INF562 – Géométrie Algorithmique et Applications

Algorithmes d'approximation géométrique

Steve Oudot

ÉCOLE POLYTECHNIQUE

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

INRIA

centre de recherche SACLAY – ÎLE-DE-FRANCE

Préliminaire : un peu de théorie de la complexité

Principe de cette théorie :

- on prend un problème (exemple : unicité des éléments d'un ensemble)

- on choisit un modèle de calcul (exemple : comparaison)

- on étudie la difficulté de ce problème pour ce modèle :

$\rightarrow$ borne inférieure : $\Omega(n \log n)$ [Ben-Or 83]

$\rightarrow$ borne supérieure : $O(n \log n)$ (on trie les elements au fur et à mesure)

Préliminaire : un peu de théorie de la complexité

Principe de cette théorie :

- on prend un problème (exemple : unicité des éléments d'un ensemble)

- on choisit un modèle de calcul (exemple : Word-RAM)

- on étudie la difficulté de ce problème pour ce modèle :

$\rightarrow$ borne inférieure : $\Omega(n)$

$\rightarrow$ borne supérieure : $O(n)$ amorti (table de hachage)

Préliminaire : un peu de théorie de la complexité

Principe de cette théorie :

- on prend un problème (exemple : unicité des éléments d'un ensemble)

- on choisit un modèle de calcul (exemple : quantum circuits)

- on étudie la difficulté de ce problème pour ce modèle :

$\rightarrow$ borne inférieure : $\Omega(n^{2/3})$ [Aaronson, Shi 04])

$\rightarrow$ borne supérieure : $O(n^{2/3})$ (Prob(résultat correct)$= \Omega(1)$ [Ambainis 07]

Note: on ne compte pas le temps de lire l'entrée

Préliminaire : un peu de théorie de la complexité

Principe de cette théorie :

- on prend un problème (exemple : unicité des éléments d'un ensemble)

- on choisit un modèle de calcul (exemple : quantum circuits)

- on étudie la difficulté de ce problème pour ce modèle :

$\rightarrow$ borne inférieure : $\Omega(n^{2/3})$ [Aaronson, Shi 04])

$\rightarrow$ borne supérieure : $O(n^{2/3})$ (Prob(résultat correct)$= \Omega(1)$ [Ambainis 07]

Quel modèle de calcul choisir ?

Modèles de calcul

Hiérarchie de Chomsky (1956):

- langages réguliers (automates finis)

  exemple: $\{a^*b^*\}$

- grammaires hors-contexte (automates finis à pile)

  exemple : $\{a^n b^n \mid n > 0\}$

- grammaires contextuelles (automates finis a bande linéairement bornée)

  exemple : $\{a^n b^n c^n \mid n > 0\}$

- langages décidables (machines de Turing)

  exemple : $\{(n, m) \mid n \mod m = 0\}$

Modèles de calcul

Hiérarchie de Chomsky (1956):

- langages réguliers (automates finis)

  exemple: $\{a^*b^*\}$

- grammaires hors-contexte (automates finis à pile)

  exemple : $\{a^n b^n \mid n > 0\}$

- grammaires contextuelles (automates finis a bande linéairement bornée)

  exemple : $\{a^n b^n c^n \mid n > 0\}$

- langages décidables (machines de Turing)

  exemple : $\{(n, m) \mid n \mod m = 0\}$

au-delà : langages récursivement énumérables, langages indécidables

Modèles de calcul

Thèse de Church-Turing (1936):

> *Tout modèle de calcul* effectif [i.e. réalisable par une machine] *peut être simulé par une machine de Turing*

Modèles de calcul

Thèse de Church-Turing (1936):

> *Tout modèle de calcul* effectif [i.e. réalisable par une machine] *peut être simulé par une machine de Turing*

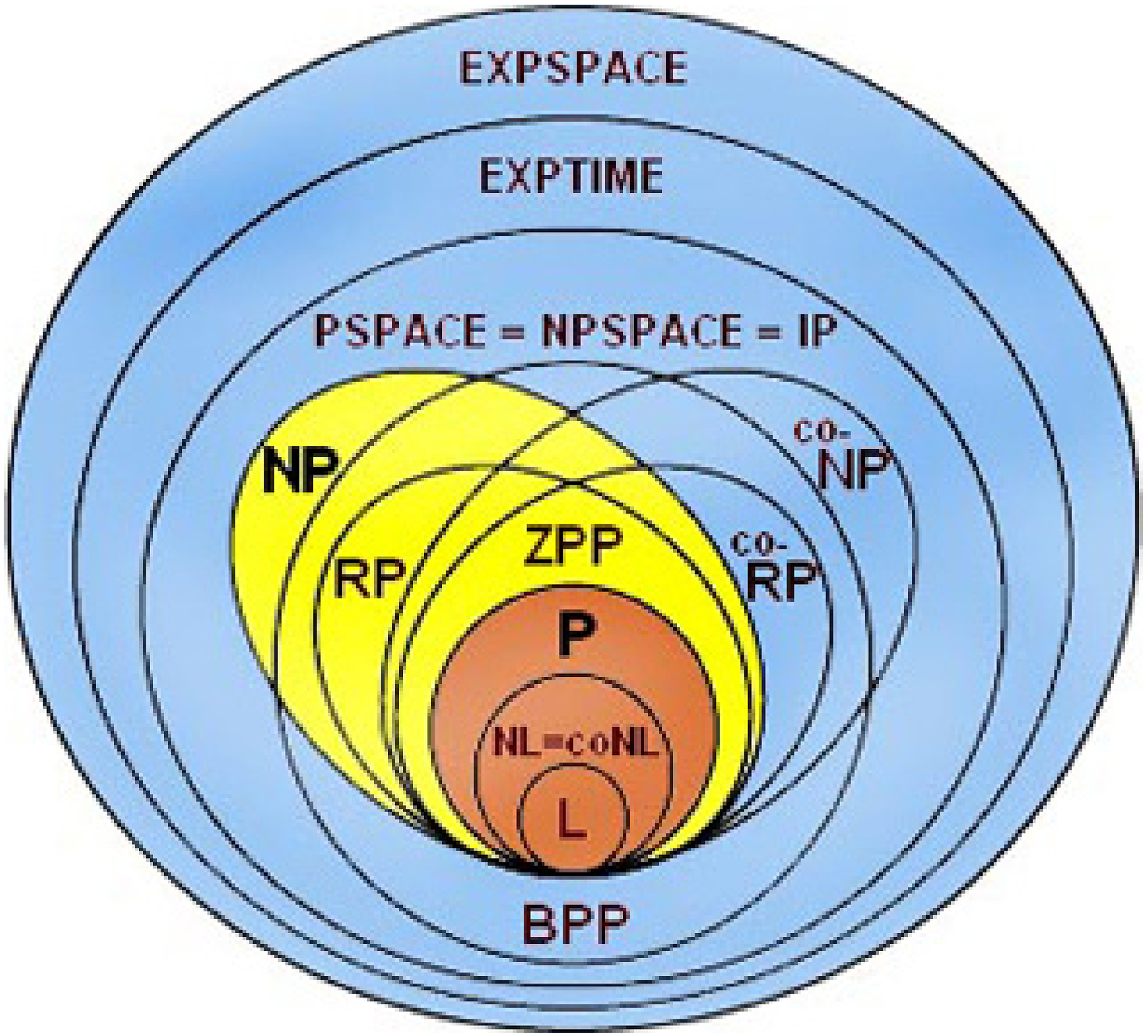
$\rightarrow$ modèle de référence : machine de Turing

$\rightarrow$ modèles équivalents (pour la calculabilité):

> λ-calcul, algebraic comparison, Pointer Machine, Real-RAM, Word-RAM, etc.

$\rightarrow$ but : classifier les problèmes de manière consistante pour tous ces modèles

Classes de complexité



Classes de complexité

$$\boxed{\mathsf{P} \stackrel{?}{=} \mathsf{NP}}$$

- enjeux colossaux dépassant largement le cadre scientifique
- implications dans de nombreux domaines des sciences et au-delà
- que faire si $\mathsf{P} \neq \mathsf{NP}$? Certains problèmes sont-ils définitivement inaccessibles ?

Classes de complexité

$$\boxed{\mathsf{P} \stackrel{?}{=} \mathsf{NP}}$$

- enjeux colossaux dépassant largement le cadre scientifique

- implications dans de nombreux domaines des sciences et au-delà

- que faire si $\mathsf{P} \neq \mathsf{NP}$? Certains problèmes sont-ils définitivement inaccessibles ?

  $\rightarrow$ calcul quantique ?

  $\rightarrow$ algorithmes d'approximation

Classes de complexité

$$\boxed{\mathsf{P} \stackrel{?}{=} \mathsf{NP}}$$

- enjeux colossaux dépassant largement le cadre scientifique

- implications dans de nombreux domaines des sciences et au-delà

- que faire si $\mathsf{P} \neq \mathsf{NP}$? Certains problèmes sont-ils définitivement inaccessibles ?

  $\rightarrow$ calcul quantique ?

  $\rightarrow$ algorithmes d'approximation

Algorithmes d'approximation

Def: Un problème d'optimisation (minimisation) p consiste, à partir d'une instance i donnée, à calculer une structure qui optimise (minimise) une certaine fonction de coût $|\cdot|$.

Algorithmes d'approximation

Def: Un problème d'optimisation (minimisation) p consiste, à partir d'une instance i donnée, à calculer une structure qui optimise (minimise) une certaine fonction de coût $|\cdot|$.

Exemple: étant donné un graphe pondéré G, calculer l'arbre couvrant de G de poids minimal.

$\rightarrow$ problème : MST

$\rightarrow$ instance : G

$\rightarrow$ fonction de coût : somme des poids des arêtes de l'arbre

Algorithmes d'approximation

Def: Un problème d'optimisation (minimisation) p consiste, à partir d'une instance i donnée, à calculer une structure qui optimise (minimise) une certaine fonction de coût $|\cdot|$.

$\rightarrow$ pour toute instance i, soit $\mathrm{OPT}_p(i)$ une structure optimale sur i

Def: Un algorithme de κ-approximation de $(p, |\cdot|)$ est un algorithme A qui, pour toute instance i de p, fournit une solution $A(i)$ (en temps Poly($|i|$)) telle que $|A(i)| \leq \kappa |\mathrm{OPT}_p(i)|$.

$\rightarrow$ κ peut être une constante ou bien une fonction (polynômiale) de $|i|$

Algorithmes d'approximation

Def: Un problème d'optimisation (minimisation) p consiste, à partir d'une instance i donnée, à calculer une structure qui optimise (minimise) une certaine fonction de coût $|\cdot|$.

$\rightarrow$ pour toute instance i, soit $\mathrm{OPT}_p(i)$ une structure optimale sur i

Def: Un algorithme de κ-approximation de $(p, |\cdot|)$ est un algorithme A qui, pour toute instance i de p, fournit une solution $A(i)$ (en temps Poly($|i|$)) telle que $|A(i)| \leq \kappa |\mathrm{OPT}_p(i)|$.

$\rightarrow$ κ peut être une constante ou bien une fonction (polynômiale) de $|i|$

Def (PTAS): Un PTAS de p vis-à-vis de $|\cdot|$ est une famille à un paramètre d'algorithmes polynômiaux, $\{A_\varepsilon\}_{\varepsilon>0}$, telle que pour tout $\varepsilon > 0$ et toute instance i de p, $|A_\varepsilon(i)| \leq (1+\varepsilon)|\mathrm{OPT}_p(i)|$.

Algorithmes d'approximation

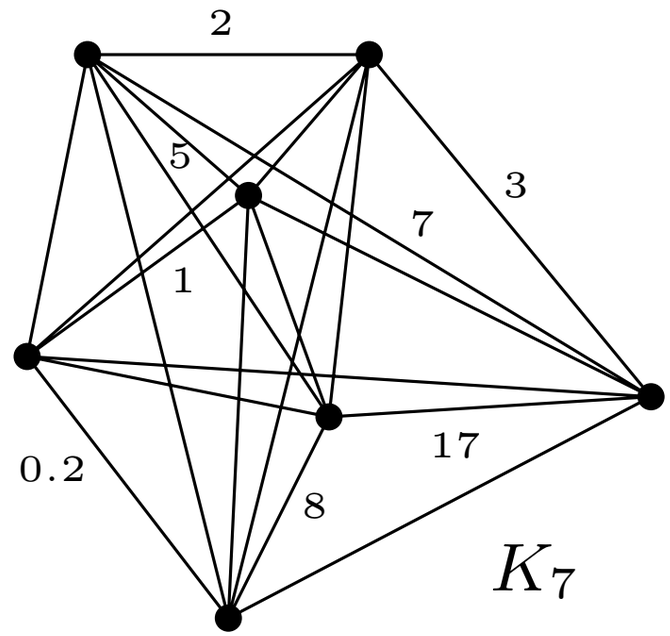
Def: Un problème d'optimisation (minimisation) p consiste, à partir d'une instance i donnée, à calculer une structure qui optimise (minimise) une certaine fonction de coût $|\cdot|$.

Plus généralement, la raison d'être d'un algorithme d'approximation (ou d'une famille d'algorithmes d'approximation) est de réduire la complexité d'un problème donné pour un modèle de calcul donné.

$\rightarrow$ dans la suite, le modèle de calcul par défaut sera Real-RAM

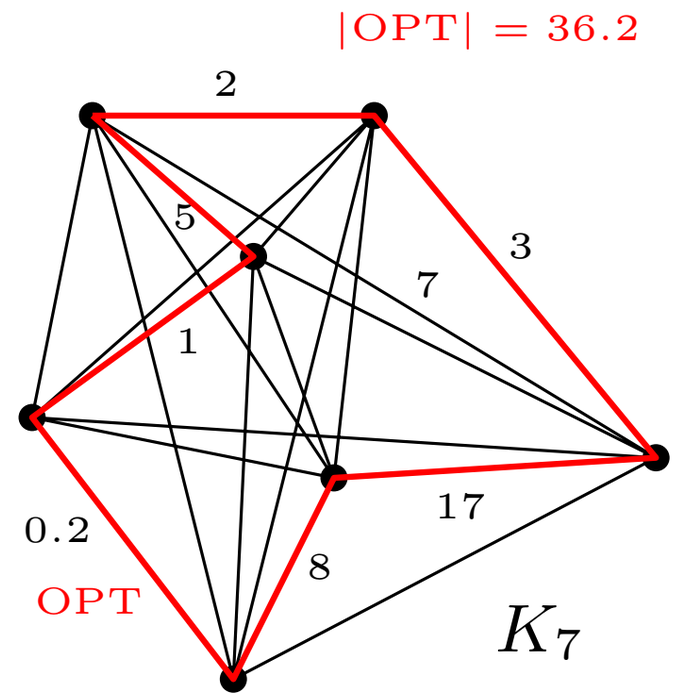
Traveling Salesman Problem

TSP: Given a complete graph $G = (V, E)$ with non-negative weights, find the **tour** (cycle passing through each vertex once) of minimum total cost.



Traveling Salesman Problem

TSP: Given a complete graph $G = (V, E)$ with non-negative weights, find the **tour** (cycle passing through each vertex once) of minimum total cost.



Traveling Salesman Problem

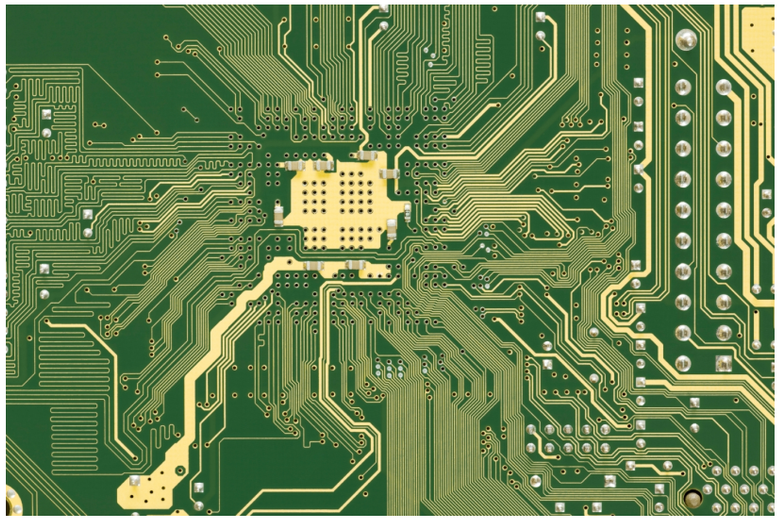
A fundamental problem in computer science:

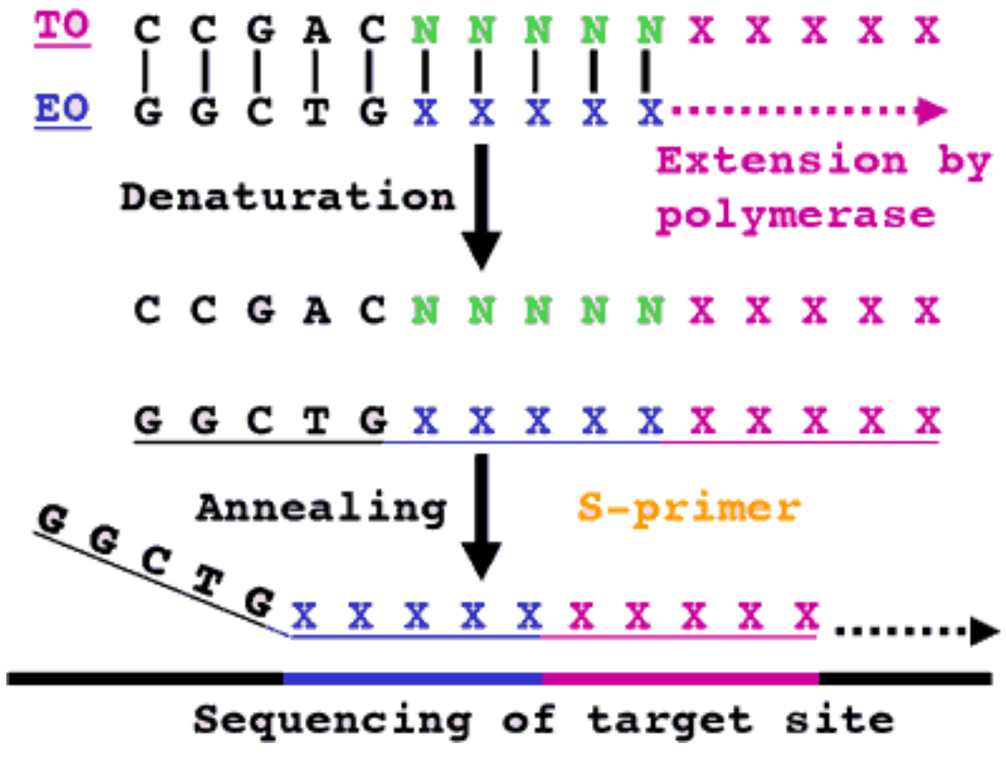
- among the most intensively studied problems over the last 50 years
- a special case (Hamiltonian Cycle) is among Karp's 21 NP-complete problems
- optimization problem with an interesting complexity
- longstanding history (roots go back to the 1800's) with several breakthroughs

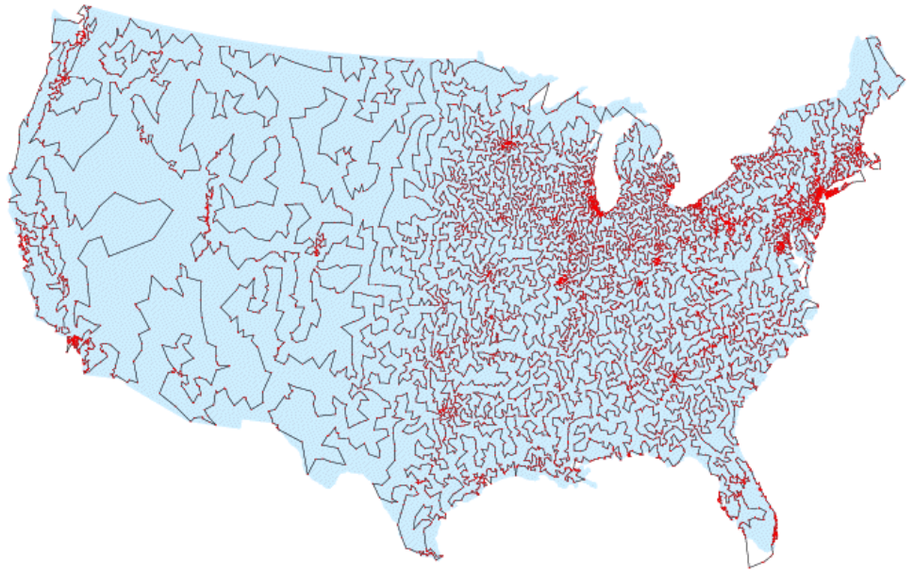
Traveling Salesman Problem

A wealth of applications:

- route planning
- transportation and logistics
- DNA sequencing
- printed circuits and microchips manufacturing
- etc.



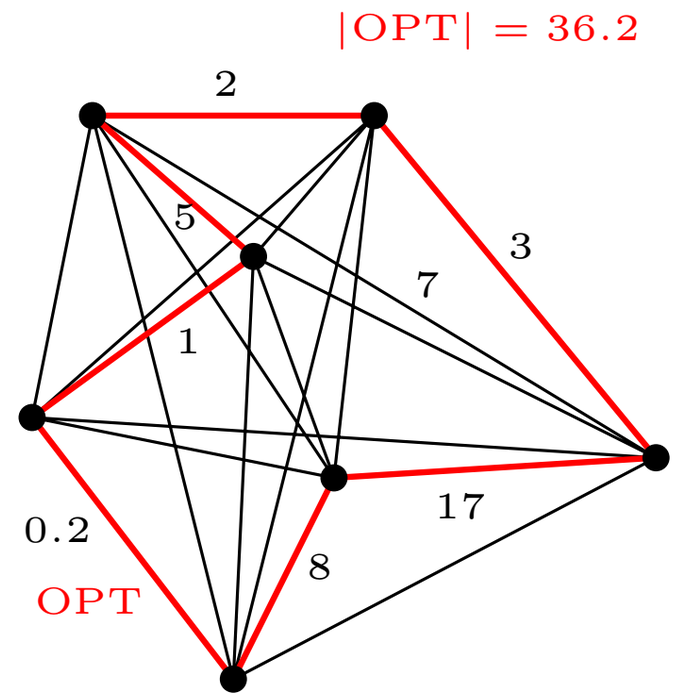




Source: TSP project @ Rice University

Traveling Salesman Problem

Decisive TSP: given a complete graph $G = (V, E)$ with non-negative weights, and an integer k, determine whether there is a tour of cost $\leq k$.

