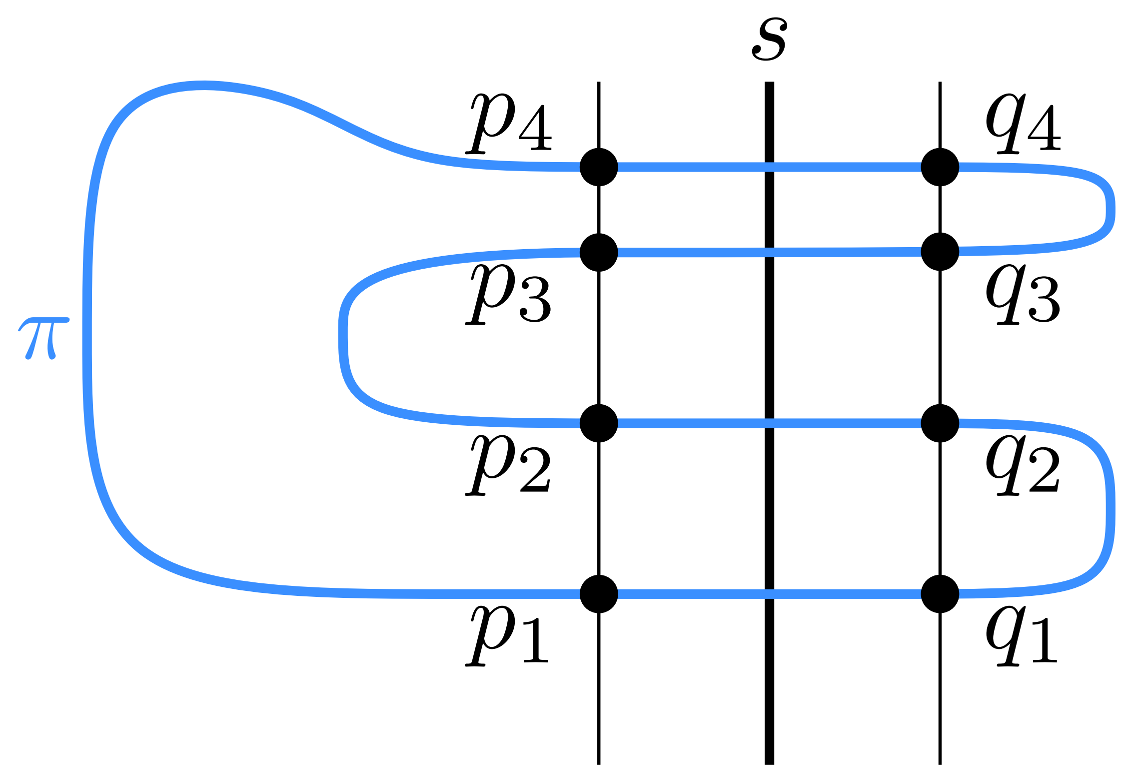
Patching lemma: closed curve π crossing segment s at least $k \geq 3$ times can be replaced by closed curve π' crossing s at most twice such that $||\pi'|| \leq ||\pi|| + 4||s||$



Patching Lemma

Patching lemma: closed curve π crossing segment s at least $k \geq 3$ times can be replaced by closed curve π' crossing s at most twice such that $||\pi'|| \leq ||\pi|| + 4||s||$

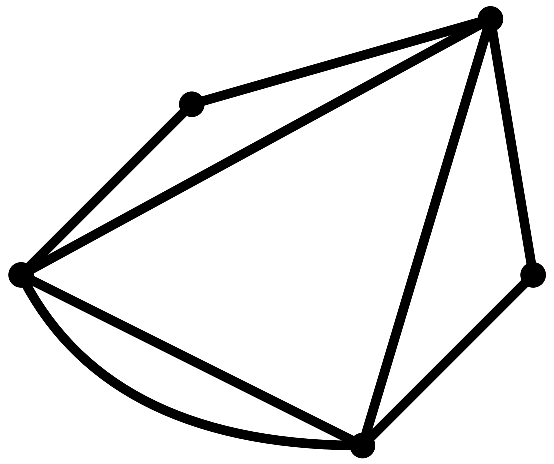
Idea: construct an Eulerian tour including π that crosses s at most twice

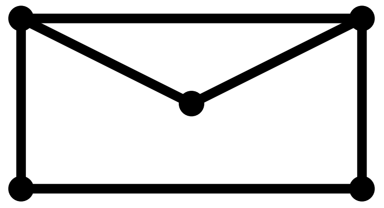


Patching Lemma

Patching lemma: closed curve π crossing segment s at least $k \geq 3$ times can be replaced by closed curve π' crossing s at most twice such that $||\pi'|| \leq ||\pi|| + 4||s||$

