The traveling salesman problem with few inner points

Michael Hoffmann & Yoshio Okamoto*

ETH Zurich

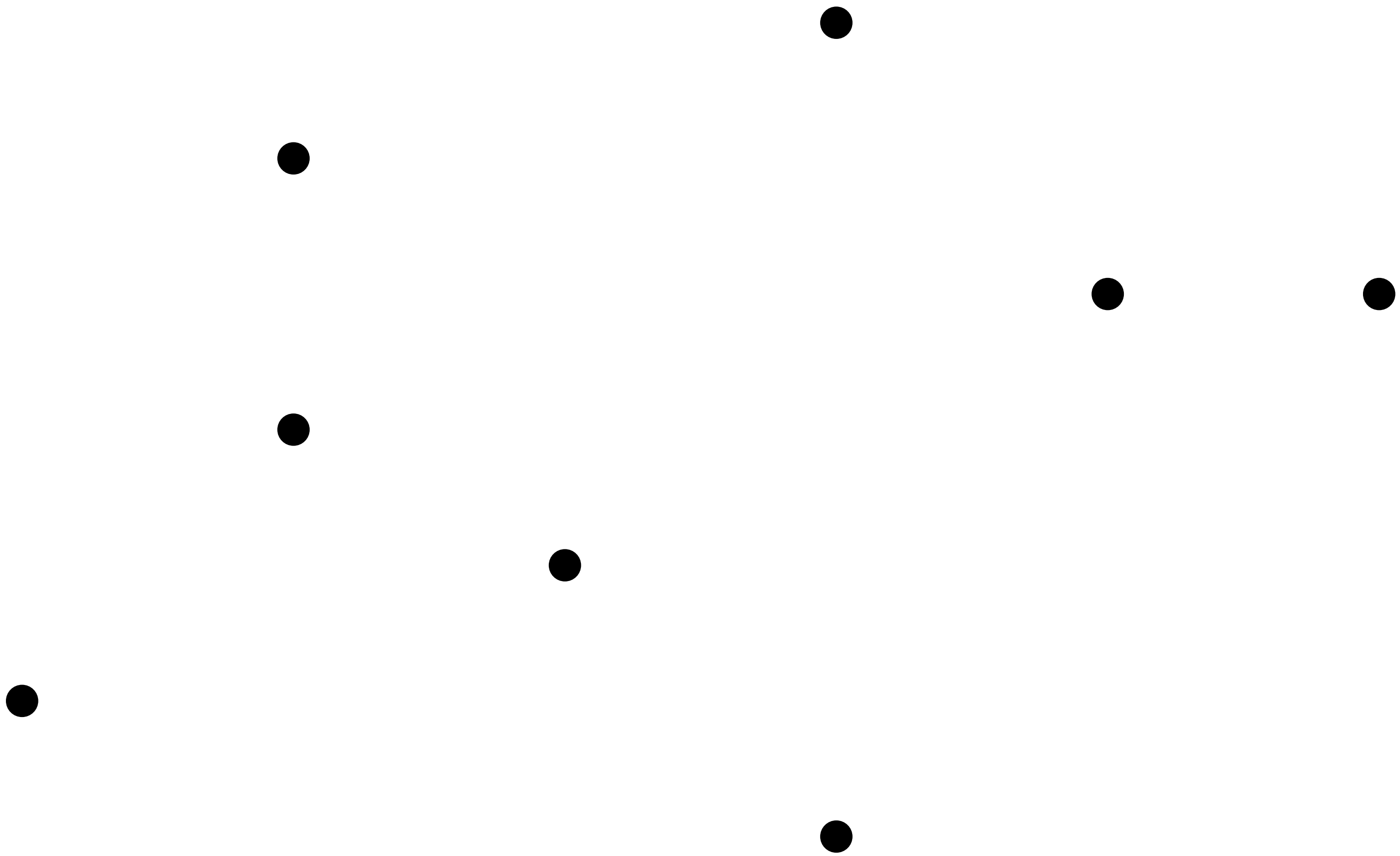
URAW 2004