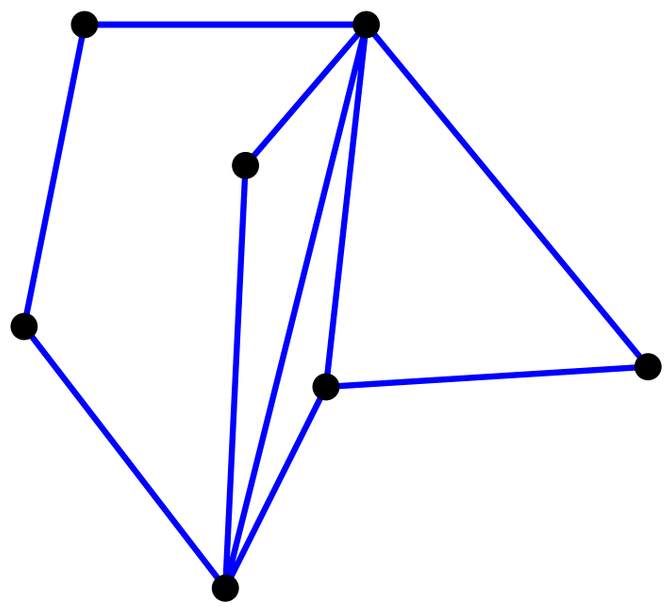


Thm: The decision version of TSP is NP-hard, i.e. it cannot be solved in PT unless $P = NP$.

Traveling Salesman Problem

Decisive TSP: given a complete graph $G = (V, E)$ with non-negative weights, and an integer k, determine whether there is a tour of cost $\leq k$.

Hamiltonian Cycle: given a graph $G = (V, E)$, determine whether G admits a tour.



Thm: The decision version of TSP is NP-hard, i.e. it cannot be solved in PT unless $P = NP$.

Proof Reduction of **Hamiltonian Cycle**:
Let $G = (V, E)$ unweighted, incomplete $\rightarrow G' = (V', E')$ where:

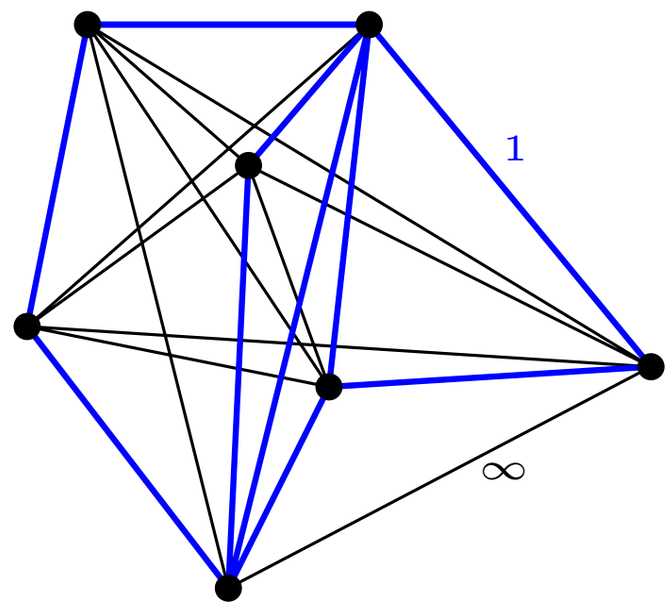
- $V' = V$
- $\forall e \in E$, add $(e, 1)$ to E'
- $\forall e \notin E$, add (e, ∞) to E'

G has a Hamiltonian cycle $\Leftrightarrow G'$ has a tour of length $\leq n$

Traveling Salesman Problem

Decisive TSP: given a complete graph $G = (V, E)$ with non-negative weights, and an integer k, determine whether there is a tour of cost $\leq k$.

Hamiltonian Cycle: given a graph $G = (V, E)$, determine whether G admits a tour.



Thm: The decision version of TSP is NP-hard, i.e. it cannot be solved in PT unless $P = NP$.

Proof Reduction of **Hamiltonian Cycle**:
Let $G = (V, E)$ unweighted, incomplete $\rightarrow G' = (V', E')$ where:

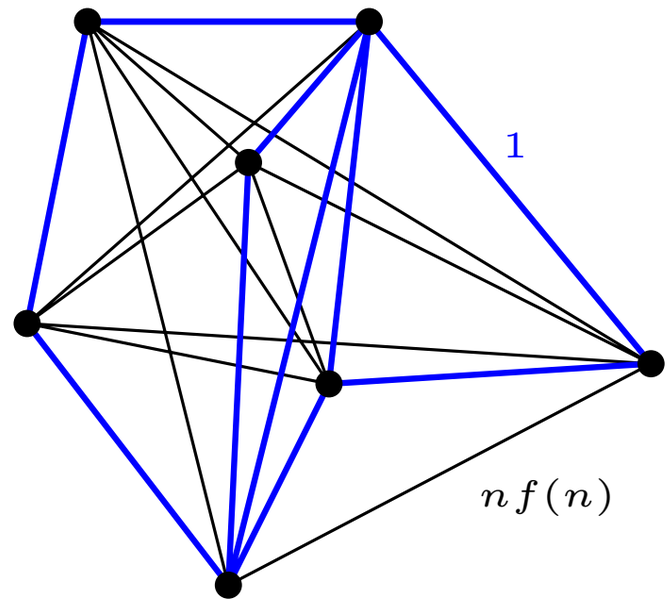
- $V' = V$
- $\forall e \in E$, add $(e, 1)$ to E'
- $\forall e \notin E$, add (e, ∞) to E'

G has a Hamiltonian cycle $\Leftrightarrow G'$ has a tour of length $\leq n$

Traveling Salesman Problem

Decisive TSP: given a complete graph $G = (V, E)$ with non-negative weights, and an integer k, determine whether there is a tour of cost $\leq k$.

Hamiltonian Cycle: given a graph $G = (V, E)$, determine whether G admits a tour.



Thm: TSP is APX-hard, i.e. for any PT computable function $f(n)$, TSP cannot be approximated in PT within a factor of $f(n)$, unless $P = NP$.

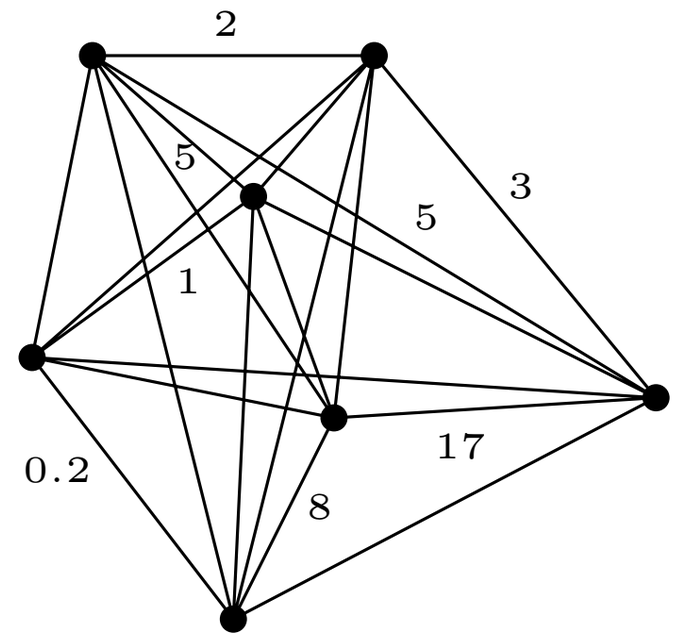
Proof Reduction of **Hamiltonian Cycle**:
Let $G = (V, E)$ unweighted, incomplete $\rightarrow G' = (V', E')$ where:

- $V' = V$
- $\forall e \in E$, add $(e, 1)$ to E'
- $\forall e \notin E$, add $(e, nf(n))$ to E'

G has a Hamiltonian cycle $\Leftrightarrow G'$ has a tour of length $\leq n$
(all tours passing in $G' \setminus G$ are at least $n - 1 + nf(n) > nf(n) = f(n)|OPT|$ long)

Metric TSP

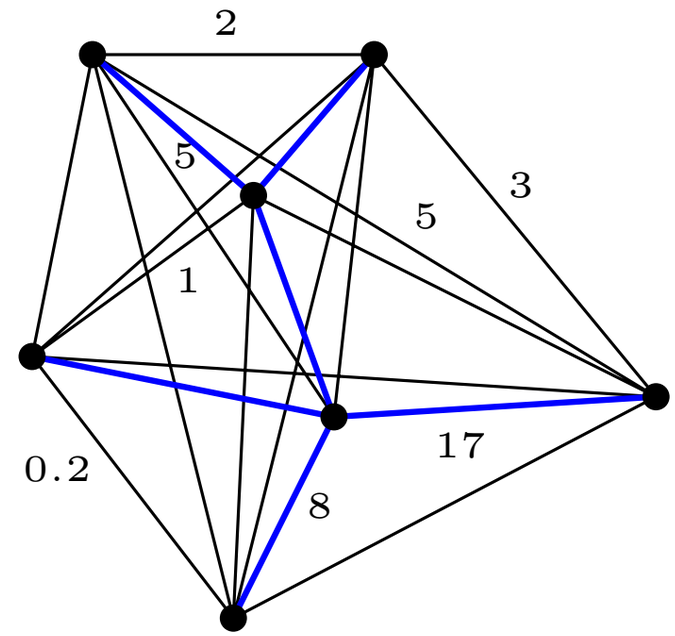
The weights of $G(V, E)$ now satisfy the triangle inequality



Metric TSP

2-approximation algorithm:

(1) build MST M of G (Kruskal)

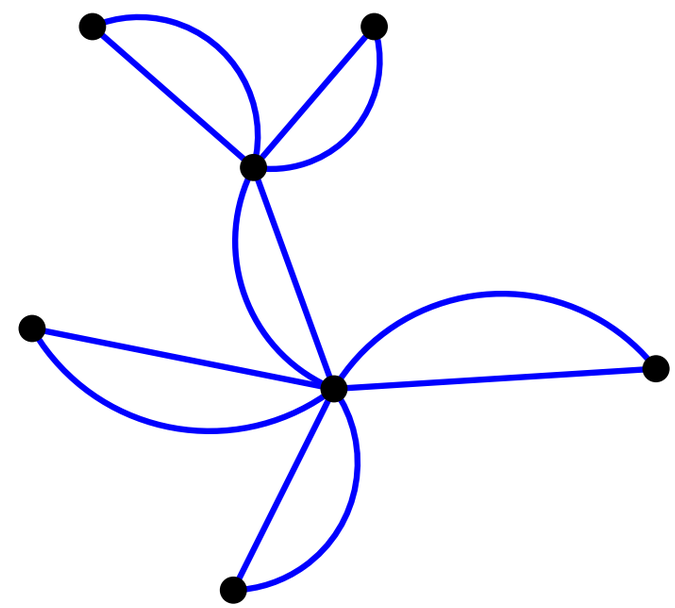


Metric TSP

2-approximation algorithm:

(1) build MST M of G (Kruskal)

(2) double edges $\rightarrow M^+$ Eulerian



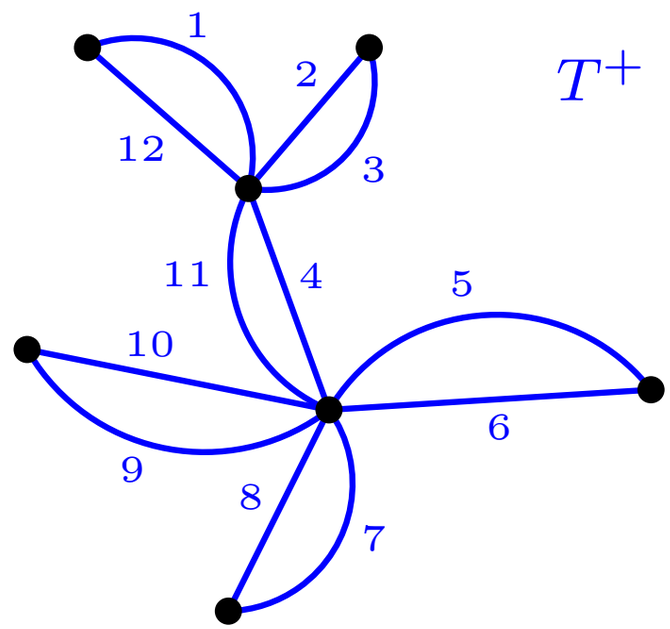
Metric TSP

2-approximation algorithm:

(1) build MST M of G (Kruskal)

(2) double edges $\rightarrow M^+$ Eulerian

(3) build greedily a Eulerian tour T^+ on M^+



Metric TSP

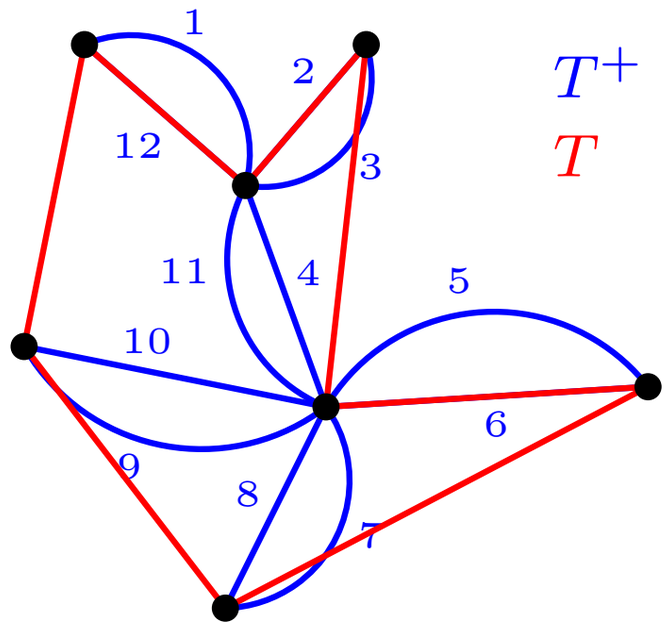
2-approximation algorithm:

(1) build MST M of G (Kruskal)

(2) double edges $\rightarrow M^+$ Eulerian

(3) build greedily a Eulerian tour T^+ on M^+

(4) Trim edges of $T^+ \rightarrow T$



Metric TSP

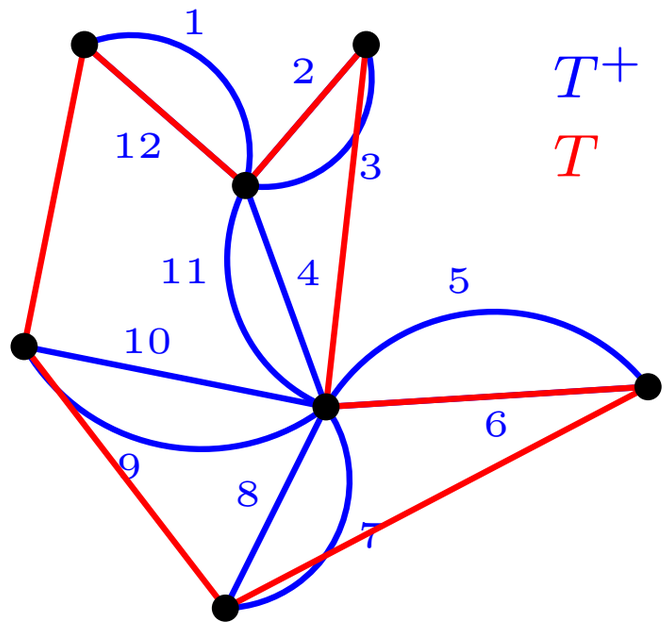
2-approximation algorithm:

(1) build MST M of G (Kruskal)

(2) double edges $\rightarrow M^+$ Eulerian

(3) build greedily a Eulerian tour T^+ on M^+

(4) Trim edges of $T^+ \rightarrow T$



Thm $|T| \leq 2|\mathrm{OPT}|$

proof $|T| \leq |T^+|$

tri. ineq.

Metric TSP

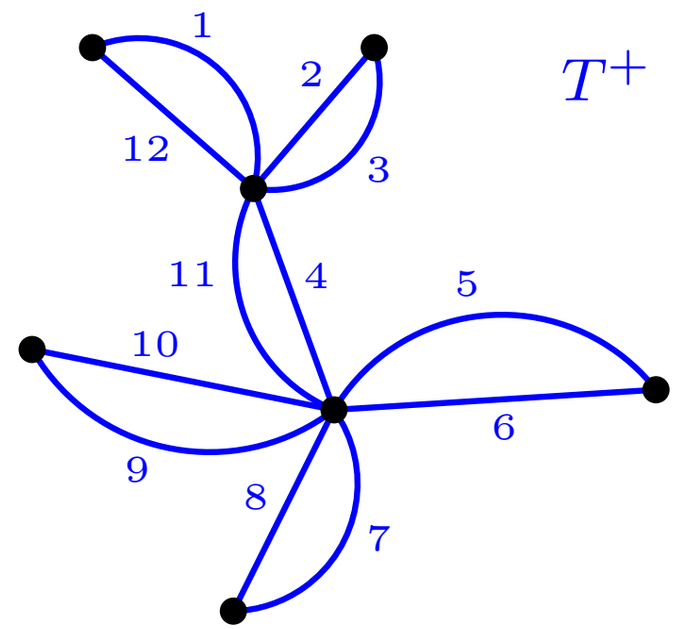
2-approximation algorithm:

(1) build MST M of G (Kruskal)

(2) double edges $\rightarrow M^+$ Eulerian

(3) build greedily a Eulerian tour T^+ on M^+

(4) Trim edges of $T^+ \rightarrow T$



Thm $|T| \leq 2|\mathrm{OPT}|$

proof $|T| \leq |T^+| = |M^+|$

tri. ineq.

Metric TSP

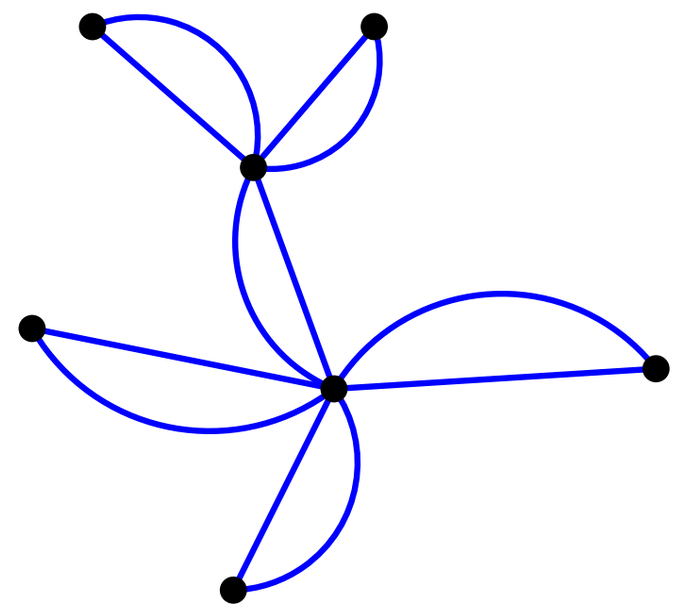
2-approximation algorithm:

(1) build MST M of G (Kruskal)

(2) double edges $\rightarrow M^+$ Eulerian

(3) build greedily a Eulerian tour T^+ on M^+

(4) Trim edges of $T^+ \rightarrow T$



Thm $|T| \leq 2|\mathrm{OPT}|$

proof $|T| \leq |T^+| = |M^+| = 2|M|$

tri. ineq.