Theorem: a connected graph admits an Eulerian tour if and only if its vertices have even degree

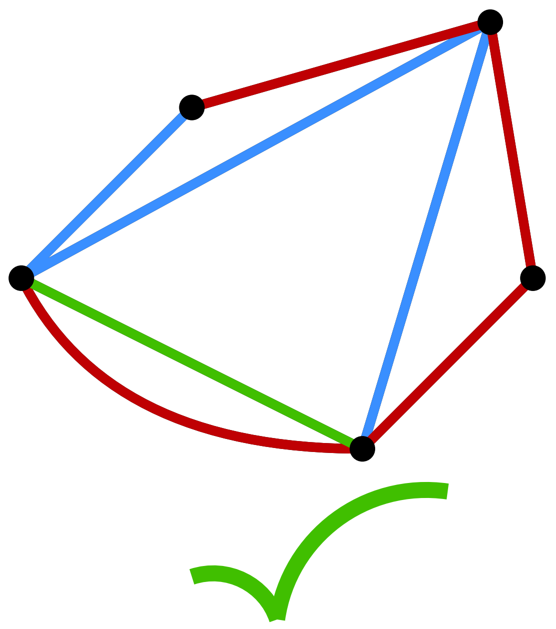


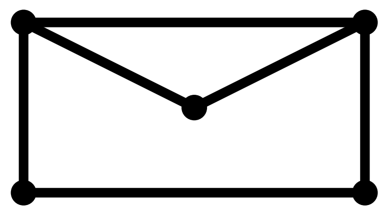


Patching Lemma

Patching lemma: closed curve π crossing segment s at least $k \geq 3$ times can be replaced by closed curve π' crossing s at most twice such that $||\pi'|| \leq ||\pi|| + 4||s||$

Theorem: a connected graph admits an Eulerian tour if and only if its vertices have even degree

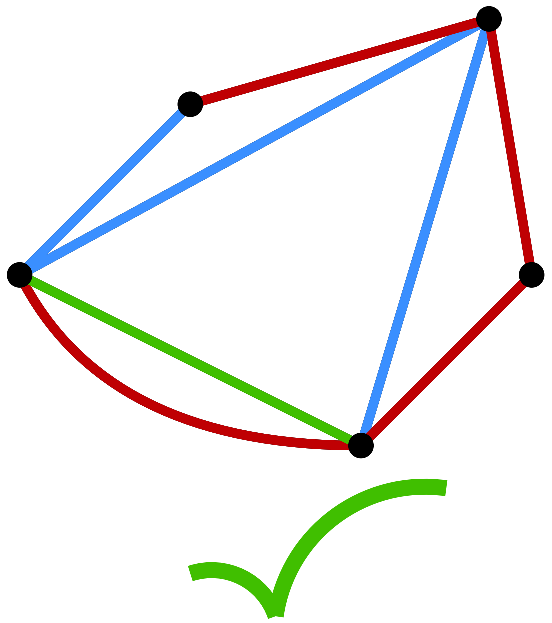


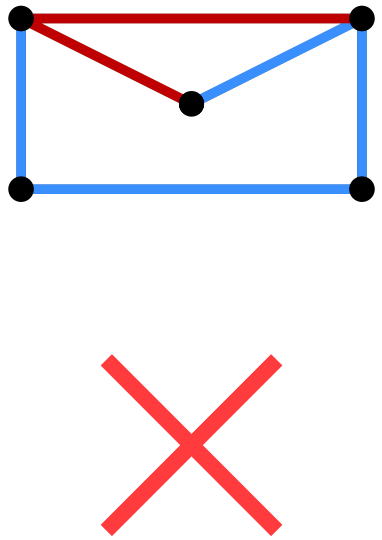


Patching Lemma

Patching lemma: closed curve π crossing segment s at least $k \geq 3$ times can be replaced by closed curve π' crossing s at most twice such that $||\pi'|| \leq ||\pi|| + 4||s||$

Theorem: a connected graph admits an Eulerian tour if and only if its vertices have even degree