* Supported by the Berlin-Zürich Joint Graduate Program

CGC Combinatorics Geometry Computation

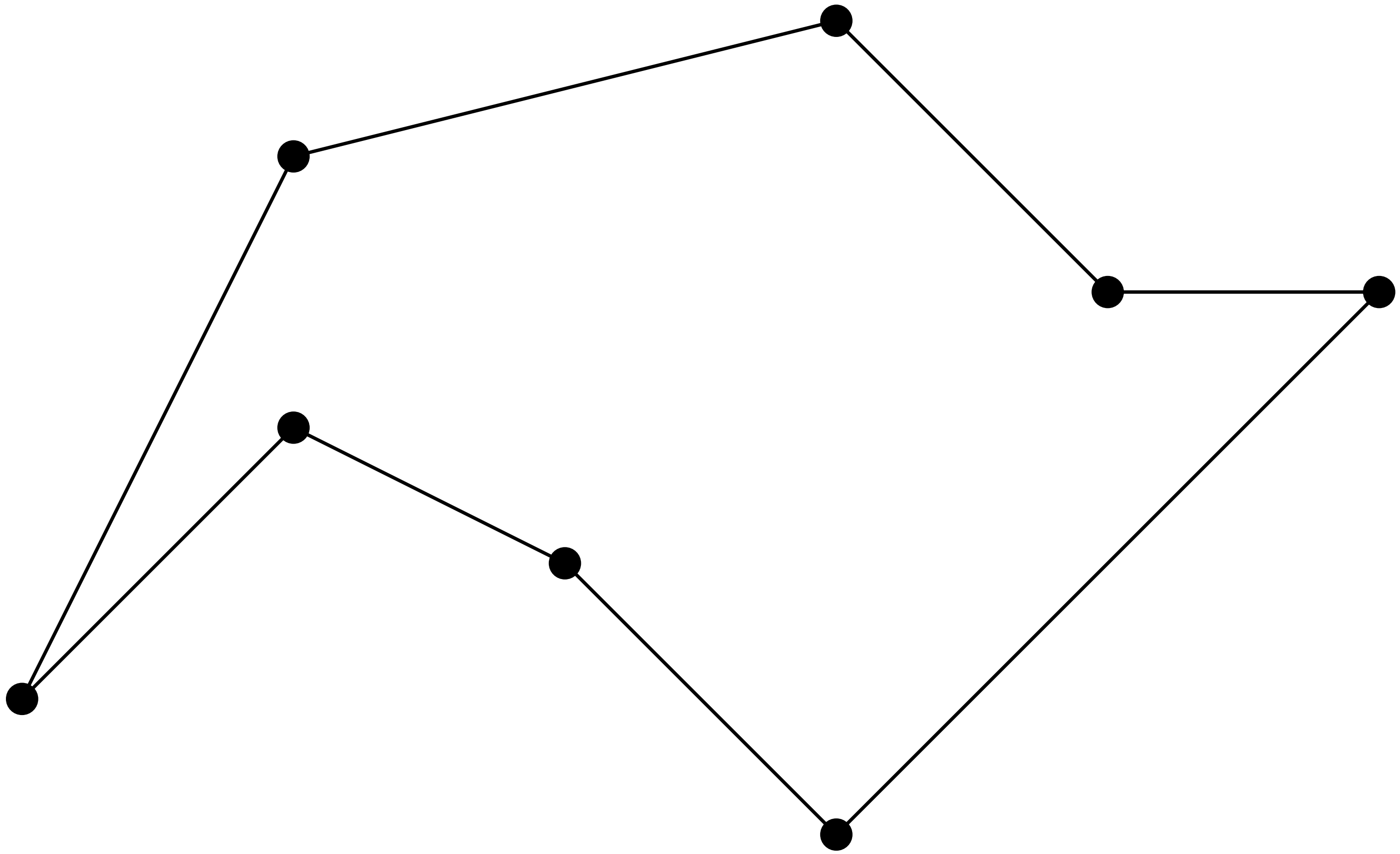
Given: finite set of points on $\mathbb{R}^2$