Metric TSP

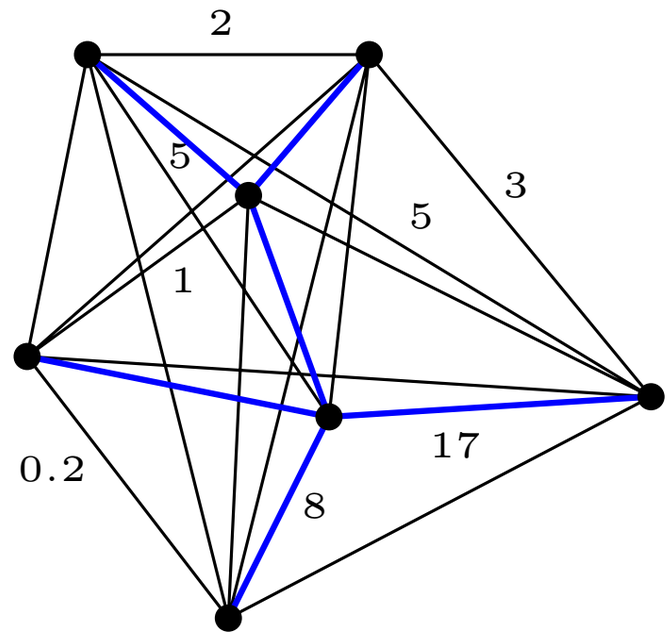
2-approximation algorithm:

(1) build MST M of G (Kruskal)

(2) double edges $\rightarrow M^+$ Eulerian

(3) build greedily a Eulerian tour T^+ on M^+

(4) Trim edges of $T^+ \rightarrow T$



Thm $|T| \leq 2|\mathrm{OPT}|$

proof $|T| \leq |T^+| = |M^+| = 2|M| \leq 2|\mathrm{OPT}|$

tri. ineq. OPT="tree+edge"

Metric TSP

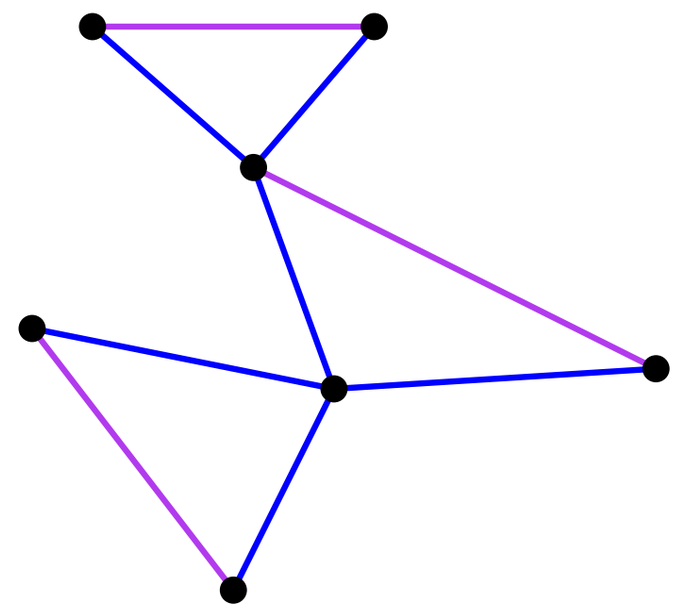
2-approximation algorithm:

(1) build MST M of G (Kruskal)

(2) double edges $\rightarrow M^+$ Eulerian

(3) build greedily a Eulerian tour T^+ on M^+

(4) Trim edges of $T^+ \rightarrow T$



Replace (2) by adding to M a min cost perfect matching of its odd-valenced vertices $\rightarrow \frac{3}{2}$-approximation [Christofides76]

Metric TSP

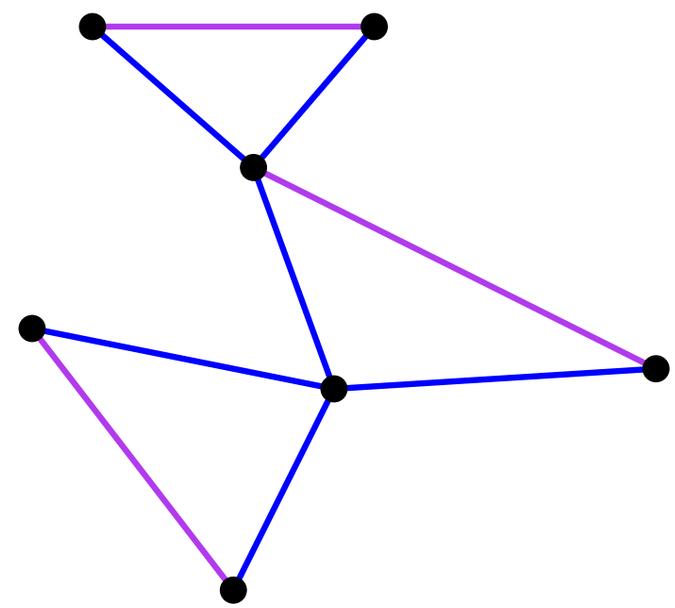
2-approximation algorithm:

(1) build MST M of G (Kruskal)

(2) double edges $\rightarrow M^+$ Eulerian

(3) build greedily a Eulerian tour T^+ on M^+

(4) Trim edges of $T^+ \rightarrow T$



Replace (2) by adding to M a min cost perfect matching of its odd-valenced vertices $\rightarrow \frac{3}{2}$-approximation [Christofides76]

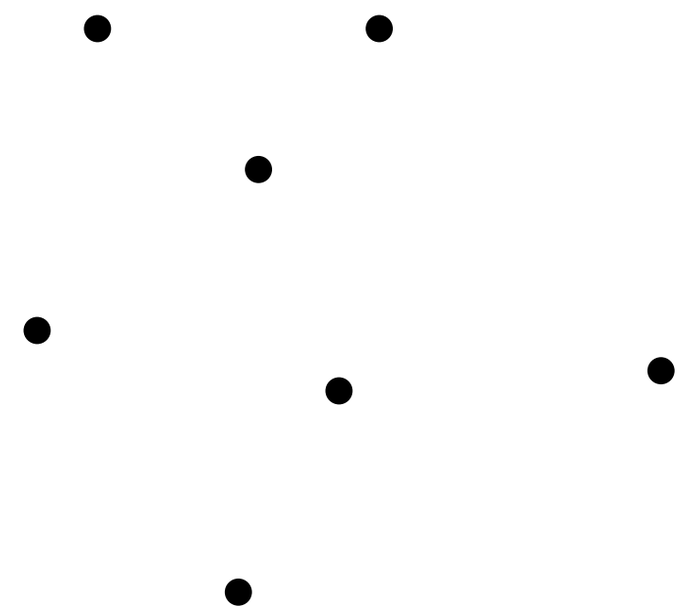
Q Can we do better?

Thm [ALMSS92] There is no PTAS for Metric TSP, unless $P = NP$

Conjecture best approximation factor: $4/3$

Euclidean TSP

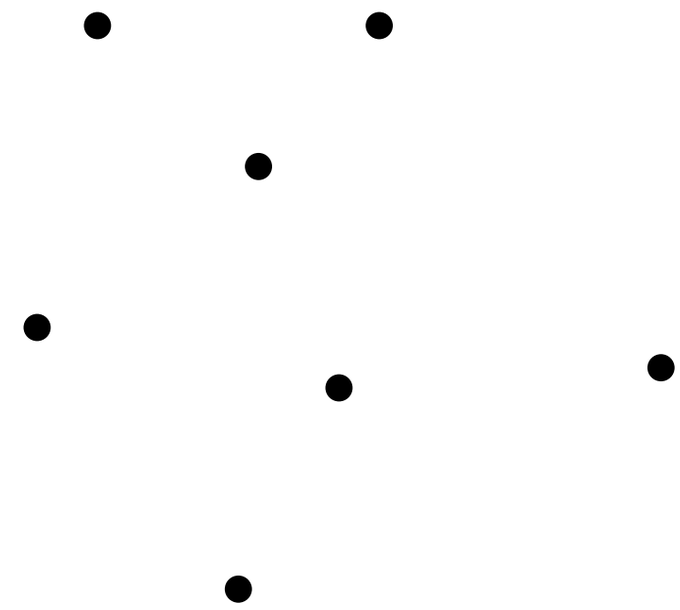
$V \subset \mathbb{R}^d$, E is the set of all pairs weighted by Euclidean distances



Euclidean TSP

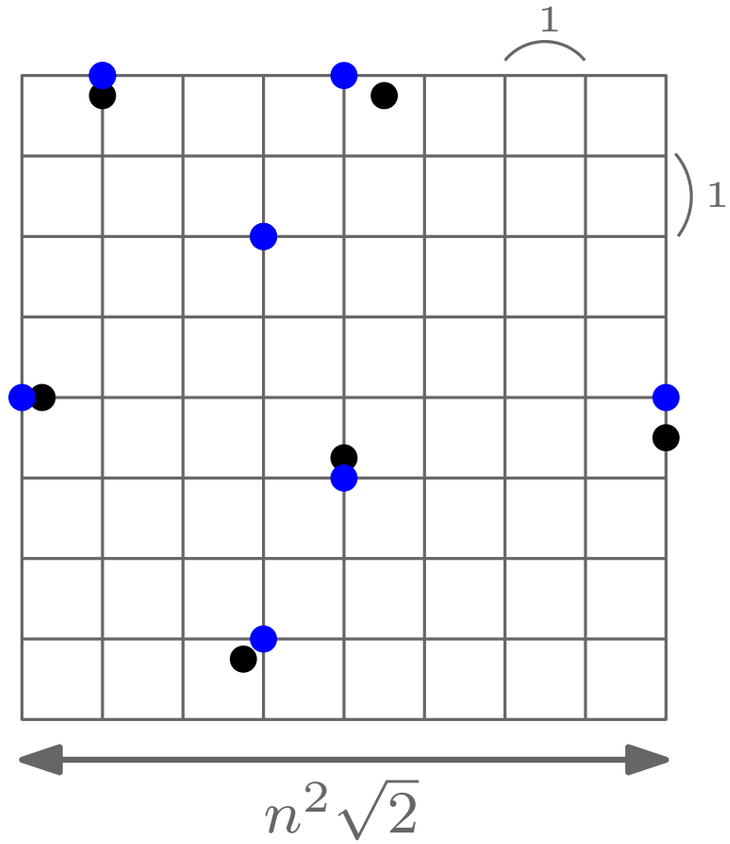
Thm [Arora '96] Euclidean TSP admits a PTAS

Overview Let $n = |V|$



Euclidean TSP

Thm [Arora '96] Euclidean TSP admits a PTAS



Overview Let $n = |V|$

(1) rescale/snap V

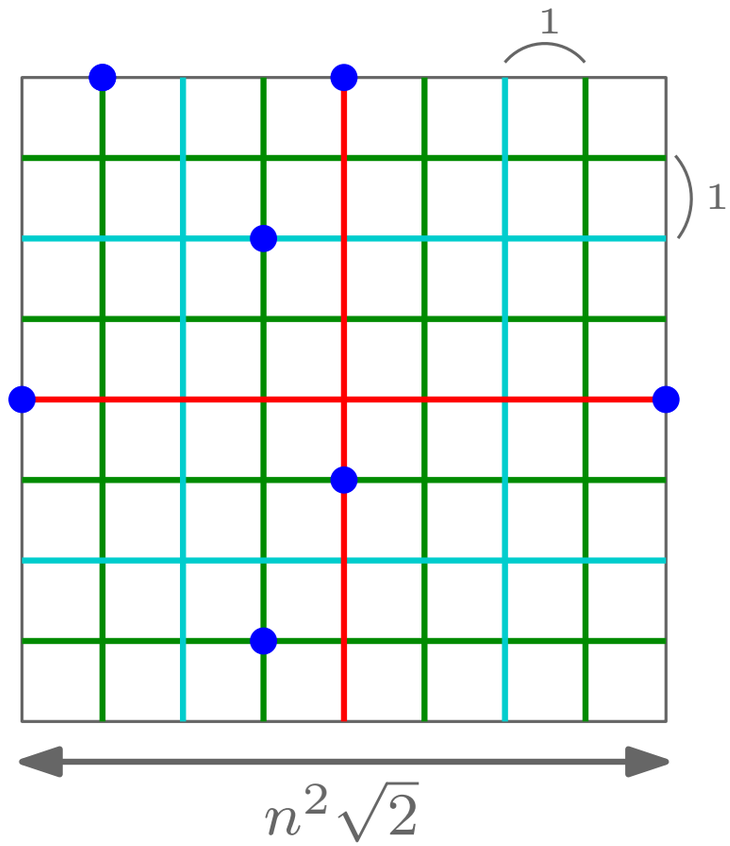
Euclidean TSP

Thm [Arora '96] Euclidean TSP admits a PTAS

Overview Let $n = |V|$

(1) rescale/snap V

(2) subdivide the grid with a quadtree



level 1

level 2

level 3

Euclidean TSP

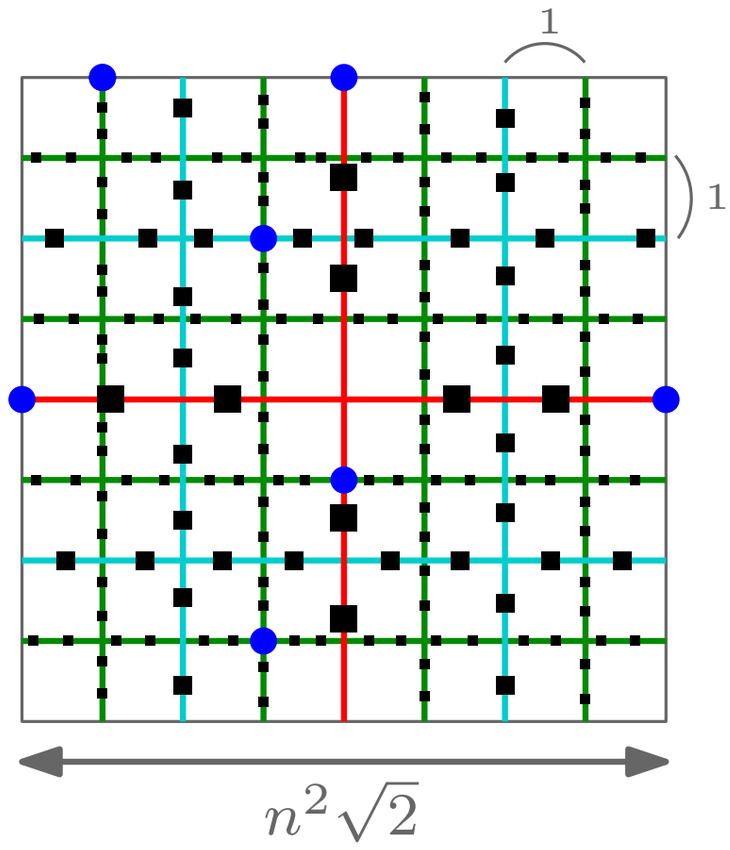
Thm [Arora '96] Euclidean TSP admits a PTAS

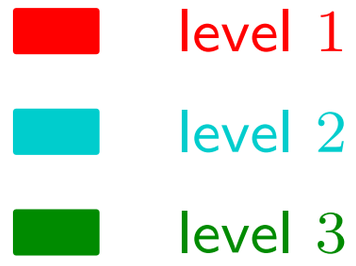
Overview Let $n = |V|$

(1) rescale/snap V

(2) subdivide the grid with a quadtree

(3) place *portals* on grid lines





Euclidean TSP

Thm [Arora '96] Euclidean TSP admits a PTAS

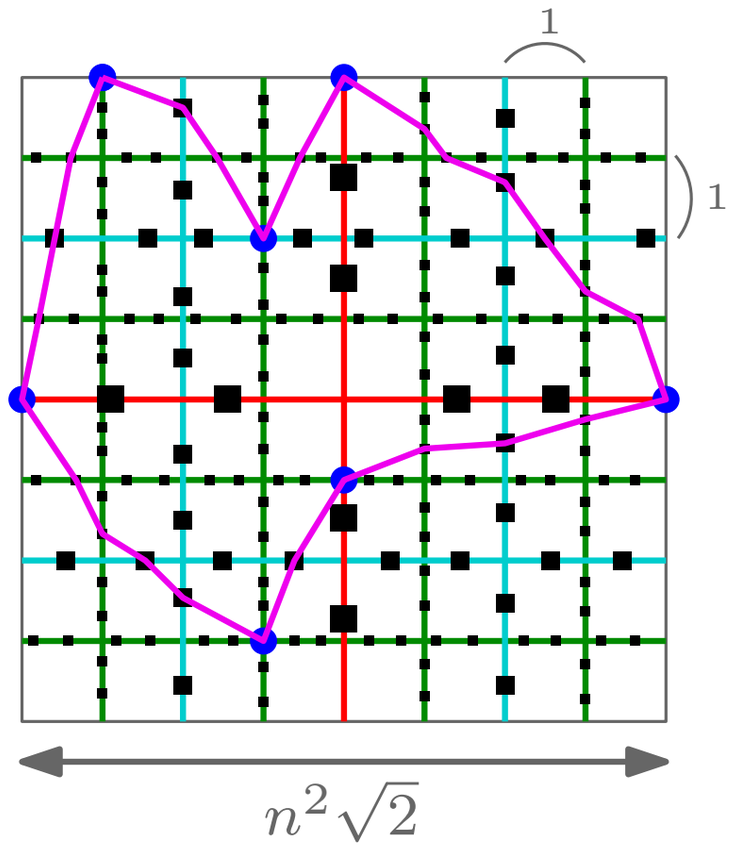
Overview Let $n = |V|$

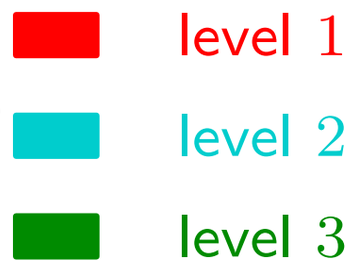
(1) rescale/snap V

(2) subdivide the grid with a quadtree

(3) place *portals* on grid lines

(4) compute the smallest *portal-respecting* tour OPT_p





Euclidean TSP

Thm [Arora '96] Euclidean TSP admits a PTAS

Overview Let $n = |V|$

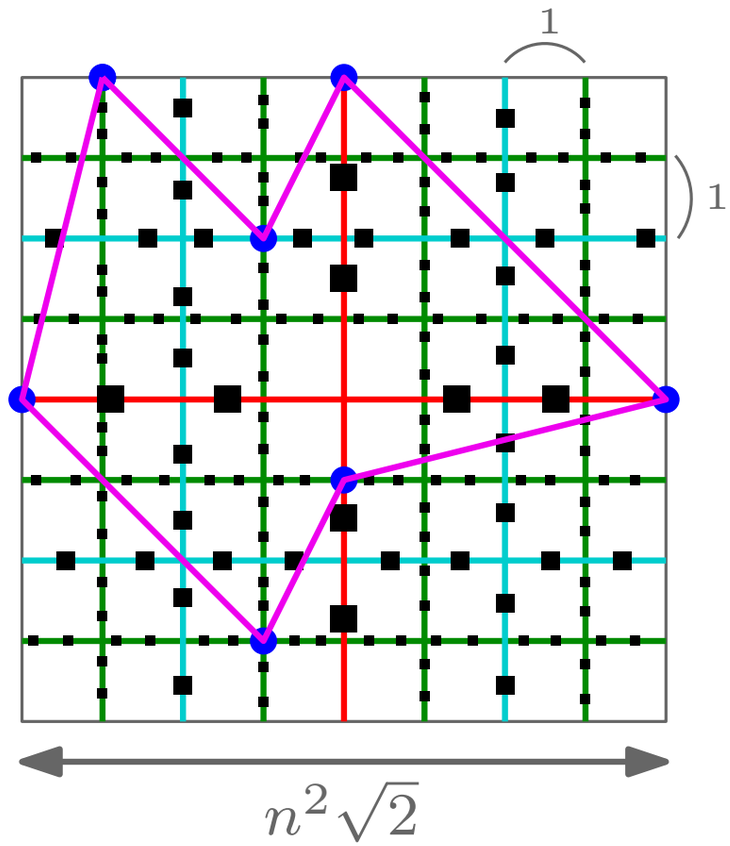
(1) rescale/snap V

(2) subdivide the grid with a quadtree

(3) place *portals* on grid lines

(4) compute the smallest *portal-respecting* tour OPT_p

(5) Trim the edges of OPT_p and output the result T



Euclidean TSP

Thm [Arora '96] Euclidean TSP admits a PTAS

Overview Let $n = |V|$

(1) rescale/snap V

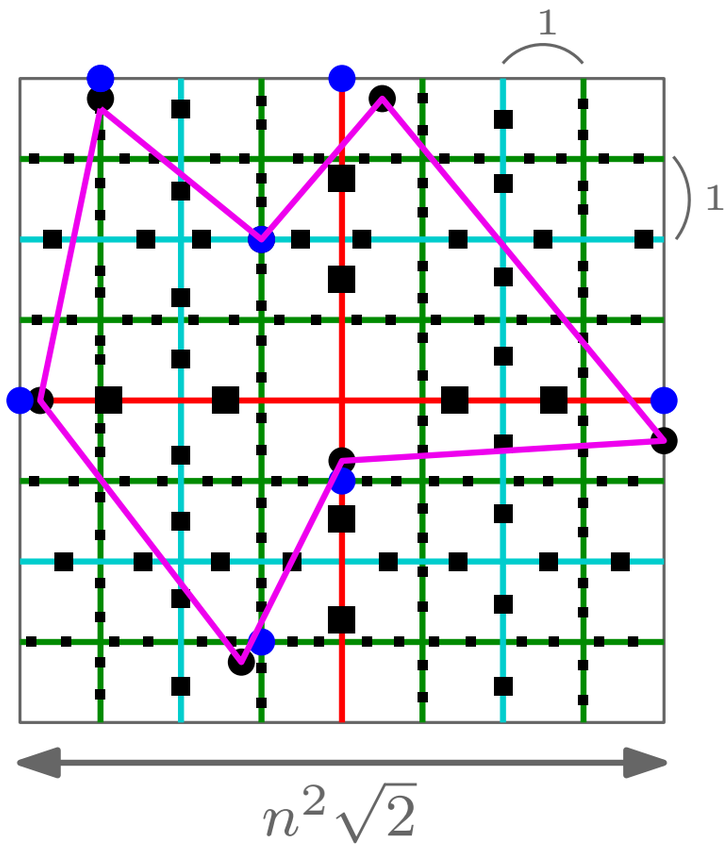
(2) subdivide the grid with a quadtree

(3) place *portals* on grid lines

(4) compute the smallest *portal-respecting* tour OPT_p

(5) Trim the edges of OPT_p and output the result T

(6) Realize the tour T on V and not on the grid vertices



Euclidean TSP

Thm [Arora '96] Euclidean TSP admits a PTAS

Overview Let $n = |V|$

(1) rescale/snap V

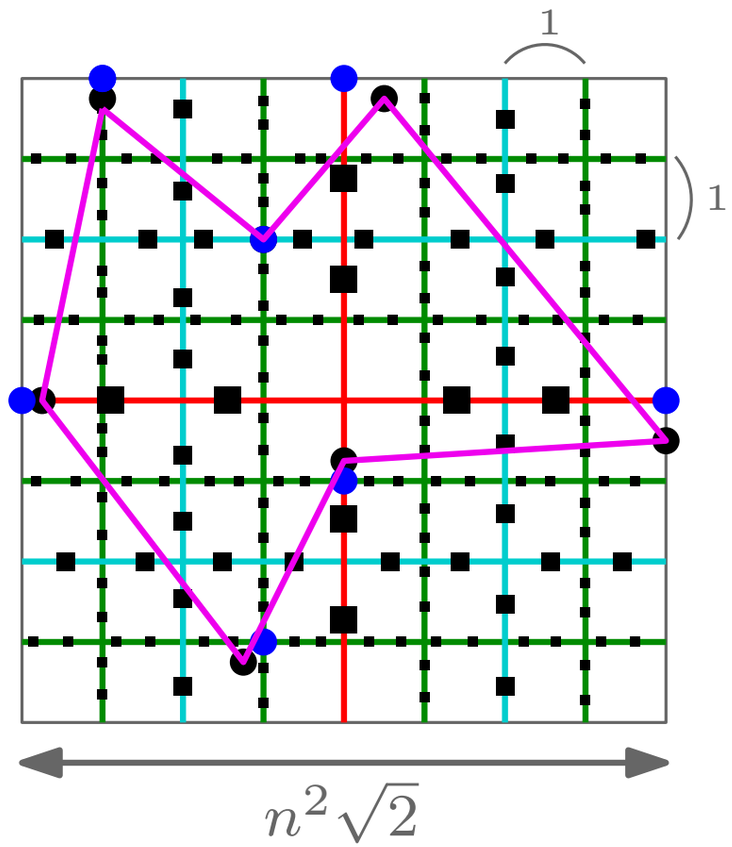
(2) subdivide the grid with a quadtree

(3) place *portals* on grid lines

(4) compute the smallest *portal-respecting* tour OPT_p

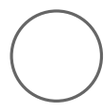
(5) Trim the edges of OPT_p and output the result T