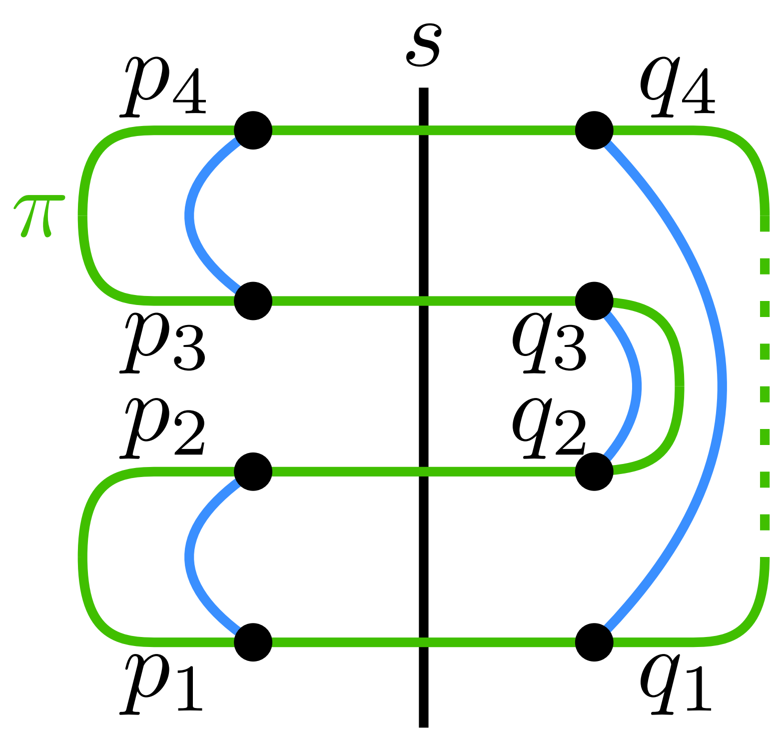


Patching Lemma

Patching lemma: closed curve π crossing segment s at least $k \geq 3$ times can be replaced by closed curve π' crossing s at most twice such that $||\pi'|| \leq ||\pi|| + 4||s||$

Theorem: a connected graph admits an Eulerian tour if and only if its vertices have even degree

Let E_π be the set of edges each representing a connected component of $\pi \setminus s$



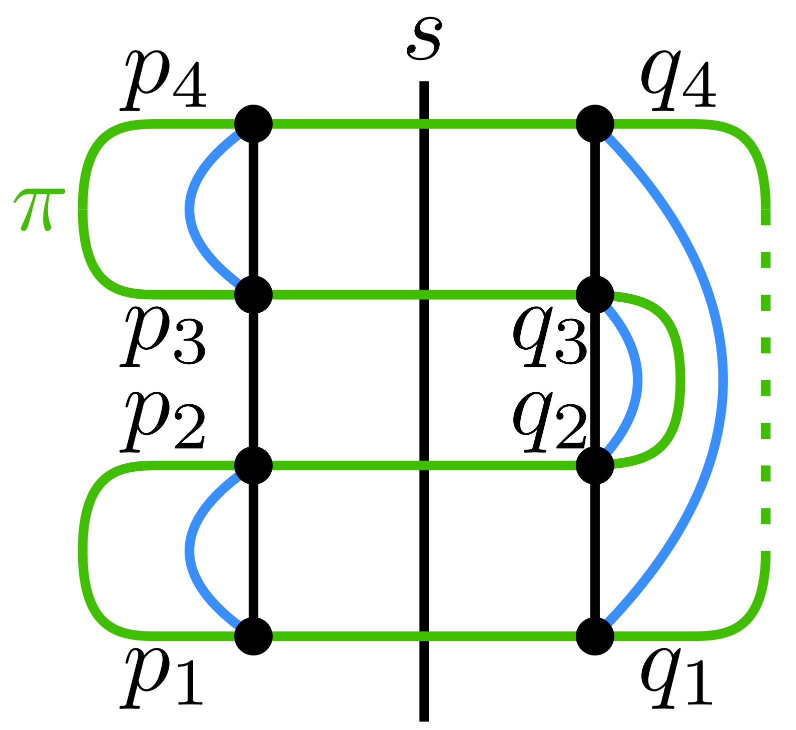
Patching Lemma

Patching lemma: closed curve π crossing segment s at least $k \geq 3$ times can be replaced by closed curve π' crossing s at most twice such that $||\pi'|| \leq ||\pi|| + 4||s||$

Theorem: a connected graph admits an Eulerian tour if and only if its vertices have even degree

Let E_π be the set of edges each representing a connected component of $\pi \setminus s$