Find: a minimum-length tour



Given: finite set of points on $\mathbb{R}^2$

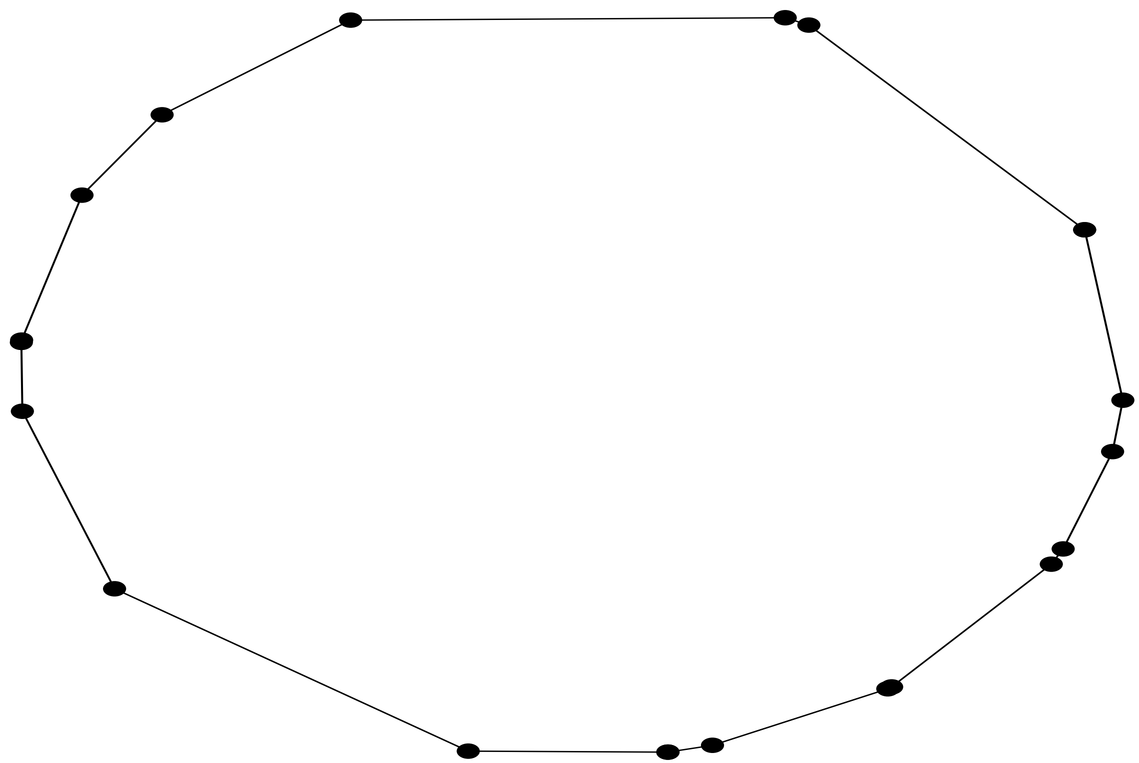
Find: a minimum-length tour



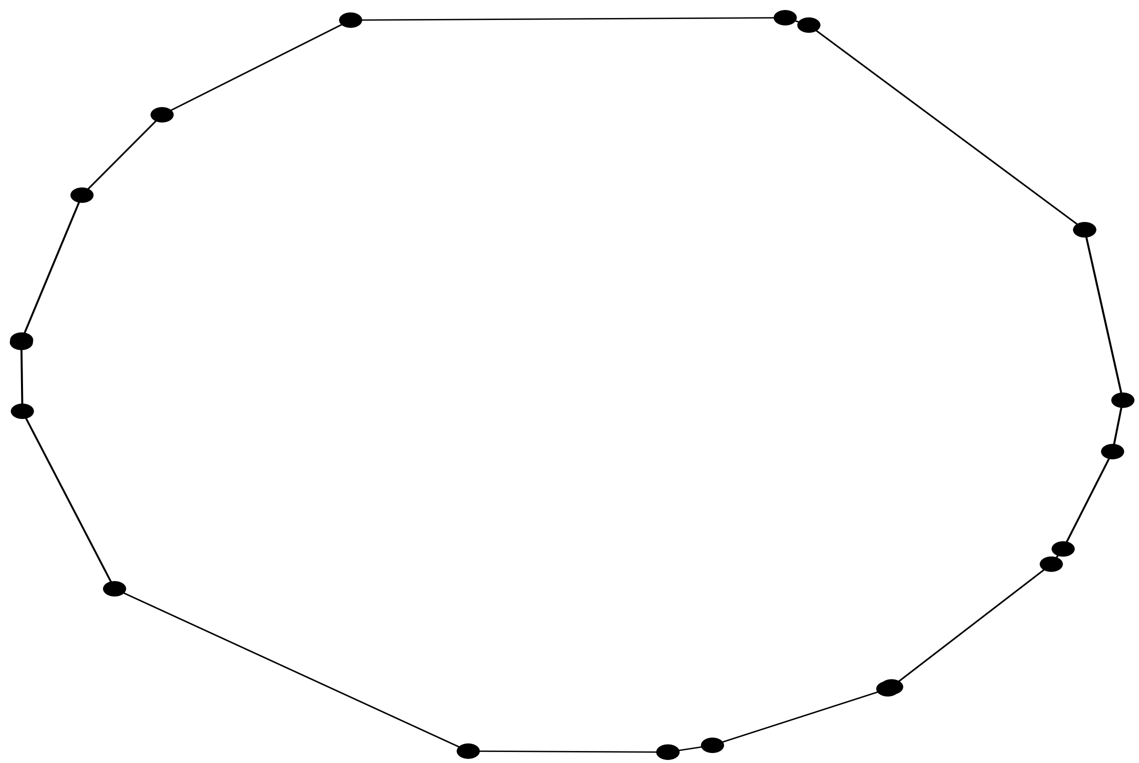
- In general, it is NP-hard.

  (Garey, Graham & Johnson '76
  Papadimitriou '77)

- When the points are in convex position, the problem is easy.