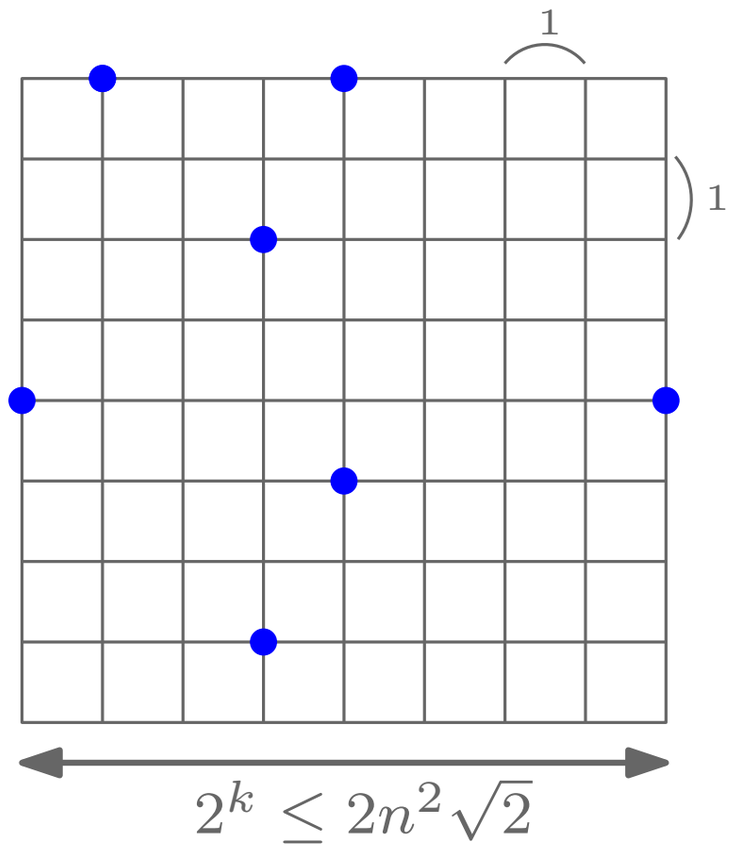
(6) Realize the tour T on V and not on the grid vertices



(2) Grid subdivision

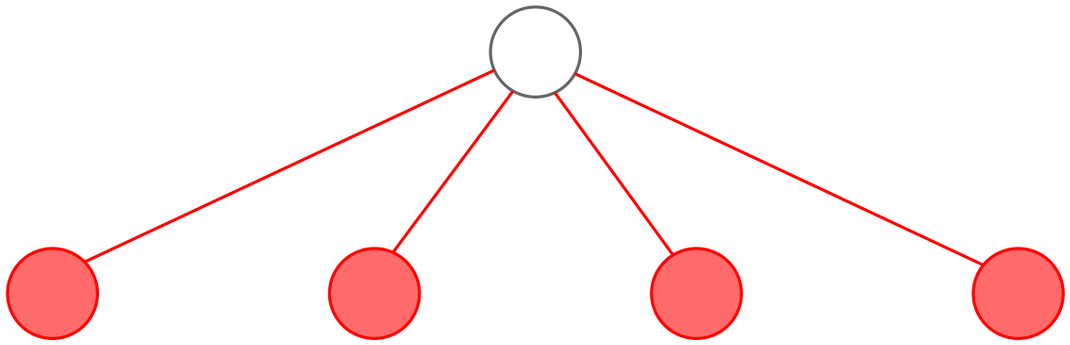
Let k s.t. $2^{k-1} \leq n^2\sqrt{2} \leq 2^k \leq 2n^2\sqrt{2}$

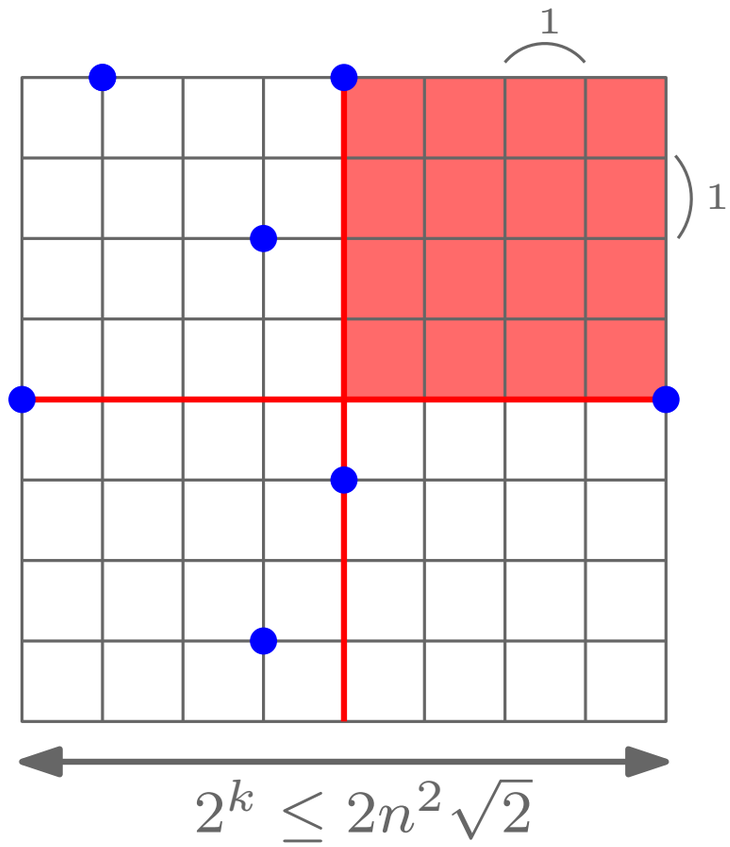




(2) Grid subdivision

Let k s.t. $2^{k-1} \leq n^2\sqrt{2} \leq 2^k \leq 2n^2\sqrt{2}$

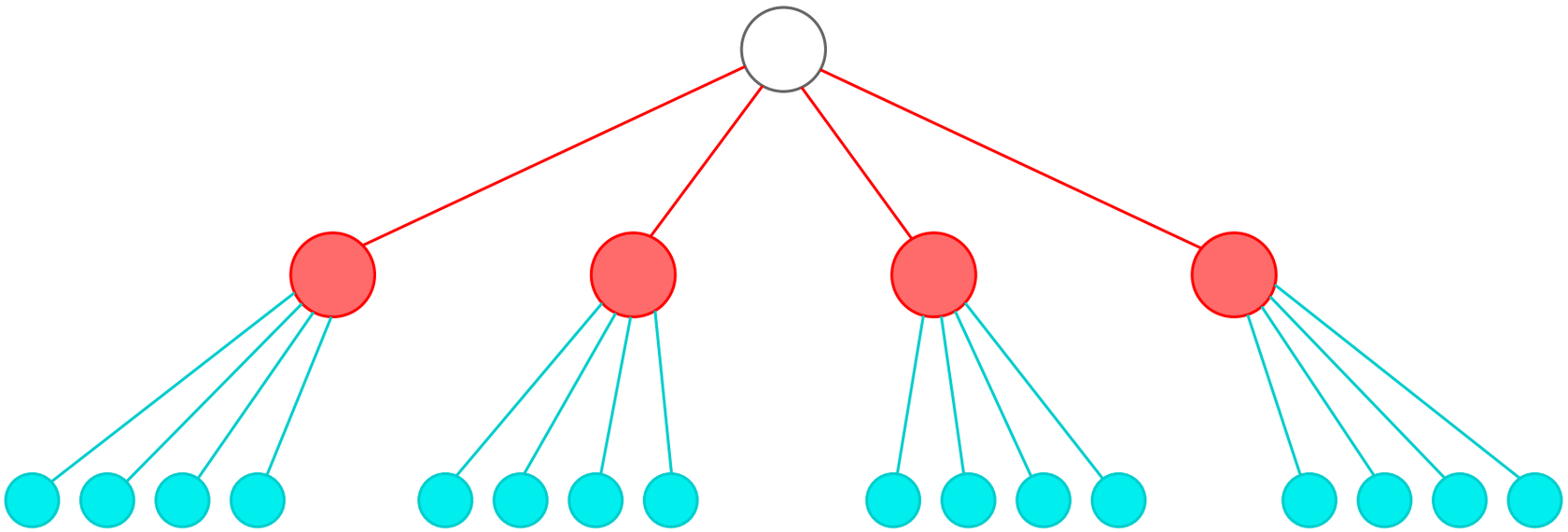


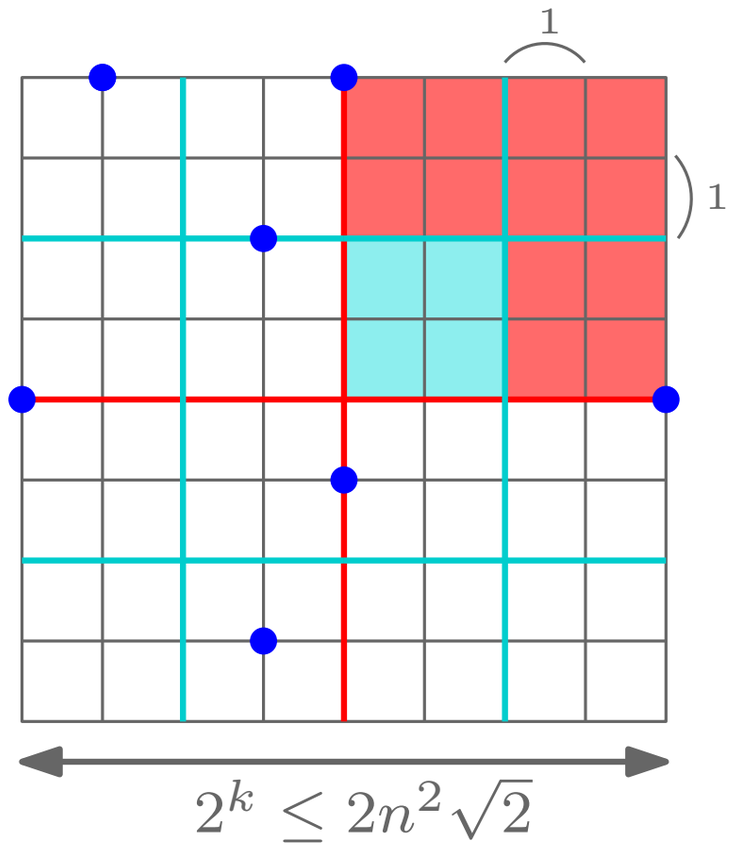


level 1

(2) Grid subdivision

Let k s.t. $2^{k-1} \leq n^2\sqrt{2} \leq 2^k \leq 2n^2\sqrt{2}$



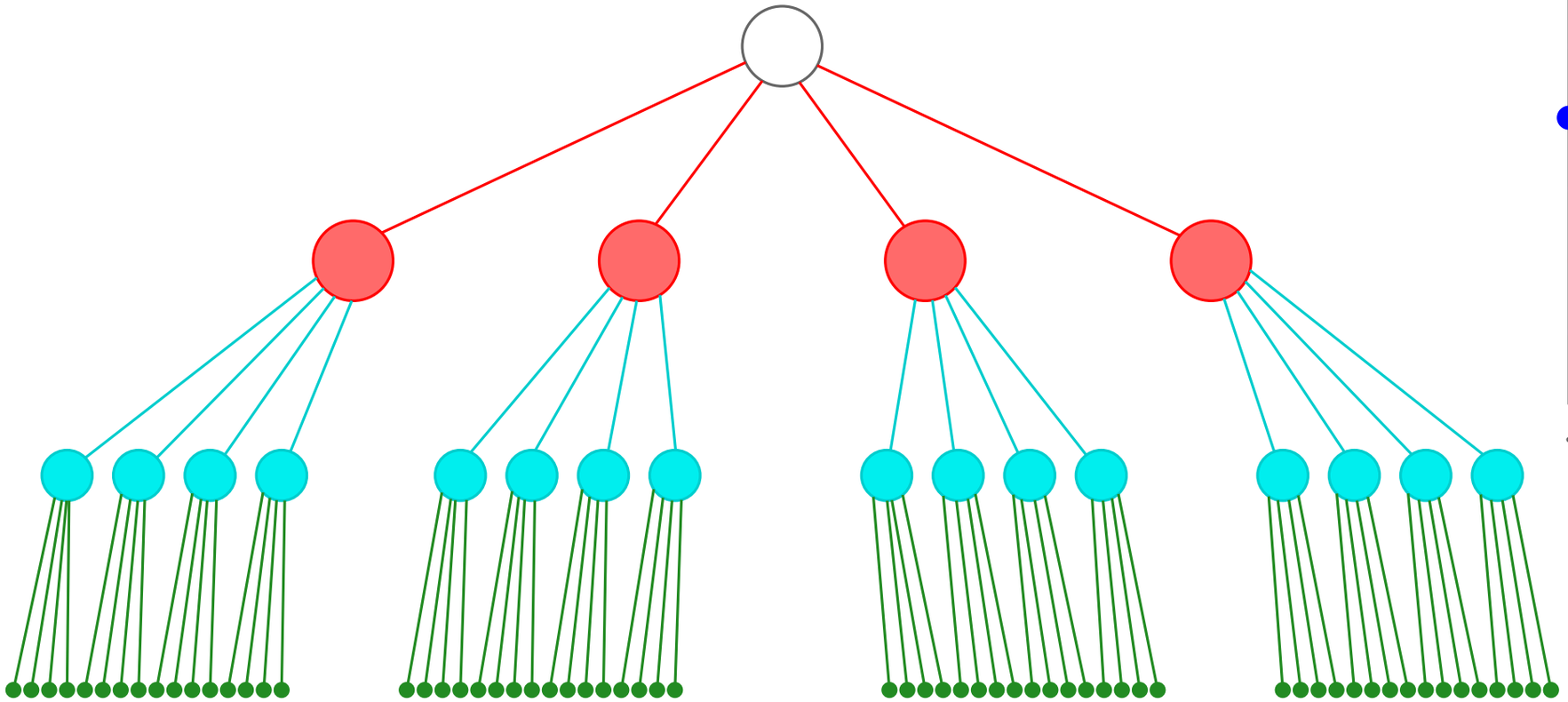


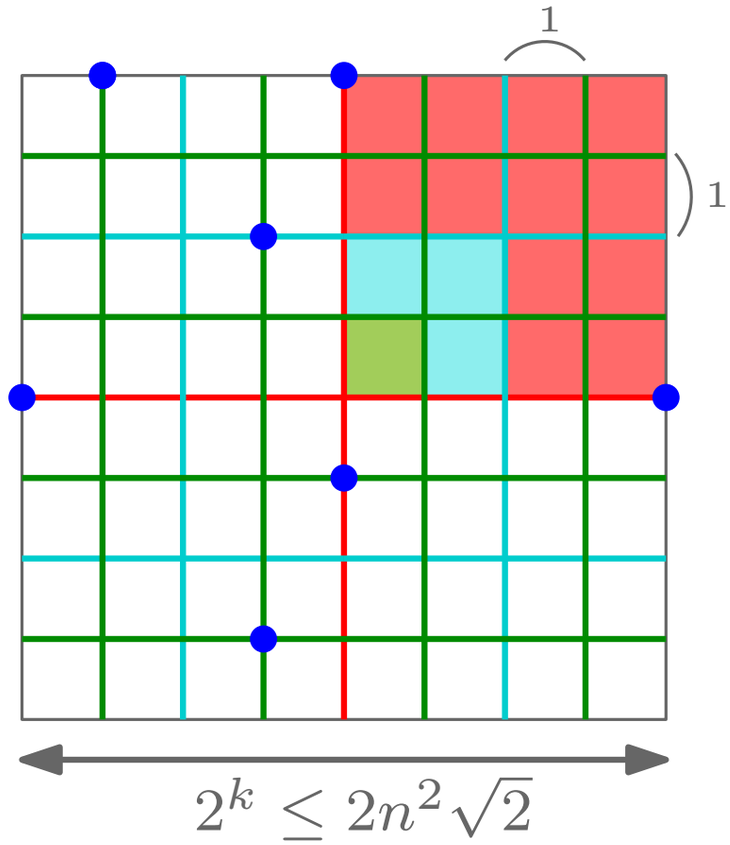
level 1

level 2

(2) Grid subdivision

Let k s.t. $2^{k-1} \leq n^2\sqrt{2} \leq 2^k \leq 2n^2\sqrt{2}$





level 1

level 2

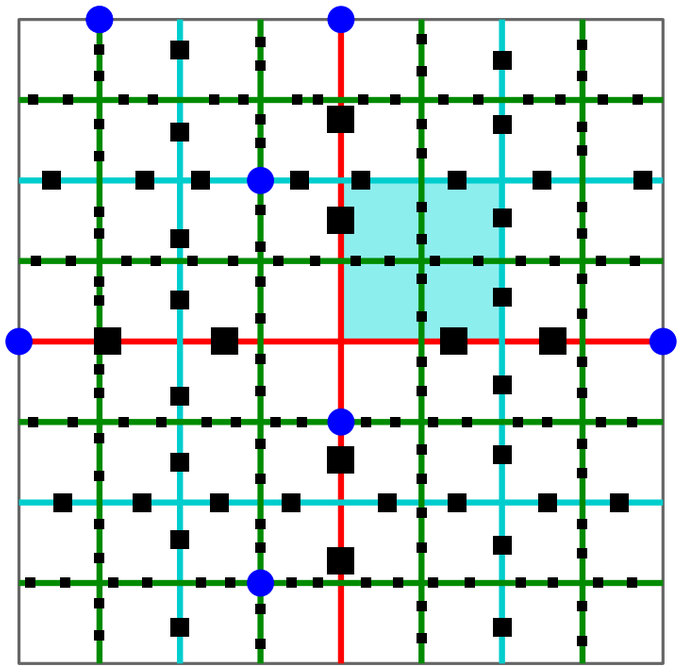
level 3

$O(n^4)$ leaves $\Rightarrow$ size $= O(n^4)$

(3) Portals

Let $m = \left\lfloor \frac{\log n}{\varepsilon} \right\rfloor$

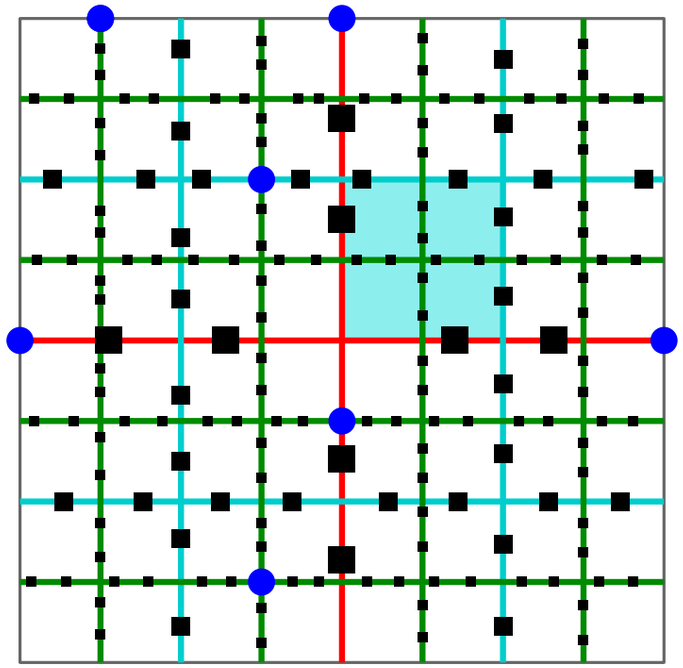
On each level i line, place $2^i m$ equally-spaced portals, plus one at each intersection with other level-i lines



(3) Portals

Let $m = \left\lfloor \frac{\log n}{\varepsilon} \right\rfloor$

On each level i line, place $2^i m$ equally-spaced portals, plus one at each intersection with other level-i lines



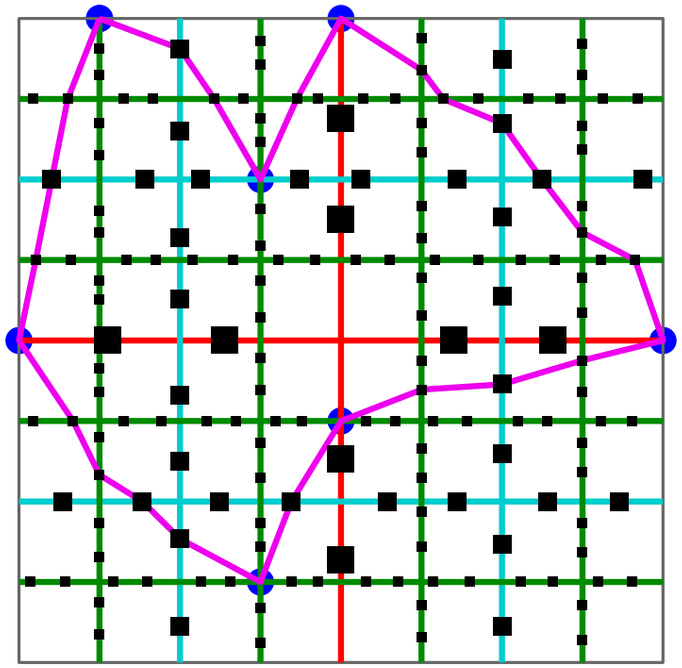
Each level i line is incident to 2^i pairs of level i squares $\Rightarrow m$ portals per pair (w/o corners)

Each level i square has a boundary made of level $j \leq i$ lines

$\Rightarrow$ at most $4m + 4$ portals per square

(4) Portal-respecting tours

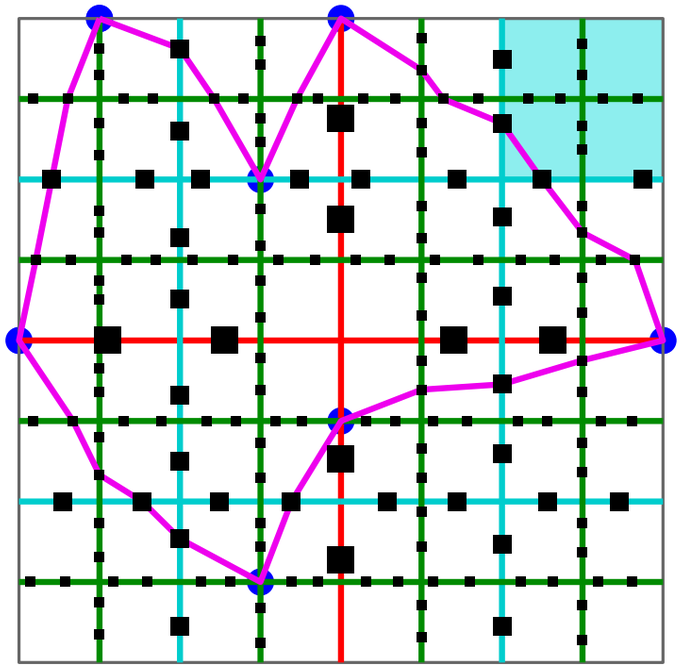
Def A tour is *portal-respecting* if it crosses the grid only at portals



(4) Portal-respecting tours

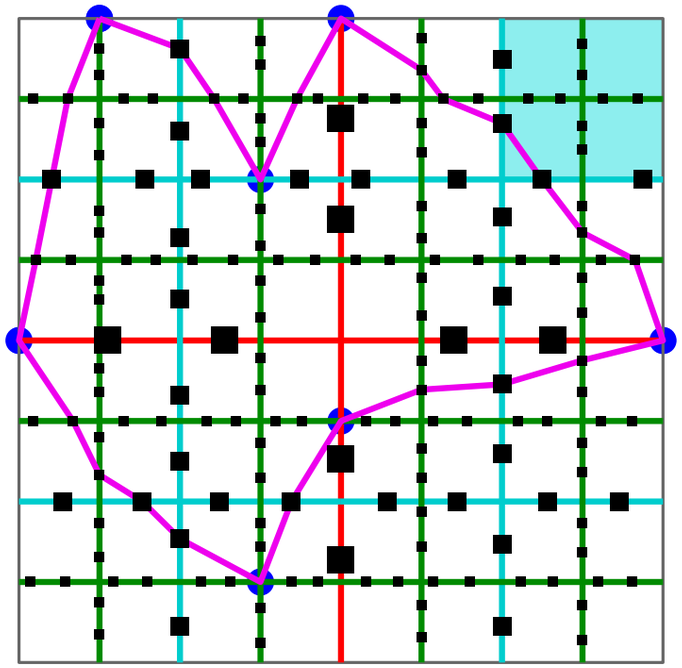
Def A tour is *portal-respecting* if it crosses the grid only at portals

Idea Consider all possible ways a portal-respecting tour can intersect a cell's boundary.