Let $s_i = (p_i, p_{i+1})$ and $s'_i = (q_i, q_{i+1})$



Patching Lemma

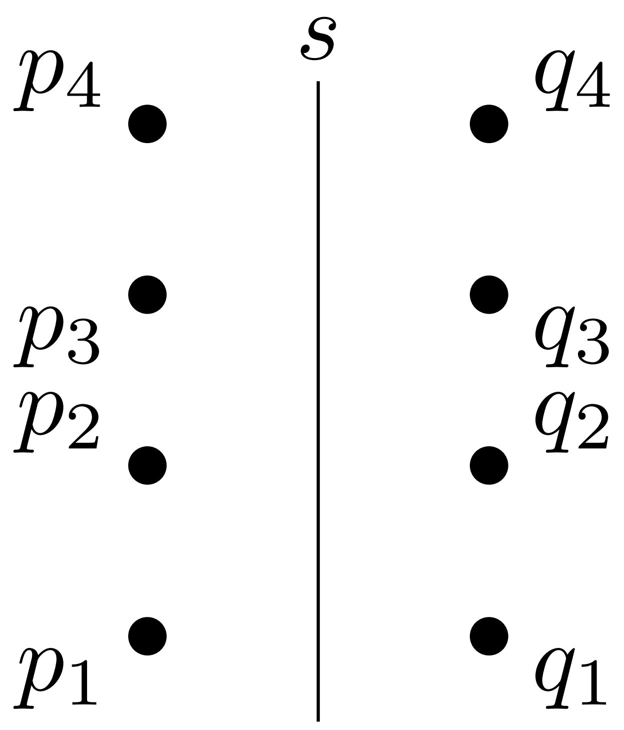
Patching lemma: closed curve π crossing segment s at least $k \geq 3$ times can be replaced by closed curve π' crossing s at most twice such that $||\pi'|| \leq ||\pi|| + 4||s||$

Theorem: a connected graph admits an Eulerian tour if and only if its vertices have even degree

Let E_π be the set of edges each representing a connected component of $\pi \setminus s$

Let $s_i = (p_i, p_{i+1})$ and $s'_i = (q_i, q_{i+1})$

Construct graph $G = (V, E)$ with $V = \bigcup_{i=1}^{k} \{p_i, q_i\}$



Patching Lemma

Patching lemma: closed curve π crossing segment s at least $k \geq 3$ times can be replaced by closed curve π' crossing s at most twice such that $||\pi'|| \leq ||\pi|| + 4||s||$

Theorem: a connected graph admits an Eulerian tour if and only if its vertices have even degree

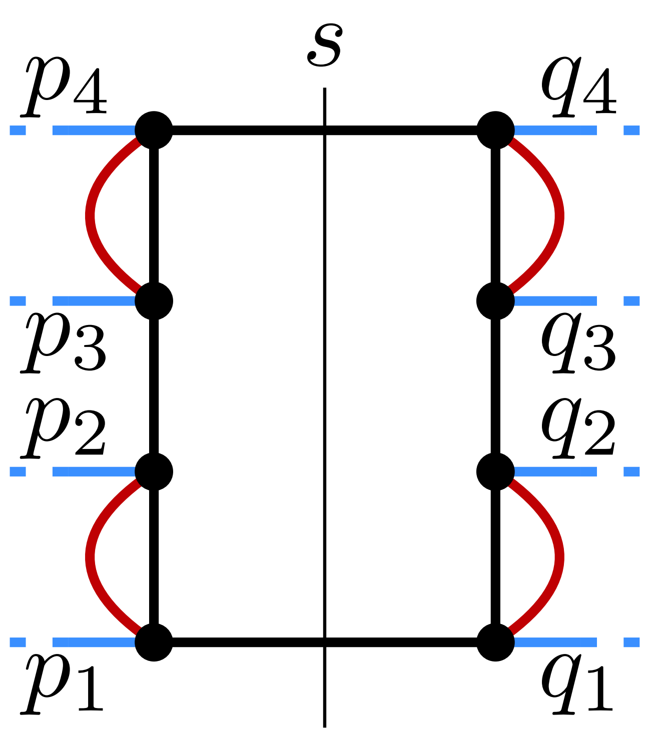
Let E_π be the set of edges each representing a connected component of $\pi \setminus s$

Let $s_i = (p_i, p_{i+1})$ and $s_i' = (q_i, q_{i+1})$

Construct graph $G = (V, E)$ with $V = \bigcup_{i=1}^{k} \{p_i, q_i\}$

$E = \bigcup_{i=1}^{k-1} \{s_i, s_i'\} \cup E_\pi \cup \bigcup_{i=1 (\text{odd})}^{k-1} \{s_i, s_i'\} \cup \{(p_1, q_1)\}$

and also add (p_k, q_k) if k is even



Patching Lemma

Patching lemma: closed curve π crossing segment s at least $k \geq 3$ times can be replaced by closed curve π' crossing s at most twice such that $||\pi'|| \leq ||\pi|| + 4||s||$

Theorem: a connected graph admits an Eulerian tour if and only if its vertices have even degree

Let E_π be the set of edges each representing a connected component of $\pi \setminus s$