(4) Portal-respecting tours

Def A tour is *portal-respecting* if it crosses the grid only at portals



Idea Consider all possible ways a portal-respecting tour can intersect a cell's boundary.

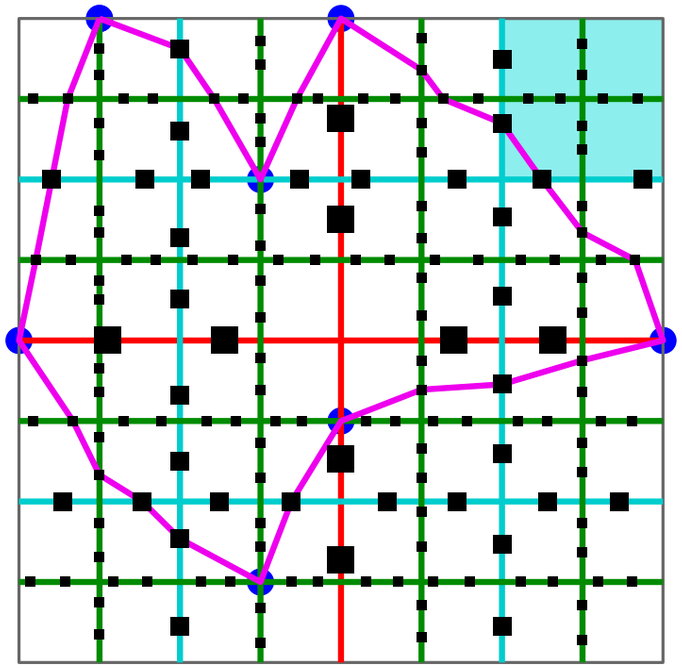
Problems:

- a tour may pass an arbitrary number of times through a portal

- even with a fixed number of crossings, combinatorics is still $\Omega(m!) = \Omega(n^{\log n})$, which is super-polynomial

(4) Portal-respecting tours

Def A tour is *portal-respecting* if it crosses the grid only at portals



Idea Consider all possible ways a portal-respecting tour can intersect a cell's boundary.

Problems:

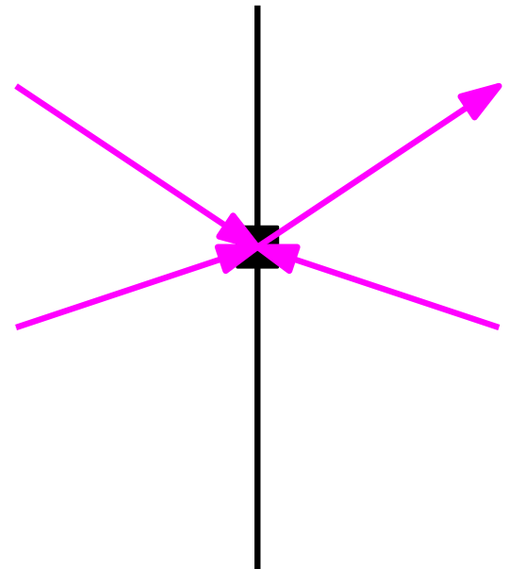
- a tour may pass an arbitrary number of times through a portal

- even with a fixed number of crossings, combinatorics is still $\Omega(m!) = \Omega(n^{\log n})$, which is super-polynomial

Goal: exploit properties of OPT_p to reduce combinatorics to polynomial.

(4) Portal-respecting tours

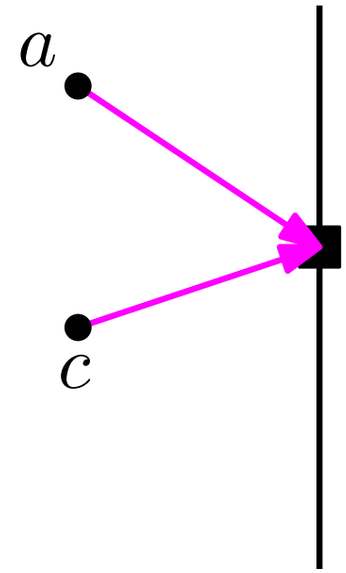
Def A tour is *portal-respecting* if it crosses the grid only at portals



Def a tour is k-*light* if each portal is visited at most k times

(4) Portal-respecting tours

Def A tour is *portal-respecting* if it crosses the grid only at portals



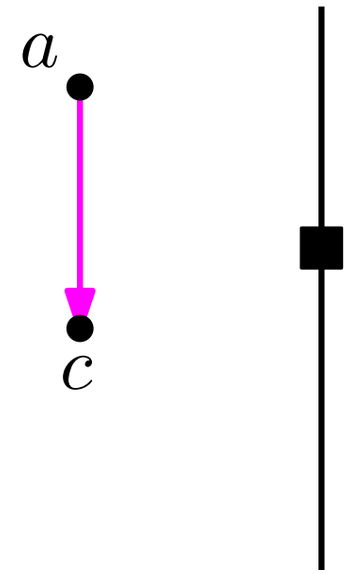
Def a tour is k-*light* if each portal is visited at most k times

Prop OPT_p is 2-light

(4) Portal-respecting tours

Def A tour is *portal-respecting* if it crosses the grid only at portals

Def a tour is k-*light* if each portal is visited at most k times



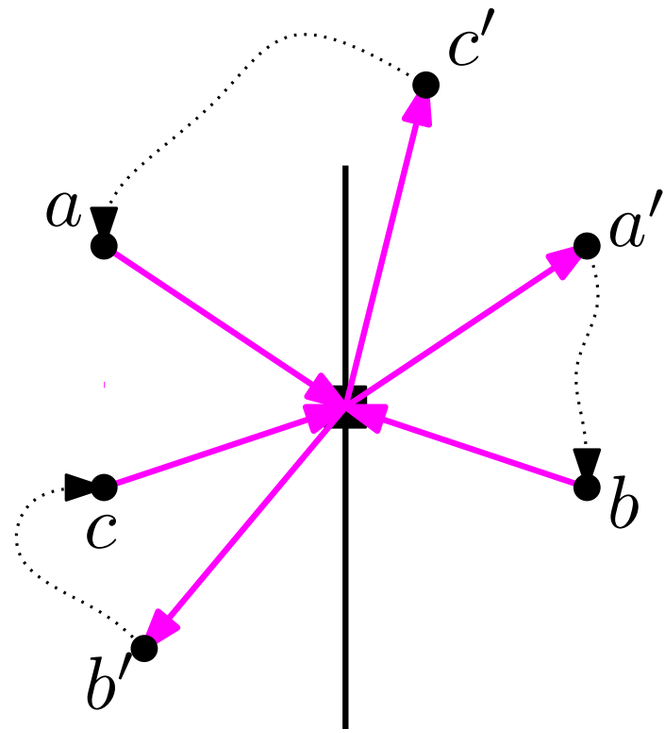
Prop OPT_p is 2-light

(4) Portal-respecting tours

Def A tour is *portal-respecting* if it crosses the grid only at portals

Def a tour is k-*light* if each portal is visited at most k times

Prop OPT_p is 2-light



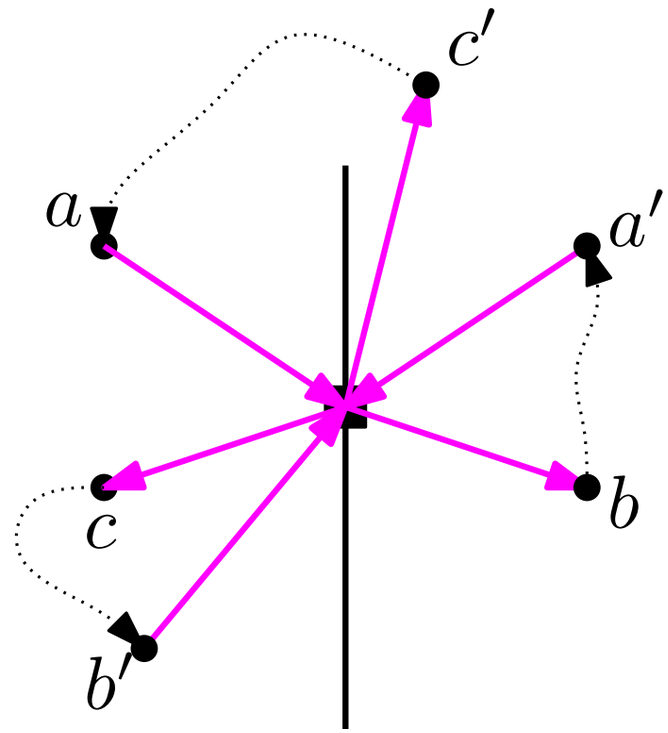
(a, a', b, b', c, c')

(4) Portal-respecting tours

Def A tour is *portal-respecting* if it crosses the grid only at portals

Def a tour is k-*light* if each portal is visited at most k times

Prop OPT_p is 2-light



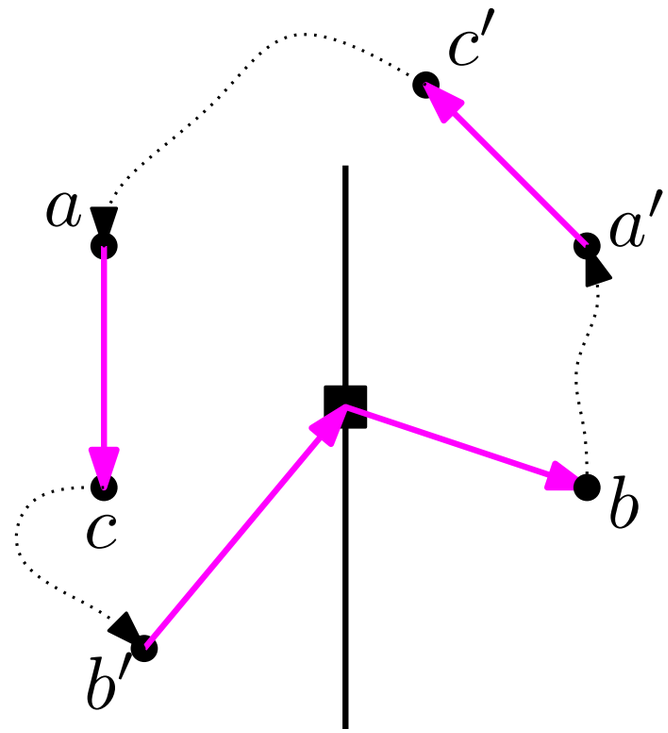
(a, c, b', b, a', c')

(4) Portal-respecting tours

Def A tour is *portal-respecting* if it crosses the grid only at portals

Def a tour is k-*light* if each portal is visited at most k times

Prop OPT_p is 2-light

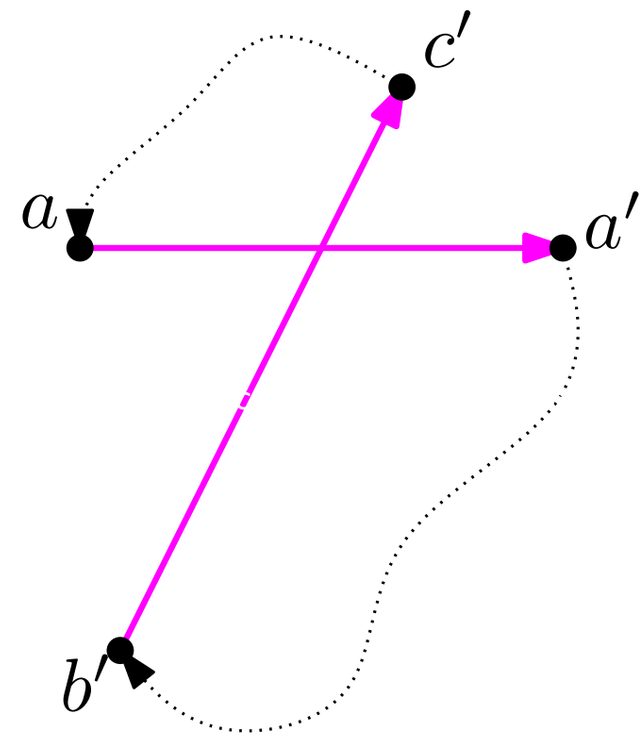


(a, c, b', b, a', c')

(4) Portal-respecting tours

Def A tour is *portal-respecting* if it crosses the grid only at portals

Def a tour is k-*light* if each portal is visited at most k times

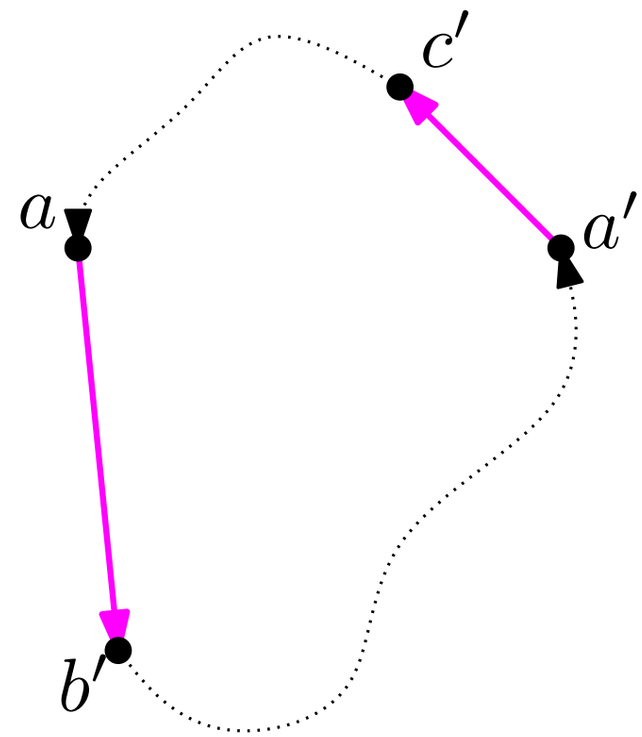


Prop OPT_p is 2-light

Prop OPT_p does not self-intersect, except at portals

(a, a', b', c')

(4) Portal-respecting tours



Def A tour is *portal-respecting* if it crosses the grid only at portals

Def a tour is k-*light* if each portal is visited at most k times

Prop OPT_p is 2-light

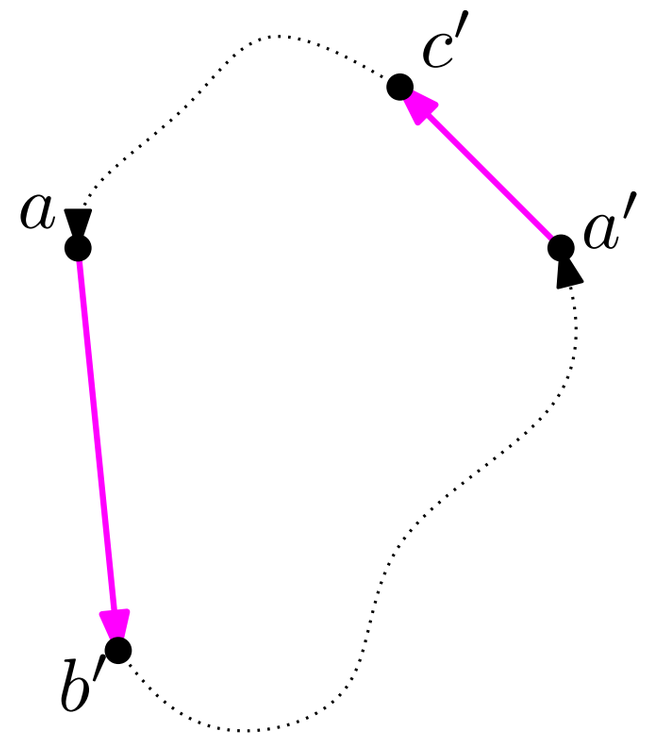
(a, b', a', c')

Prop OPT_p does not self-intersect, except at portals

(4) Portal-respecting tours

Def A tour is *portal-respecting* if it crosses the grid only at portals

Def a tour is k-*light* if each portal is visited at most k times



Prop OPT_p is 2-light

(a, b', a', c')

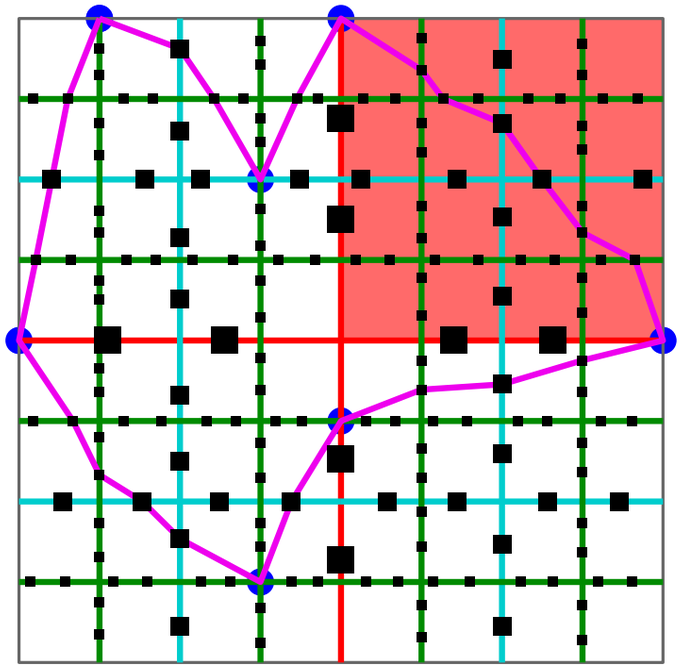
Prop OPT_p does not self-intersect, except at portals

$\Rightarrow$ valid pairings of portals along cell boundary map injectively to balanced arrangements of parentheses

(4) Portal-respecting tours

Goal: find shortest tour that is:

- portal-respecting
- 2-light
- non self-intersecting (except at portals)

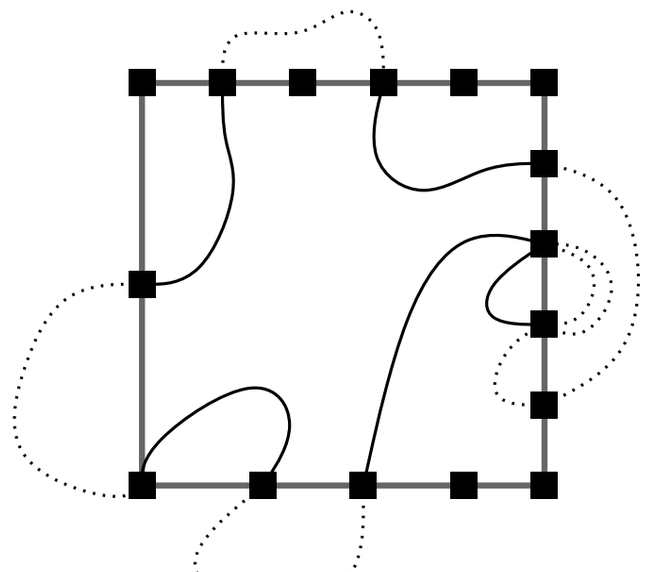


$\rightarrow$ divide-and-conquer approach using quadtree:

(4) Portal-respecting tours

Goal: find shortest tour that is:

- portal-respecting
- 2-light
- non self-intersecting (except at portals)



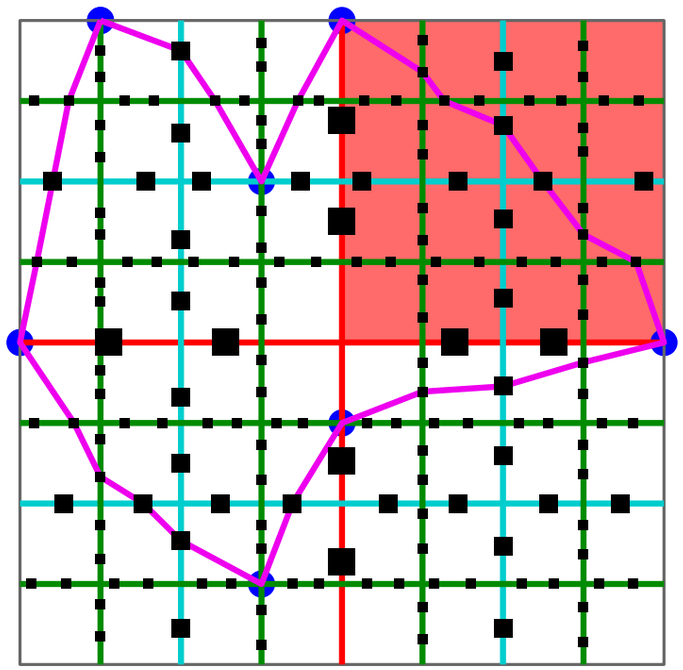
$\rightarrow$ divide-and-conquer approach using quadtree:

- for every cell s, *interface* is defined by:
  - a number of passes through each portal of s $\leftarrow$ $3^{O(m)} = n^{O(1/\varepsilon)}$
  - a paring between selected portals $\leftarrow$ $O(C_m) = O\left(2^{2m}\right) = n^{O(1/\varepsilon)}$

(4) Portal-respecting tours

Goal: find shortest tour that is:

- portal-respecting
- 2-light
- non self-intersecting (except at portals)



$\rightarrow$ divide-and-conquer approach using quadtree:

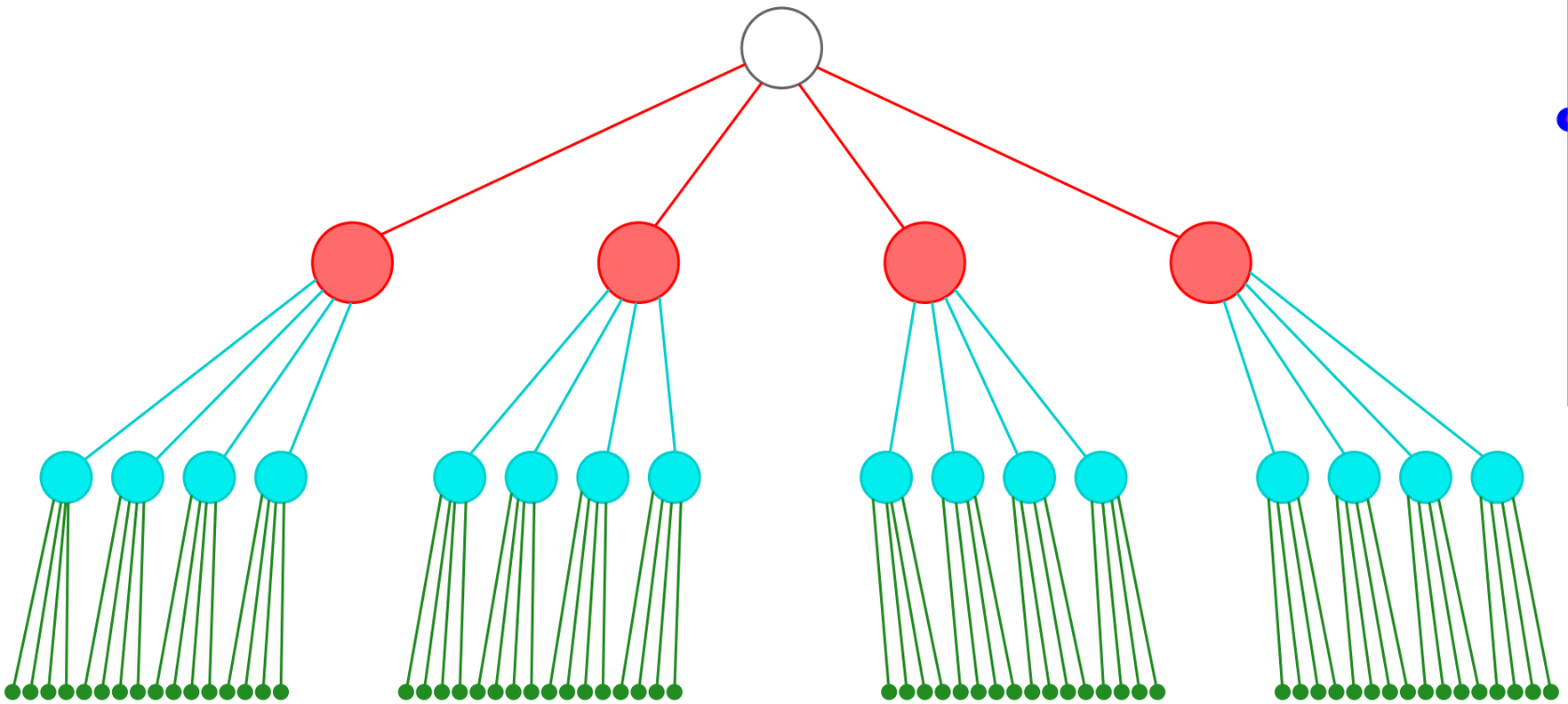
• for every cell s, *interface* is defined by:

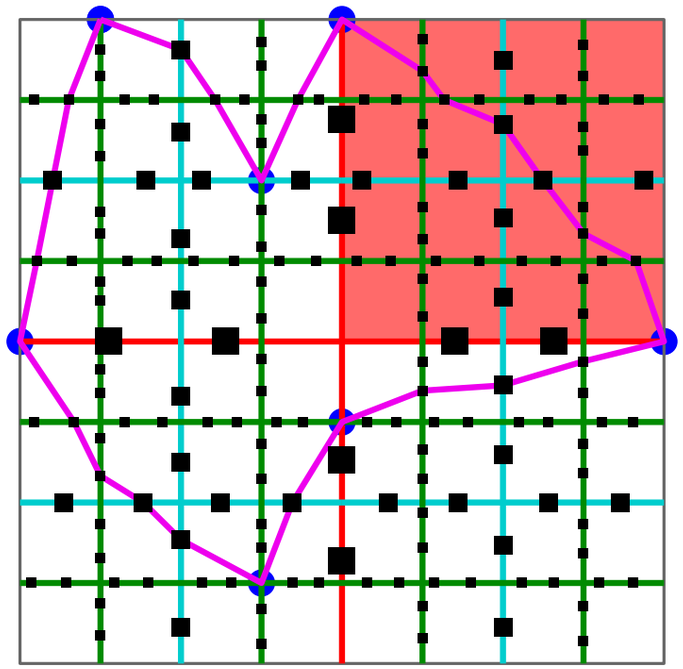
- a number of passes through each portal of s $\quad 3^{O(m)} = n^{O(1/\varepsilon)}$
- a paring between selected portals $\quad O(C_m) = O\left(2^{2m}\right) = n^{O(1/\varepsilon)}$

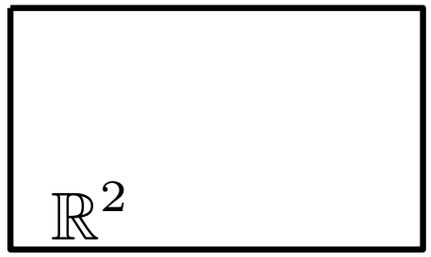
• dynamic programming: given the interface of a level-i cell, combine results of its level-(i+1) children for all their possible compatible interfaces

(4) Portal-respecting tours

Lookup table:

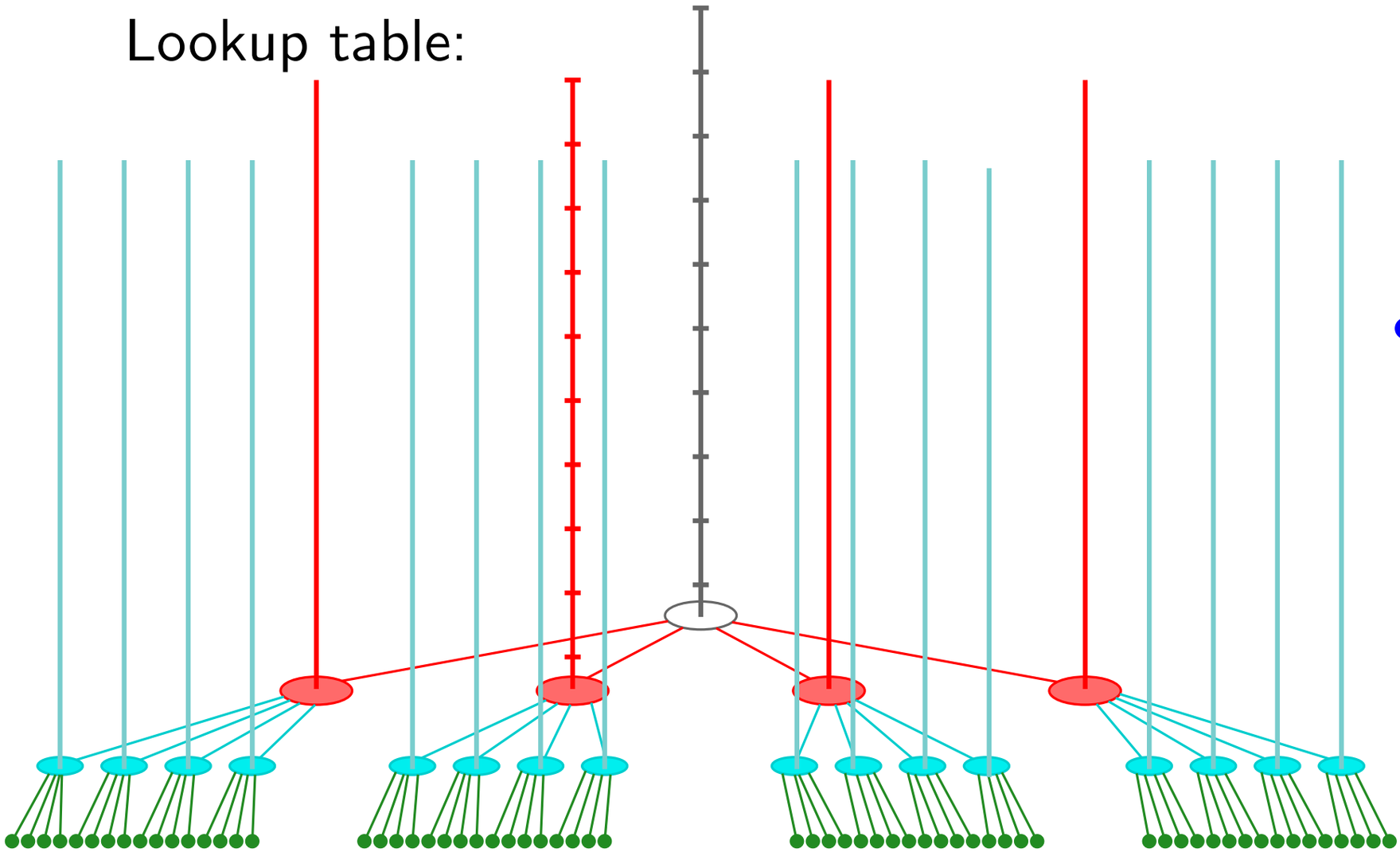


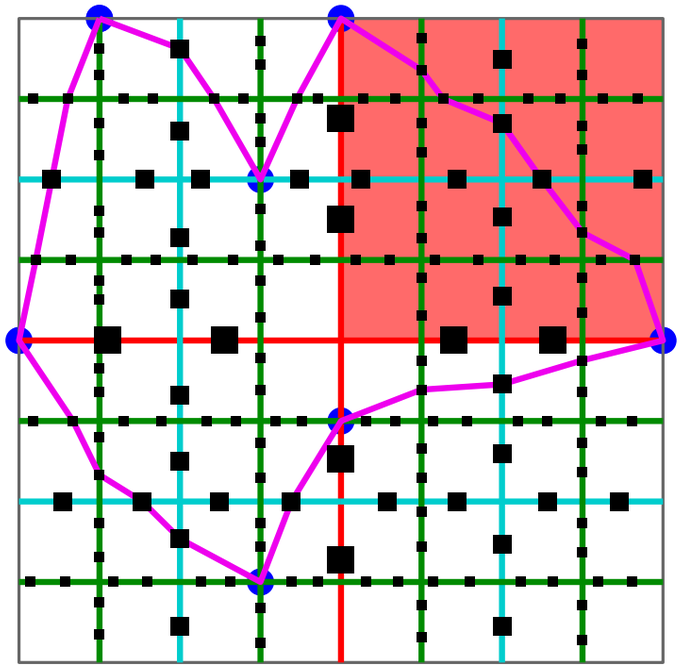




(4) Portal-respecting tours

Lookup table:

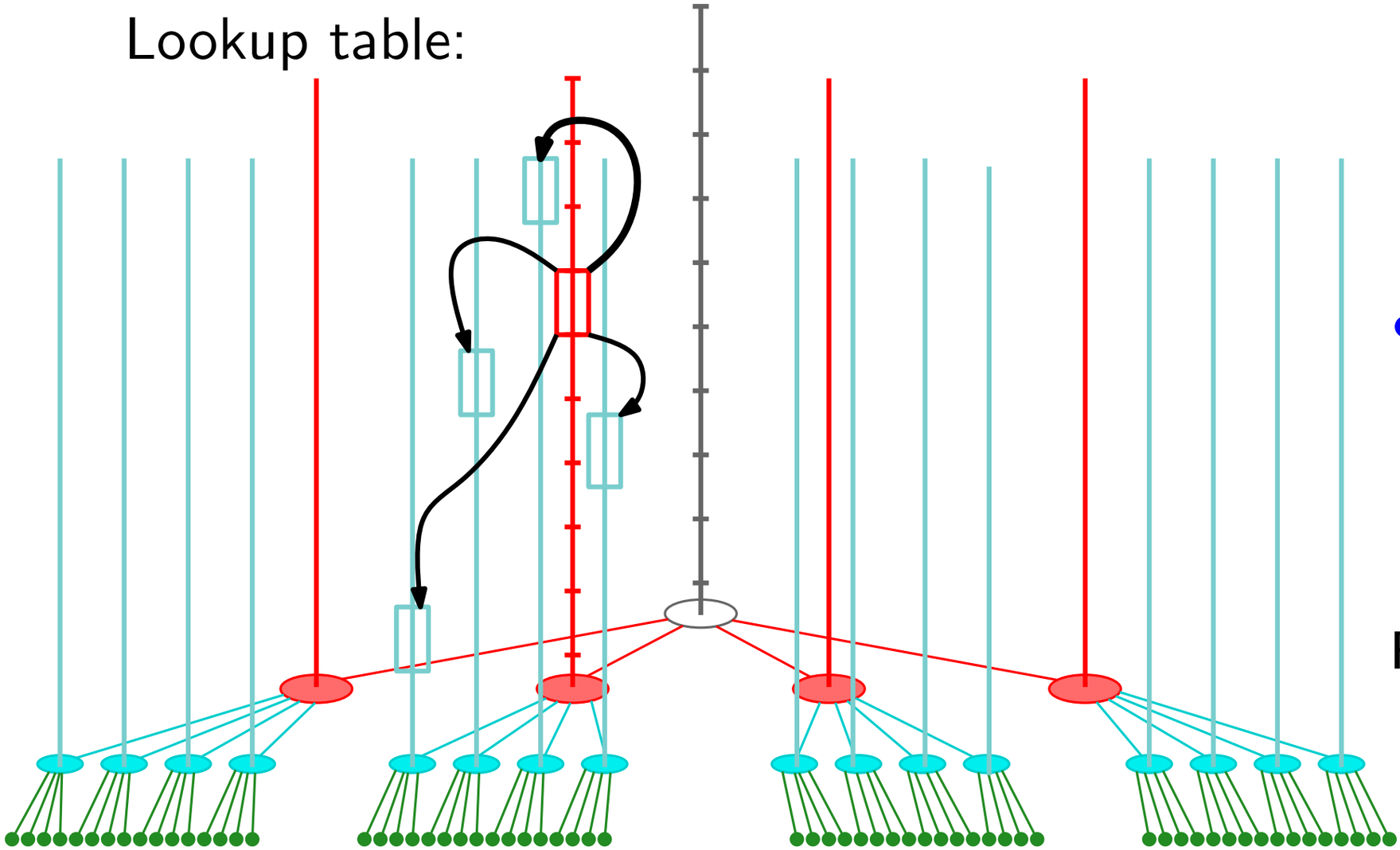


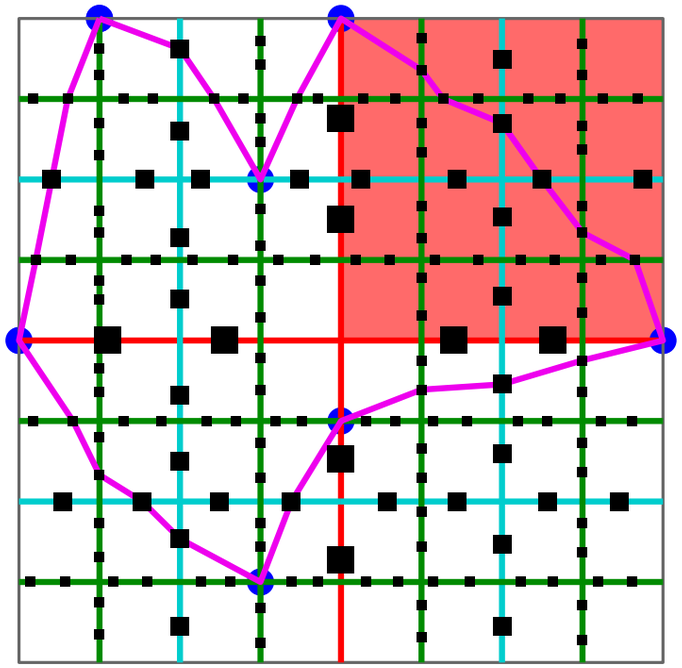


size: $O\left(n^4\ n^{O(1/\varepsilon)}\right)$

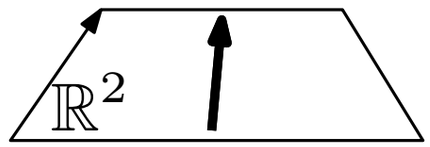
(4) Portal-respecting tours

Lookup table:





Fill the table "in depth"



$\forall$ (leaf,interface),

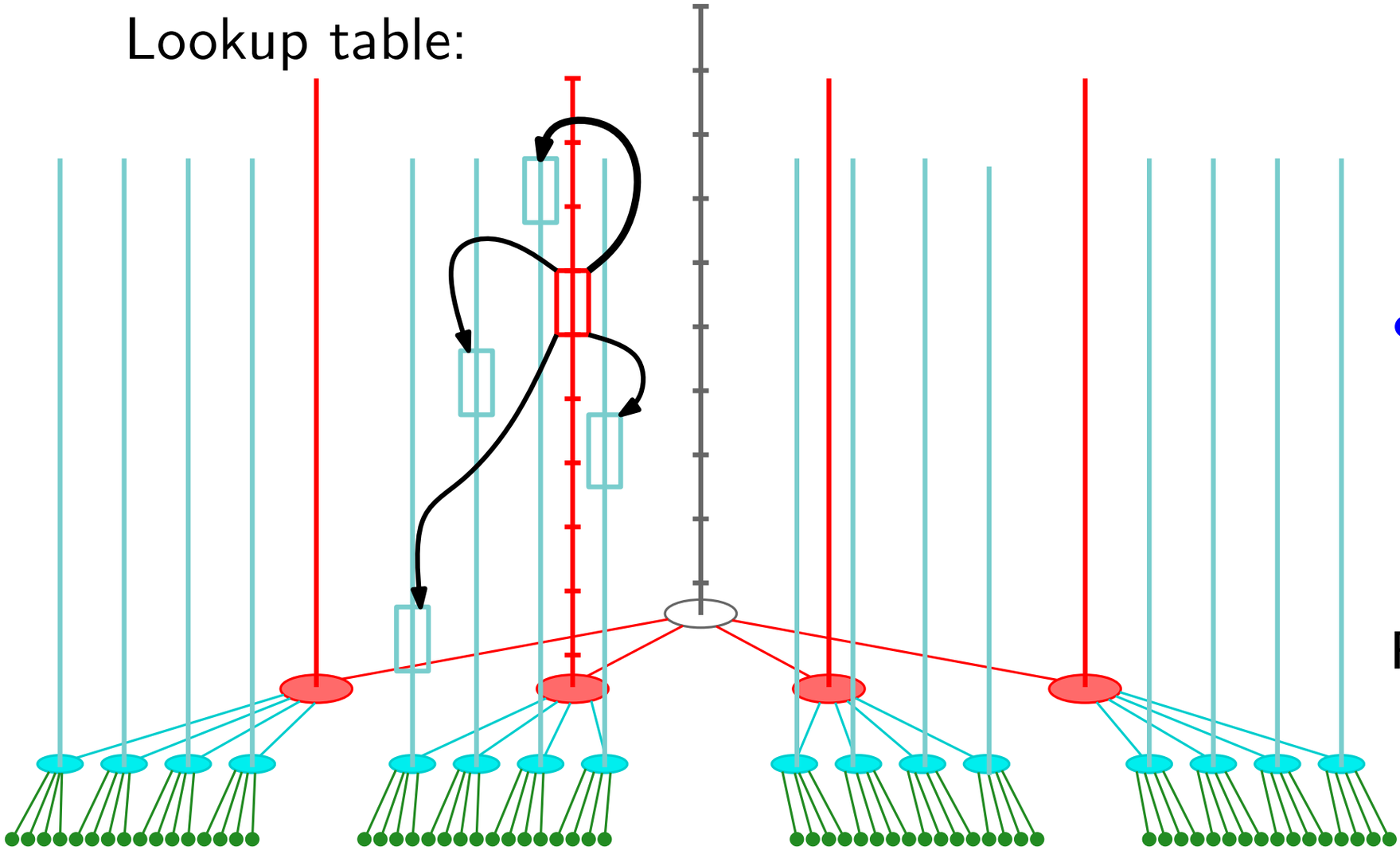
report total length of pairing w/ straight-line segments (nodes are portals) ← $O(1)$

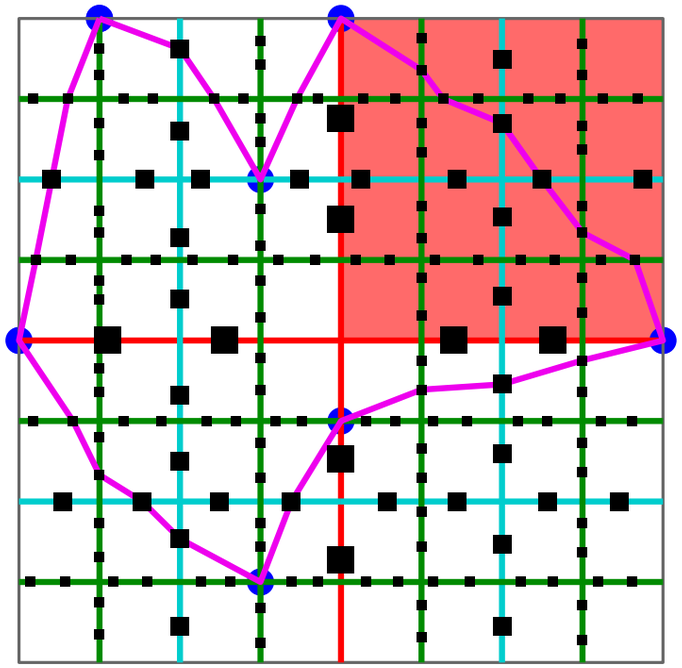
$\forall$ (node, interface),

- select interface for every son ← $n^{O(1/\varepsilon)}$
- retrieve best tour for each selected (son, interface) ← $O(1)$