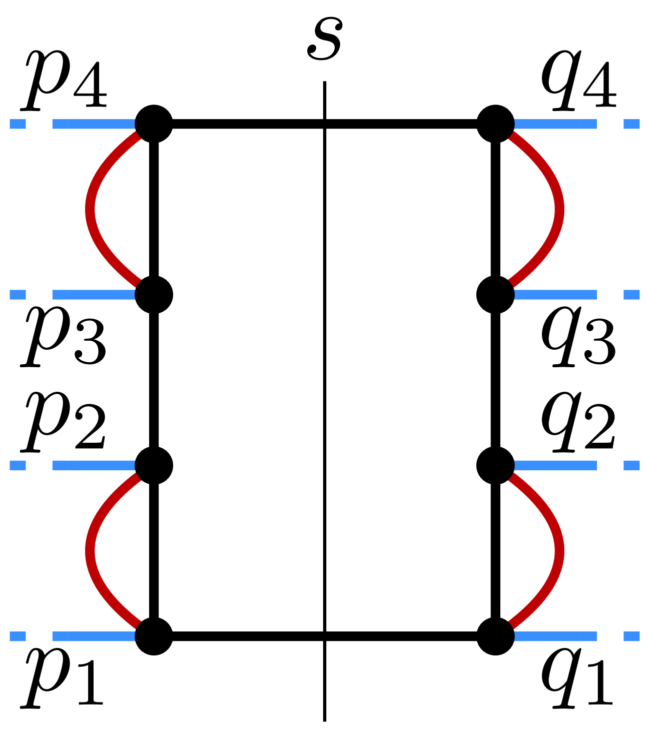
$E = \bigcup_{i=1}^{k-1} \{s_i, s_i'\} \cup E_\pi \cup \bigcup_{i=1(\text{odd})}^{k-1} \{s_i, s_i'\} \cup \{(p_1, q_1)\} (\cup \{(p_k, q_k)\})$

G is connected and its vertices have even degree

Hence it admits an Eulerian tour π'

π' visits all of π outside of s as $E_\pi \subseteq E$

π' crosses s at most twice: at (p_1, q_1) and (p_k, q_k)



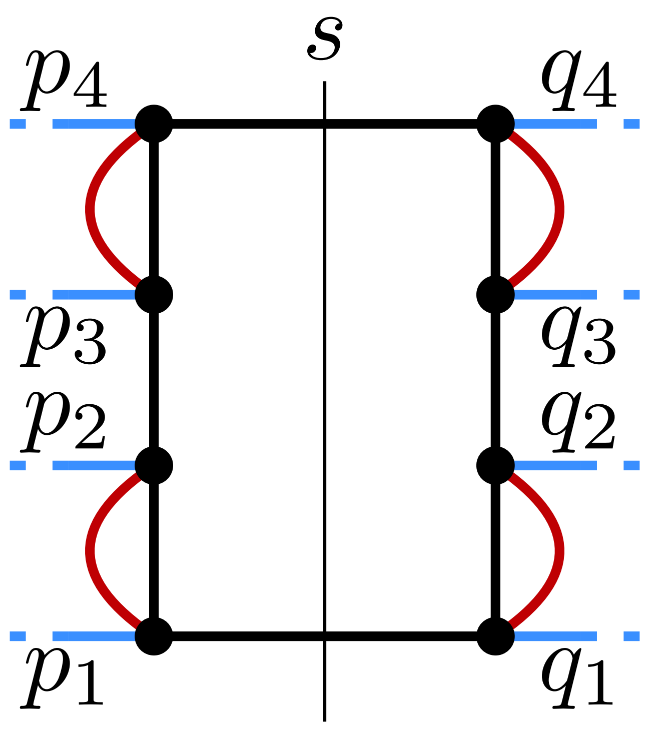
Patching Lemma

Patching lemma: closed curve π crossing segment s at least $k \geq 3$ times can be replaced by closed curve π' crossing s at most twice such that $||\pi'|| \leq ||\pi|| + 4||s||$

Theorem: a connected graph admits an Eulerian tour if and only if its vertices have even degree

$$E = \bigcup_{i=1}^{k-1} \{s_i, s_i'\} \cup E_\pi \cup \bigcup_{i=1 (\text{odd})}^{k-1} \{s_i, s_i'\} \cup \{(p_1, q_1)\} (\cup \{(p_k, q_k)\})$$