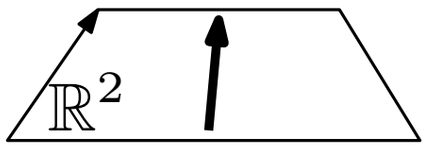
(4) Portal-respecting tours

Lookup table:





Fill the table "in depth"



total running time: $O\left(n^4\ n^{O(1/\varepsilon)}\right)$

Output is the shortest tour that is portal-respecting
(and 2-light and non self-intersecting)

Euclidean TSP

Thm [Arora96] Euclidean TSP admits a PTAS

Overview Let $n = |V|$

(1) rescale/snap V

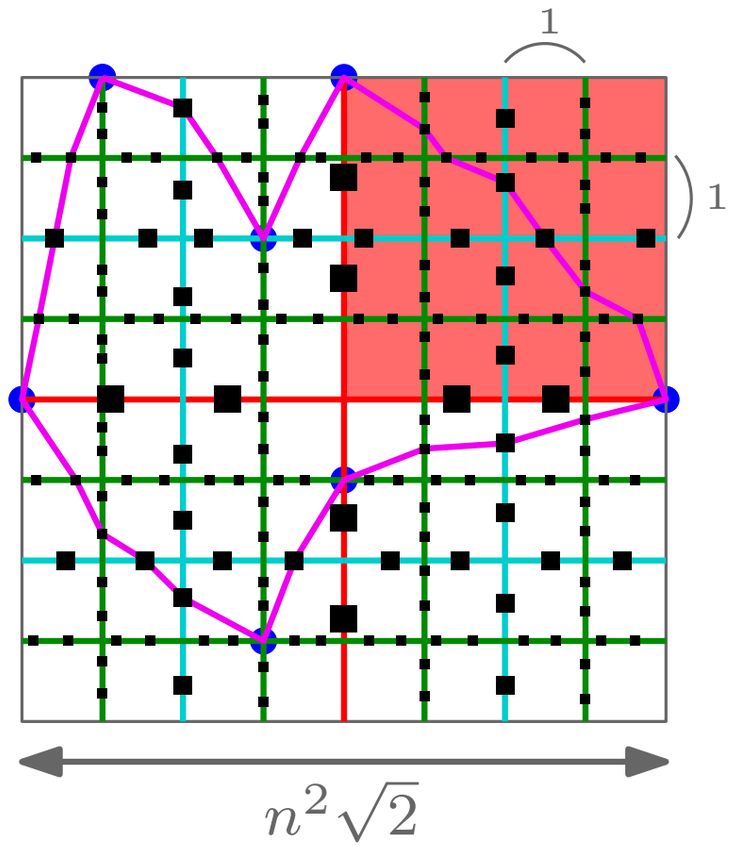
(2) subdivide the gri tree

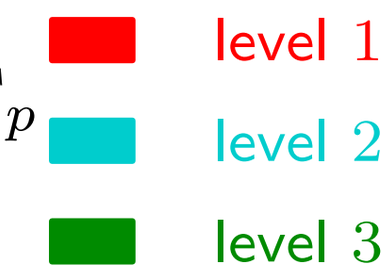
(3) place *port* s

(4) com est *portal-respecting* tour OPT_p

(5) ges of OPT_p and output the result T

(e the tour T on V and not on the grid vertices





Euclidean TSP

Thm [Arora96] Euclidean TSP admits a PT

Overview Let $n = |V|$

(1) rescale/snap V

(2) subdivide the grid wi adtree

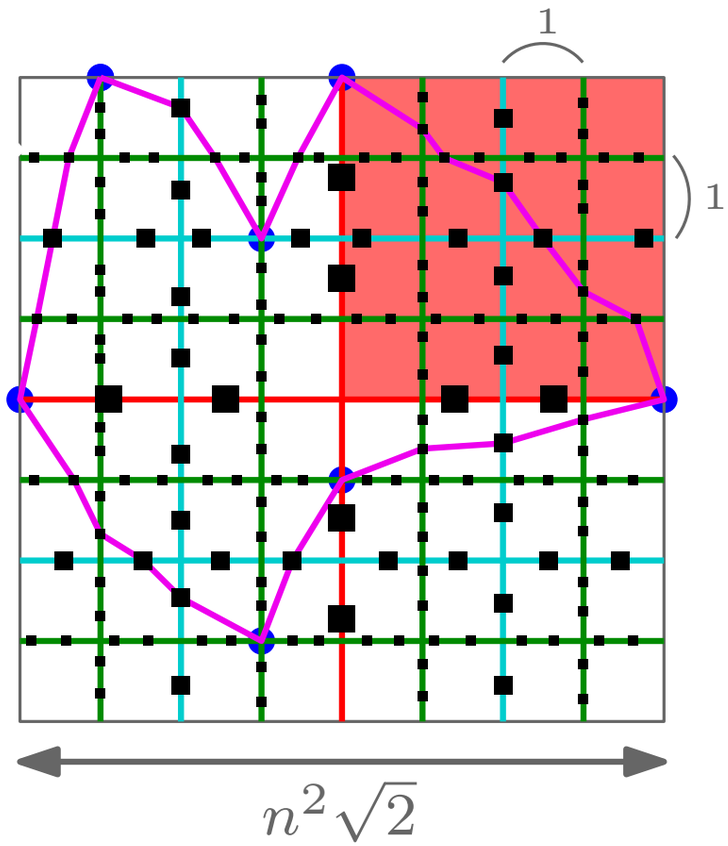
(3) place *portals* o nes

(4) compute allest *portal-respecting* tour OPT_p

(5) Tri dges of OPT_p and output the result T

(ize the tour T on V and not on the grid vertices

Output quality? Do we have $|T| - |\mathrm{OPT}| \leq O(\varepsilon)\,|\mathrm{OPT}|$?



Euclidean TSP

Thm [Arora96] Euclidean TSP admits a PT

Overview Let $n = |V|$

(1) rescale/snap V

(2) subdivide the grid wi adtree

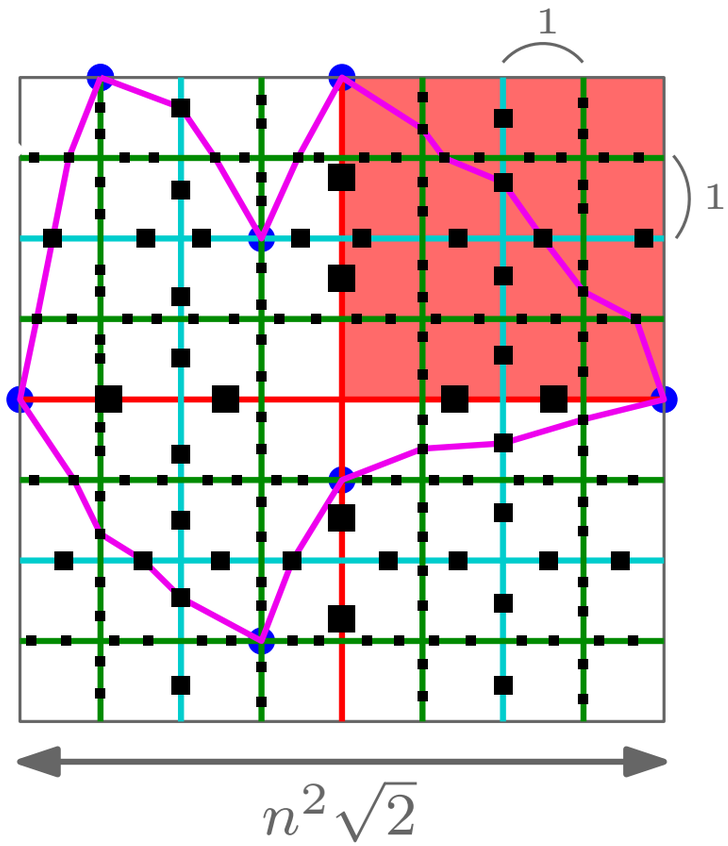
(3) place *portals* o nes

(4) compute allest *portal-respecting* tour OPT_p

(5) Tri dges of OPT_p and output the result T

(ize the tour T on V and not on the grid vertices

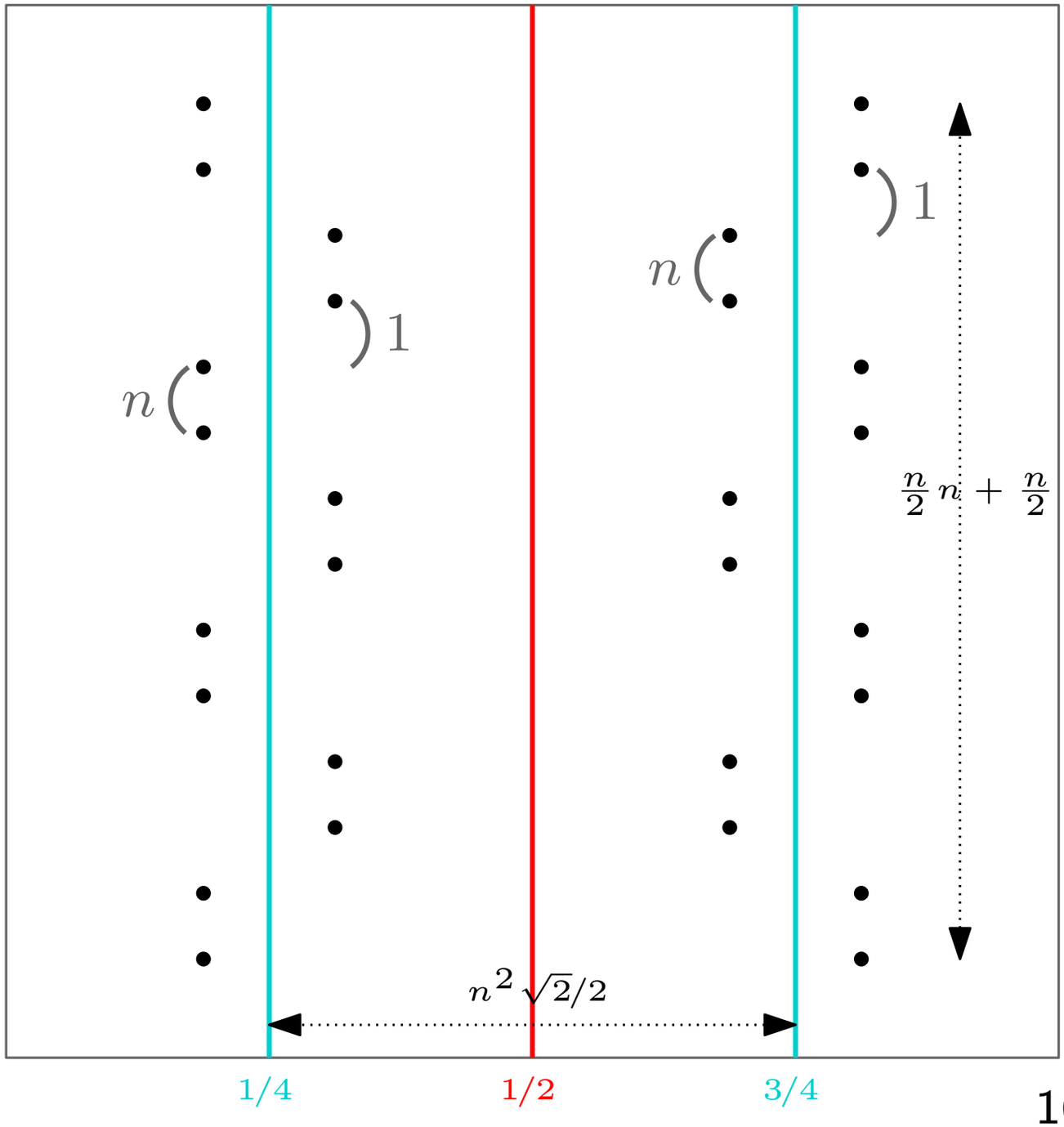
Output quality? Do we have $|\mathrm{OPT}_p| - |\mathrm{OPT}| \leq O(\varepsilon)\,|\mathrm{OPT}|$?



Structure theorem

Pb: $|\mathrm{OPT}_p|$ can be made arbitrarily large compared to $|\mathrm{OPT}|$

$|V| = 2n$

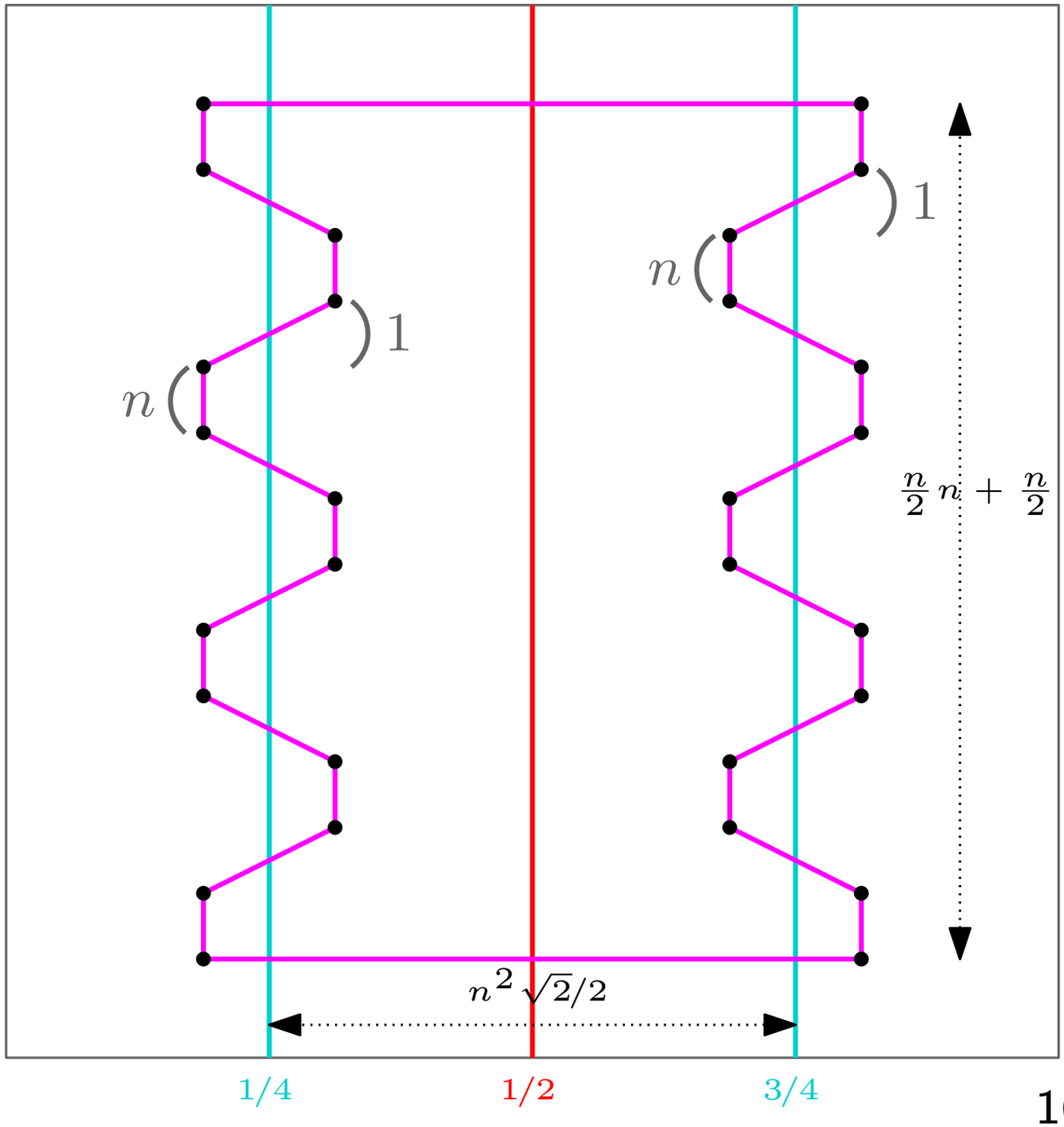


Structure theorem

Pb: $|\mathrm{OPT}_p|$ can be made arbitrarily large compared to $|\mathrm{OPT}|$

$|V| = 2n$

$|\mathrm{OPT}| \leq 2\frac{n}{2}n + 2\frac{n}{2}2\sqrt{2} + 2n^2\frac{\sqrt{2}}{2} = n^2(1+\sqrt{2}) + 2n\sqrt{2}$



Structure theorem

Pb: $|\mathrm{OPT}_p|$ can be made arbitrarily large compared to $|\mathrm{OPT}|$

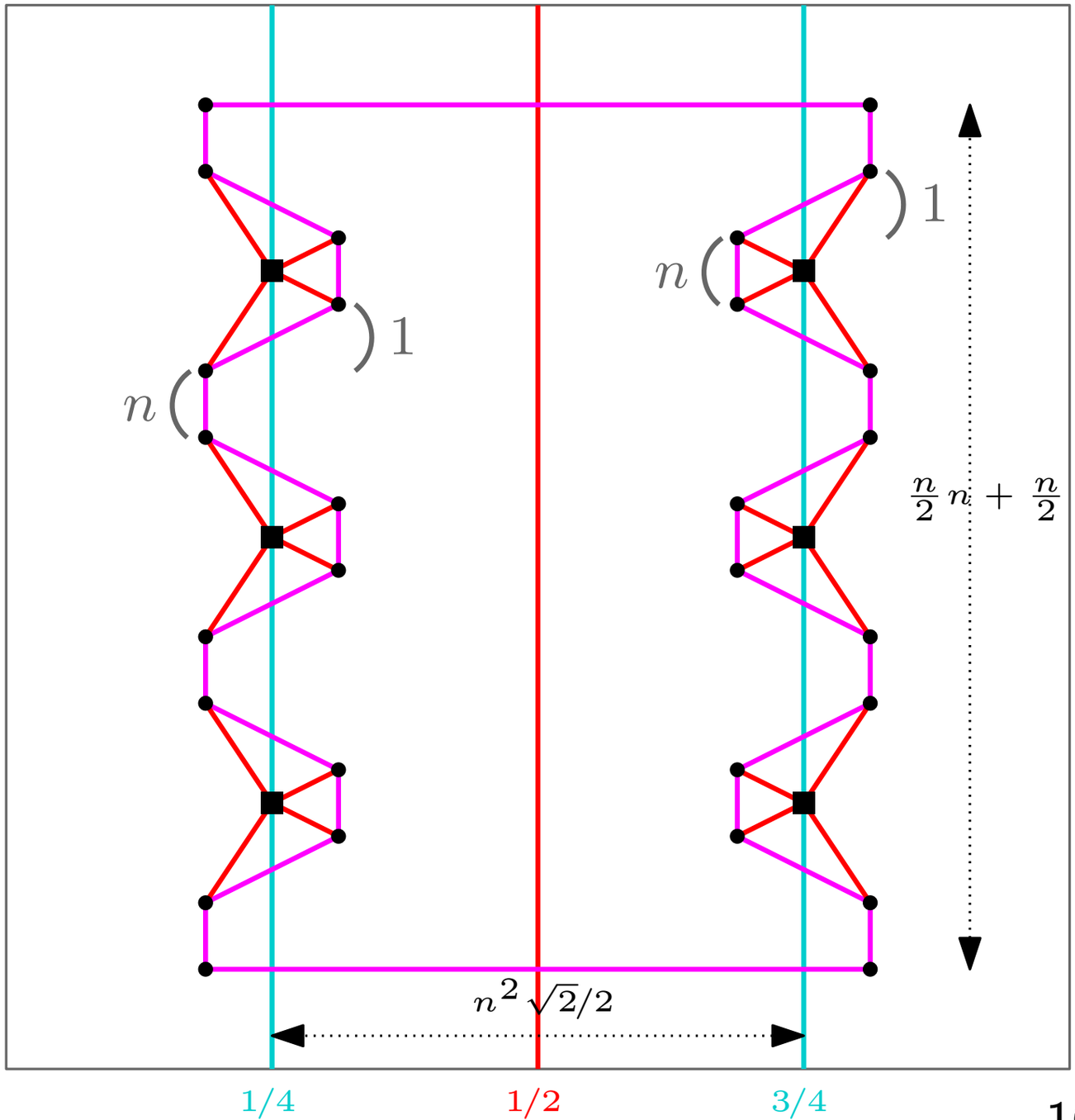
$|V| = 2n$

$|\mathrm{OPT}| \leq 2\frac{n}{2}n + 2\frac{n}{2}2\sqrt{2} + 2n^2\frac{\sqrt{2}}{2} = n^2(1+\sqrt{2}) + 2n\sqrt{2}$

At level 2, $4m$ portals $\Rightarrow$ inter-portal distance $\delta = \frac{n^2+2n}{8m} \gg n$

One crossing every n $\Rightarrow$ overhead per consecutive portals $\geq 2\frac{\delta}{4} = \frac{\delta}{2}$
$\Rightarrow$ total overhead $\geq 4m\frac{\delta}{2} = \frac{(n^2+2n)^2}{4} = \Omega(|\mathrm{OPT}|)$ (indep. of ε)

(same for tours close to OPT)

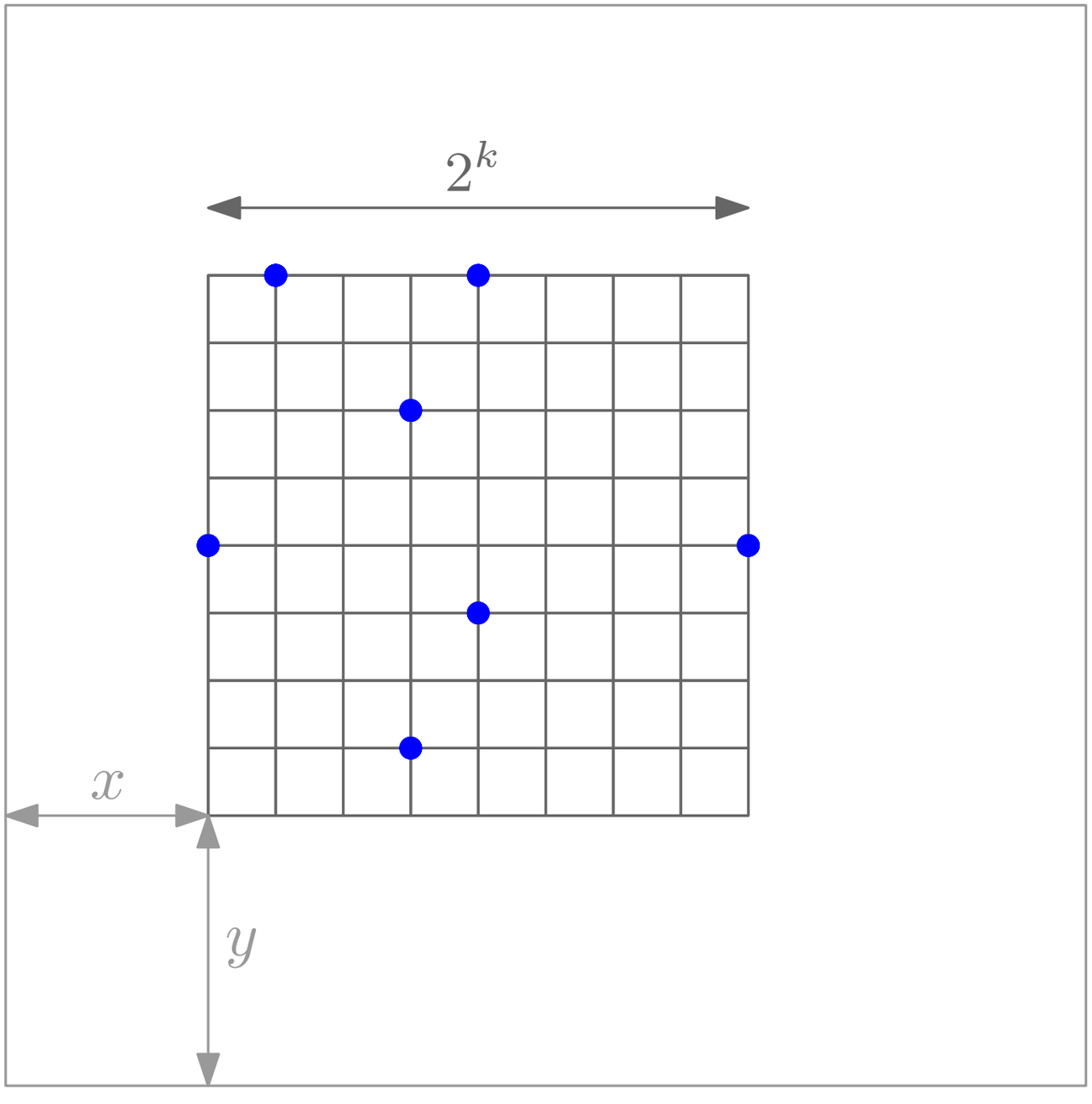


Structure theorem

Pb: $|\mathrm{OPT}_p|$ can be made arbitrarily large compared to $|\mathrm{OPT}|$

Patch: randomize the algorithm:

Choose random integers $0 \leq x, y \leq 2^k$, then apply (2)-(5) to square of sidelength 2^{k+1} shifted by $(-x, -y)$.

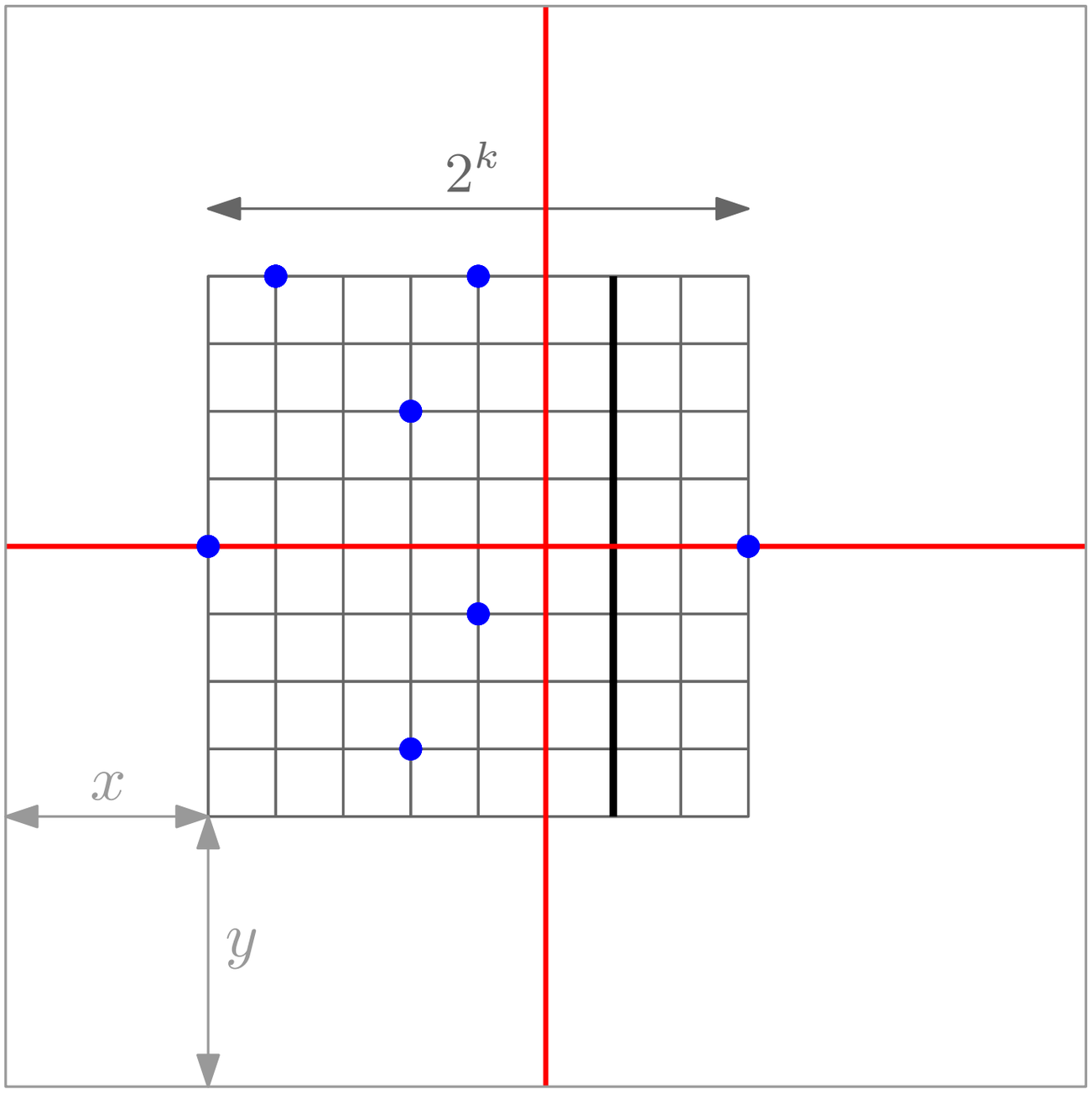


Structure theorem

Thm The expectation (over x, y) of $|\mathrm{OPT}_p| - |\mathrm{OPT}|$ is at most $\frac{k+1}{m}\,|\mathrm{OPT}|$

For any vertical line l in domain,
$P_x(\text{l is at level } i) = \frac{2^{i-2}}{1+2^k}$

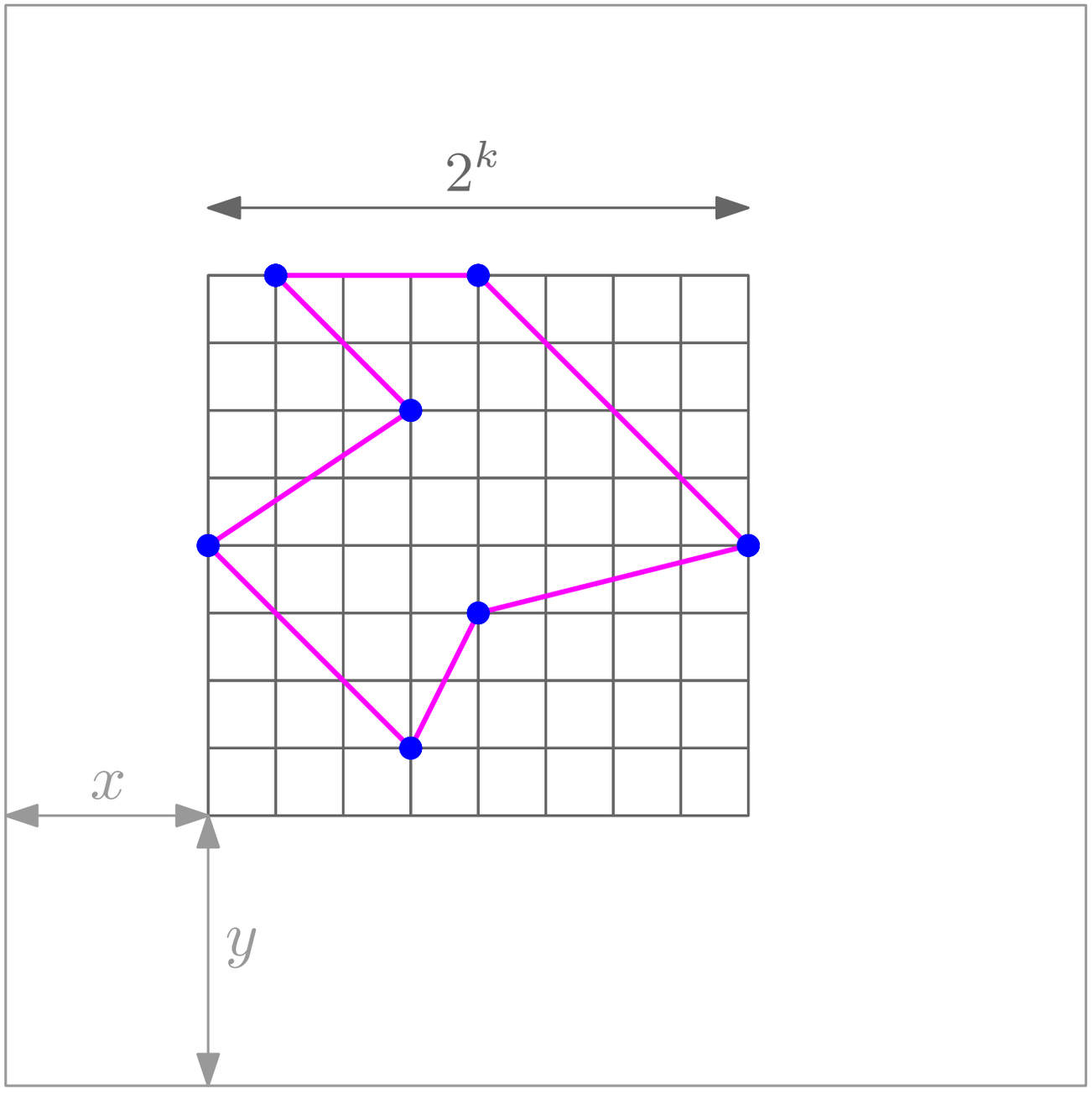
(2^{i-1} level i lines, half of which reach l
$1 + 2^k$ possible values for x



Structure theorem

Thm The expectation (over x, y) of $|\mathrm{OPT}_p| - |\mathrm{OPT}|$ is at most $\frac{k+1}{m} |\mathrm{OPT}|$

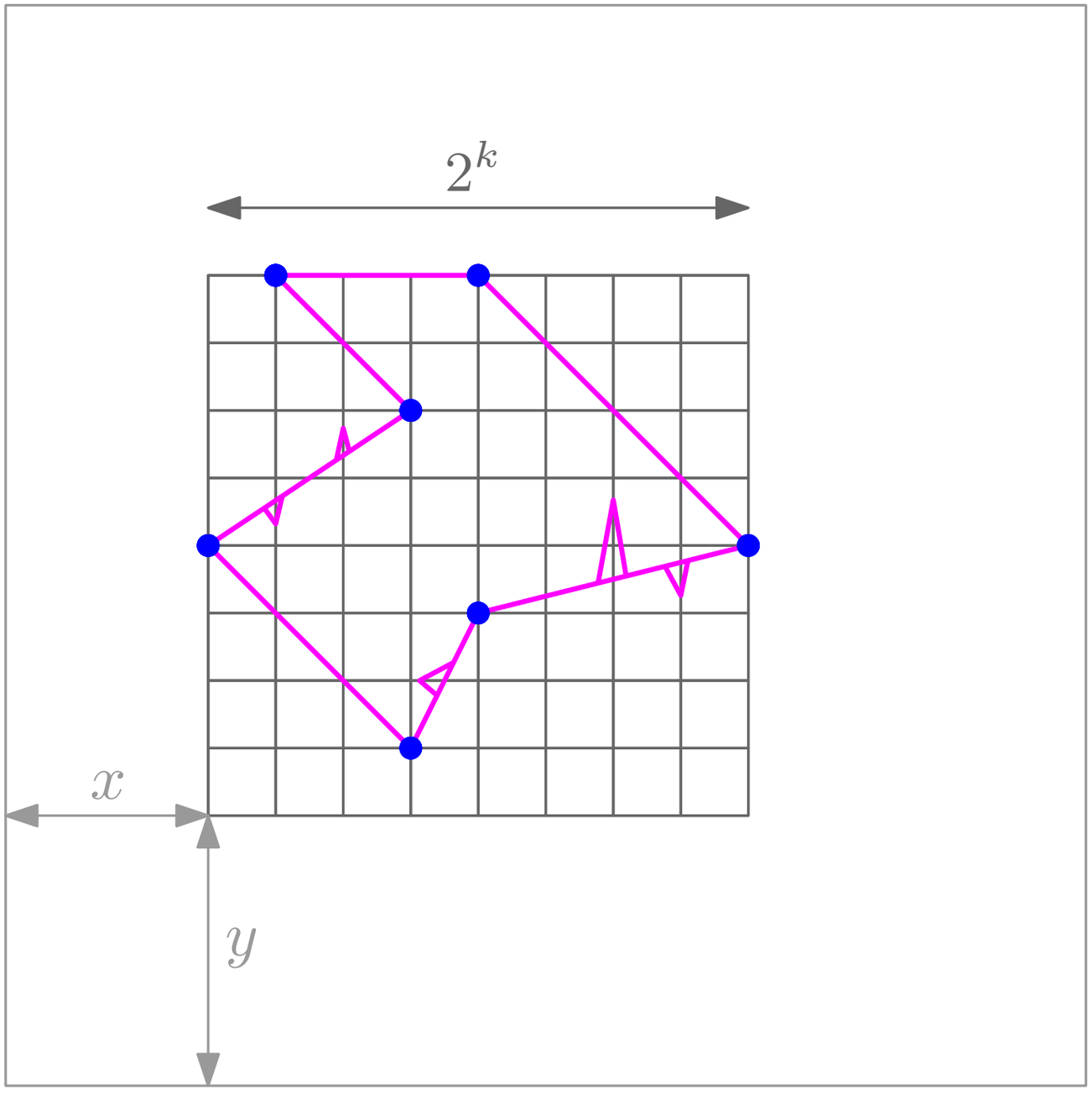
$\rightarrow$ transform OPT into a portal-respecting tour:



Structure theorem

Thm The expectation (over x, y) of $|\mathrm{OPT}_p| - |\mathrm{OPT}|$ is at most $\frac{k+1}{m}\,|\mathrm{OPT}|$

$\rightarrow$ transform OPT into a portal-respecting tour:



Structure theorem

Thm The expectation (over x, y) of $|\mathrm{OPT}_p| - |\mathrm{OPT}|$ is at most $\frac{k+1}{m}\,|\mathrm{OPT}|$

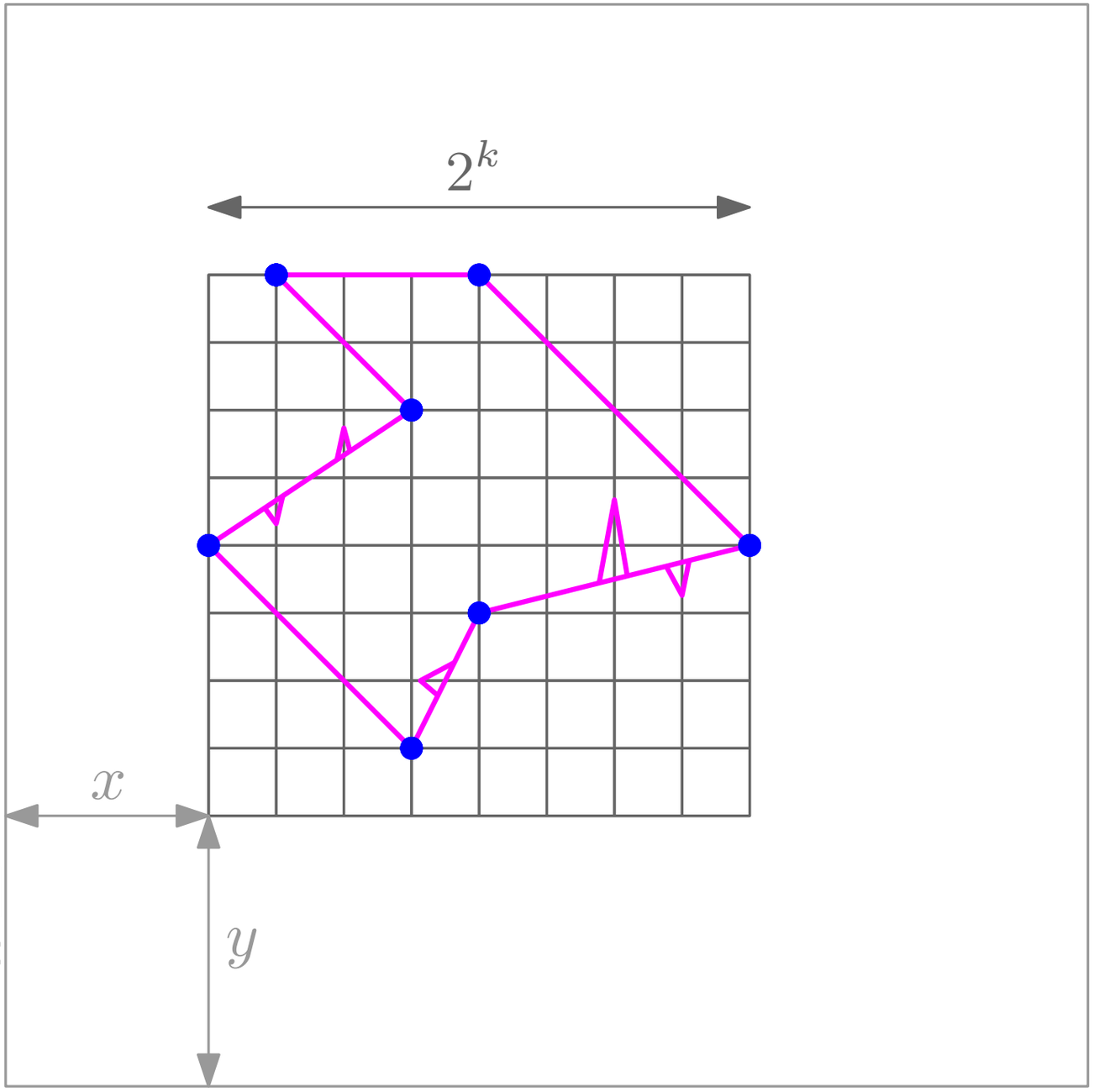
$\rightarrow$ transform OPT into a portal-respecting tour:

For every crossing, overhead ≤ 2 times half the interportal distance $= \frac{2^{k+1}}{m\ 2^i}$

$P_x(\text{level i}) = \frac{2^{i-2}}{1+2^k}$ (same for y)

Expected overhead: $\sum_{i=1}^{k+1} \frac{2^{i-2}}{1+2^k} \frac{2^{k+1}}{m\ 2^i}$
$\leq \sum_{i=1}^{k+1} \frac{2^{i-2}}{2^k} \frac{2^{k+1}}{m\ 2^i} = \frac{k+1}{2m}$

OPT crosses the grid at most $2|\mathrm{OPT}|$ times $\Rightarrow$ total expected overhead: $\frac{k+1}{m}\,|\mathrm{OPT}|$



Structure theorem

Thm The expectation (over x, y) of $|\mathrm{OPT}_p| - |\mathrm{OPT}|$ is at most $\frac{k+1}{m}\,|\mathrm{OPT}| \leq \frac{2\log n + 3/2 + 1}{\log n / 2\varepsilon}\,|\mathrm{OPT}| \leq (4 + 5/\log n)\,\varepsilon\,|\mathrm{OPT}| \leq 9\varepsilon\,|\mathrm{OPT}|.$
$(n \geq 2)$

Structure theorem

Thm The expectation (over x, y) of $|\mathrm{OPT}_p| - |\mathrm{OPT}|$ is at most $\frac{k+1}{m} |\mathrm{OPT}| \leq \frac{2 \log n + 3/2 + 1}{\log n / 2\varepsilon} |\mathrm{OPT}| \leq (4 + 5/\log n) \varepsilon |\mathrm{OPT}| \leq 9\varepsilon |\mathrm{OPT}|$.

Corollary $P_{x,y} (|\mathrm{OPT}_p| - |\mathrm{OPT}| \leq 18\varepsilon |\mathrm{OPT}|) \geq 1/2$

$\rightarrow$ **Monte-Carlo procedure** given a constant $0 < c < 1$, repeat $\lceil \log(1/c) \rceil$ times the process "randomization + (2)-(5)" and keep the best computed tour T. Then, $P (|\mathrm{OPT}_p| - |\mathrm{OPT}| \leq 18\varepsilon |\mathrm{OPT}|) \geq 1 - c$

$\rightarrow$ **Derandomization** try all possible choices of (x, y) (there are $O(n^4)$ of those), and keep best tour.

Take-home messages

• Even though TSP is APX-hard, by choosing a relevant class of inputs, and by using approximation, we can build reasonable solutions in (weakly) polynomial time.

• PTAS works in any arbitrary dimension, and also for all l_p-norms with $p \geq 1$. In fact, PTAS works in any metric space with bounded doubling dimension [Talwar '04].

• Impacts many other NP-hard problems, e.g. Minimum Steiner Tree.

• Has many applications down the road, e.g. curve reconstruction (see TD).