$||\pi'||$ is the length of all edges in E



Patching Lemma

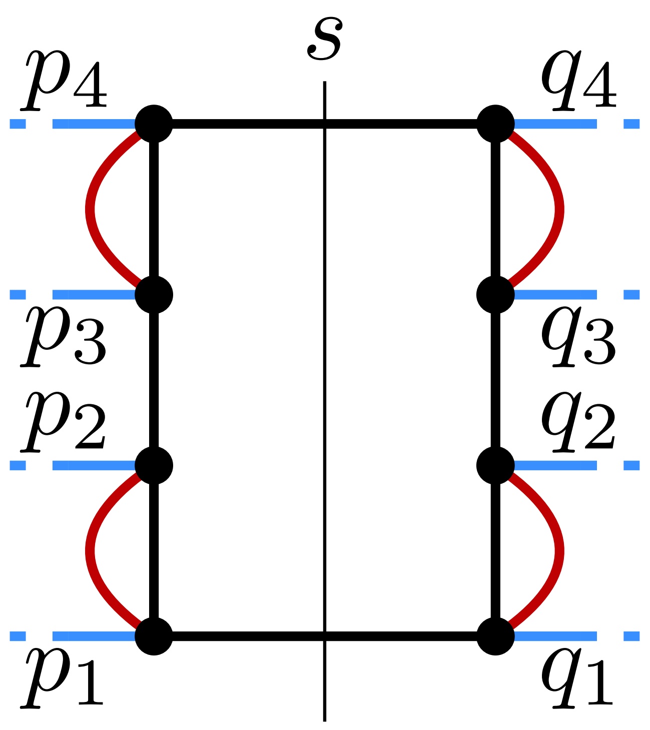
Patching lemma: closed curve π crossing segment s at least $k \geq 3$ times can be replaced by closed curve π' crossing s at most twice such that $||\pi'|| \leq ||\pi|| + 4||s||$

Theorem: a connected graph admits an Eulerian tour if and only if its vertices have even degree

$$E = \underbrace{\bigcup_{i=1}^{k-1}\{s_i, s_i'\}}_{\leq 2||s||} \cup \underbrace{E_\pi}_{||\pi||} \cup \underbrace{\bigcup_{i=1\text{(odd)}}^{k-1}\{s_i, s_i'\}}_{\leq 2||s||} \cup \underbrace{\{(p_1, q_1)\}}_{0}(\cup\underbrace{\{(p_k, q_k)\}}_{0})$$

$||\pi'||$ is the length of all edges in E

$||\pi'|| \leq ||\pi|| + 2||s||| + 2||s|| = ||\pi|| + 4||s||$



Patching Lemma: Consequences

Patching lemma: closed curve π crossing segment s at least $k \geq 3$ times can be replaced by closed curve π' crossing s at most twice such that $||\pi'|| \leq ||\pi|| + 4||s||$

Patching Lemma: Consequences

Patching lemma: closed curve π crossing segment s at least $k \geq 3$ times can be replaced by closed curve π' crossing s at most twice such that $||\pi'|| \leq ||\pi|| + 4||s||$

Using portals only twice

Patching Lemma: Consequences

Patching lemma: closed curve π crossing segment s at least $k \geq 3$ times can be replaced by closed curve π' crossing s at most twice such that $||\pi'|| \leq ||\pi|| + 4||s||$

Using portals only twice

apply lemma to portals $\leftarrow$ segments of length 0

Patching Lemma: Consequences

Patching lemma: closed curve π crossing segment s at least $k \geq 3$ times can be replaced by closed curve π' crossing s at most twice such that $||\pi'|| \leq ||\pi|| + 4||s||$

Using portals only twice

apply lemma to portals $\leftarrow$ segments of length 0

Using only k portals of a square

bottom-up (i.e. starting with small cells): When $> k$ intersections, patch!

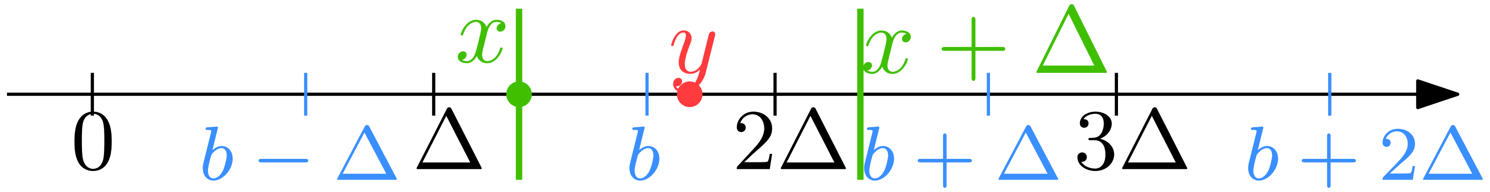
intuition: patching on low levels: relatively cheap, and also helps higher levels

+ fewer (exponentially decreasing) intersections at higher levels: shifting

Shifted Grids

recall: shifted partition of real line

Let $\Delta > 0$ and $b \in [0, \Delta]$ uniformly distributed. We shift the grid G_Δ by b



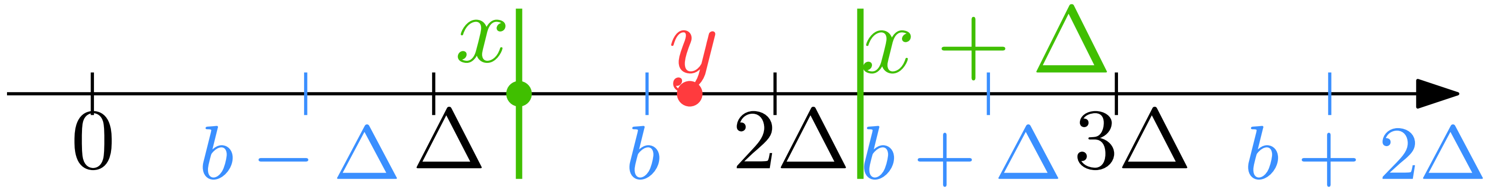
$$h_{b,\Delta}(x) = \lfloor \frac{x-b}{\Delta} \rfloor$$

Lemma: For $x, y \in \mathbb{R}$ holds $\mathbb{P}\left[h_{b,\Delta}(x) \neq h_{b,\Delta}(y)\right] = min\left(\frac{|x-y|}{\Delta}, 1\right)$

Shifted Grids

recall: shifted partition of real line

Let $\Delta > 0$ and $b \in [0, \Delta]$ uniformly distributed. We shift the grid G_Δ by b



$$h_{b,\Delta}(x) = \lfloor \frac{x-b}{\Delta} \rfloor$$

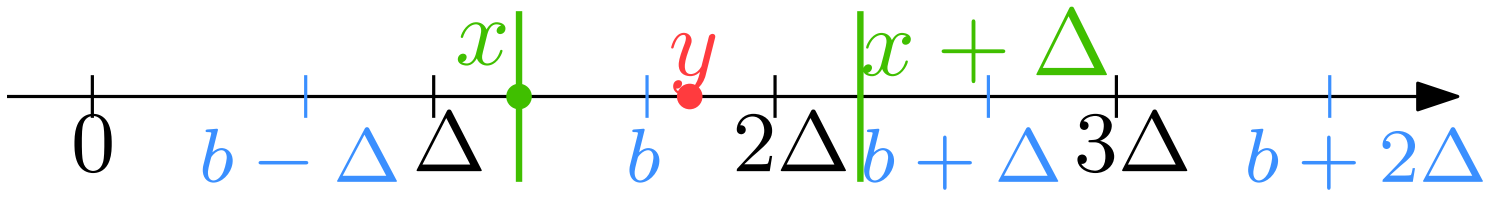
Lemma: For $x, y \in \mathbb{R}$ holds $\mathbb{P}\left[h_{b,\Delta}(x) \neq h_{b,\Delta}(y)\right] = min\left(\frac{|x-y|}{\Delta}, 1\right)$

. . . generalizes to grids and quadtrees . . .

Shifted Grids

recall: shifted partition of real line

Let $\Delta > 0$ and $b \in [0, \Delta]$ uniformly distributed. We shift the grid G_Δ by b



$$h_{b,\Delta}(x) = \lfloor \frac{x-b}{\Delta} \rfloor$$

Lemma: For $x, y \in \mathbb{R}$ holds $\mathbb{P}\left[h_{b,\Delta}(x) \neq h_{b,\Delta}(y)\right] = min\left(\frac{|x-y|}{\Delta}, 1\right)$

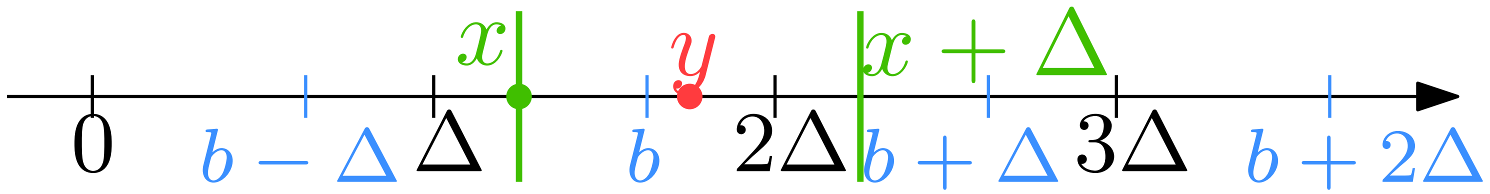
. . . generalizes to grids and quadtrees . . .

Lemma: Let s be a segment in the plane, The probability that s intersects the shifted grid of side length Δ is at most $\sqrt{2}\|s\|/\Delta$.

Shifted Grids

recall: shifted partition of real line

Let $\Delta > 0$ and $b \in [0, \Delta]$ uniformly distributed. We shift the grid G_Δ by b



$$h_{b,\Delta}(x) = \lfloor \tfrac{x-b}{\Delta} \rfloor$$

Lemma: For $x, y \in \mathbb{R}$ holds $\mathbb{P}\left[h_{b,\Delta}(x) \neq h_{b,\Delta}(y)\right] = min\left(\frac{|x-y|}{\Delta}, 1\right)$

. . . generalizes to grids and quadtrees . . .

Lemma: Let s be a segment in the plane, The probability that s intersects the shifted grid of side length Δ is at most $\sqrt{2}\|s\|/\Delta$.

Furthermore the expected number of intersection of s with vertical and horizonal lines of G_Δ is in the range $\left[\|s\|/\Delta, \sqrt{2}\|s\|/\Delta\right]$

Shifted Grids: Consequences

patching lemma + shifted grids $\rightarrow k$ portals per cell suffice

(see patching lemma)

Shifted Grids: Consequences

patching lemma + shifted grids $\rightarrow k$ portals per cell suffice

(see patching lemma)

expected snapping error $\leq \frac{2H}{m+1} \|\pi_{opt}\|$

Shifted Grids: Consequences

patching lemma + shifted grids $\rightarrow k$ portals per cell suffice

(see patching lemma)

expected snapping error $\leq \frac{2H}{m+1} \|\pi_{opt}\|$

error for one intersection with an edge e of a cell $\leq 2 \frac{\|e\|}{2(m+1)} = \|e\|/(m+1)$

Shifted Grids: Consequences

patching lemma + shifted grids $\rightarrow k$ portals per cell suffice

(see patching lemma)

expected snapping error $\leq \frac{2H}{m+1}\|\pi_{opt}\|$

error for one intersection with an edge e of a cell $\leq 2\frac{\|e\|}{2(m+1)} = \|e\|/(m+1)$

expected error for all intersections with a grid with side length Δ:

$\leq \frac{\sqrt{2}\|\pi_{opt}\|}{\Delta}\frac{\Delta}{m+1} \leq 2\|\pi_{opt}\|/(m+1)$

Shifted Grids: Consequences

patching lemma + shifted grids $\rightarrow k$ portals per cell suffice

(see patching lemma)

expected snapping error $\leq \frac{2H}{m+1} \|\pi_{opt}\|$

error for one intersection with an edge e of a cell $\leq 2\frac{\|e\|}{2(m+1)} = \|e\|/(m+1)$

expected error for all intersections with a grid with side length Δ:

$\leq \frac{\sqrt{2}\|\pi_{opt}\|}{\Delta} \frac{\Delta}{m+1} \leq 2\|\pi_{opt}\|/(m+1)$

add over H levels of quadtree

Quality of Approximation

Introduced error when:

- snapping to the grid
- bounding the number of intersections at k per side of each square
  uses: patching lemma (+ shifting)
- requiring the use of portals
  uses: shifting

Quality of Approximation

Introduced error when:

- snapping to the grid $1 + \frac{\varepsilon}{2}$
- bounding the number of intersections at k per side of each square $1 + \frac{8}{k-2}$
  uses: patching lemma (+ shifting)
- requiring the use of portals $1 + \frac{2H}{m+1}$
  uses: shifting

Recall: $k = \frac{90}{\varepsilon}$ and $m \geq \frac{20H}{\varepsilon}$

Quality of Approximation

Introduced error when:

- snapping to the grid $1+\frac{\varepsilon}{2}$
- bounding the number of intersections at k per side of each square $1+\frac{8}{k-2}$
  uses: patching lemma (+ shifting)
- requiring the use of portals $1+\frac{2H}{m+1}$
  uses: shifting

Recall: $k = \frac{90}{\varepsilon}$ and $m \geq \frac{20H}{\varepsilon}$

$$\left(1+\frac{\varepsilon}{2}\right)\left(1+\frac{8}{k-2}\right)\left(1+\frac{2H}{m+1}\right)||\pi_{\mathrm{OPT}}|| \leq \left(1+\frac{\varepsilon}{2}\right)\left(1+\frac{\varepsilon}{10}\right)^2||\pi_{\mathrm{OPT}}||$$

$$\leq (1+\varepsilon)||\pi_{\mathrm{OPT}}||$$

Summary

shifted quadtree with points snapped to grid

dynamic programming on quadtrees

running time: not too many subproblems, since subsquares only connect at few portals

correctness: patching lemma + shifting + snap to grid

Summary

shifted quadtree with points snapped to grid

dynamic programming on quadtrees

running time: not too many subproblems, since subsquares only connect at few portals

correctness: patching lemma + shifting + snap to grid

For a set P of n points in $\mathbb{R}^2$ and $\varepsilon > 0$, we can compute a tour π over P with expected length $(1+\varepsilon)||\pi_{\text{OPT}}||$ in time $(\varepsilon^{-1}\log n)^{\mathcal{O}(1/\varepsilon)}$