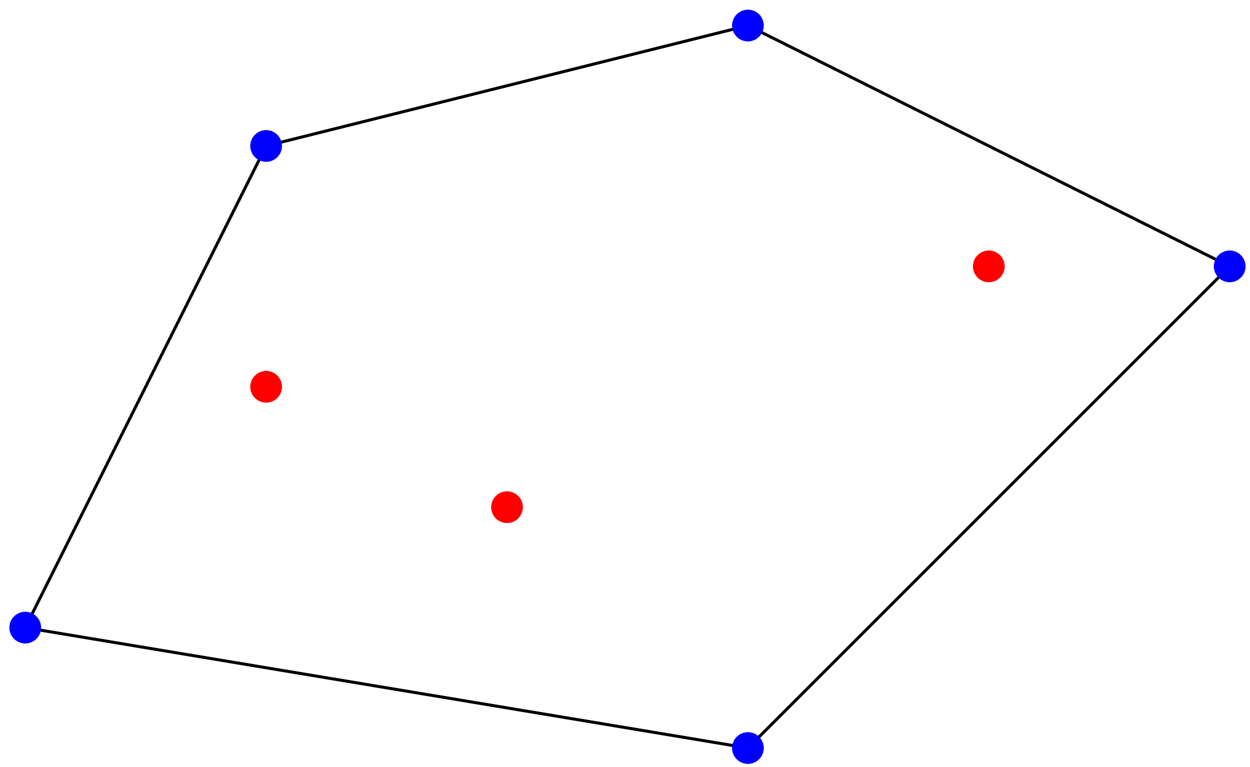


- In general, it is NP-hard.
  (Garey, Graham & Johnson '76
  Papadimitriou '77)
- When the points are in convex position, the problem is easy.



Observation

The **inner points** make the problem difficult.



Observation

The **inner points** make the problem difficult.

Motivation

How many inner points can we have in order to obtain a polynomial-time algorithm?

Result

We give two simple algorithms.

$n :=$ the total number of points
$k :=$ the number of inner points

- First algorithm runs in polynomial time when $k = O(1)$.
- Second algorithm runs in polynomial time when $k = O(\log n / \log\log n)$.

Open problem: Improve the bound!

Result

We give two simple algorithms.

$n :=$ the total number of points
$k :=$ the number of inner points

- First algorithm runs in polynomial time when $k = O(1)$.
- Second algorithm runs in polynomial time when $k = O(\log n / \log \log n)$.

Open problem: Improve the bound!

Result

We give two simple algorithms.

$n :=$ the total number of points
$k :=$ the number of inner points

- First algorithm runs in polynomial time when $k = O(1)$.
- Second algorithm runs in polynomial time when $k = O(\log n / \log \log n)$.

Open problem: Improve the bound!

Result

We give two simple algorithms.

$n :=$ the total number of points
$k :=$ the number of inner points

- First algorithm runs in polynomial time when $k = O(1)$.
- Second algorithm runs in polynomial time when $k = O(\log n / \log \log n)$.

Open problem: Improve the bound!

Result

We give two simple algorithms.

$n :=$ the total number of points
$k :=$ the number of inner points

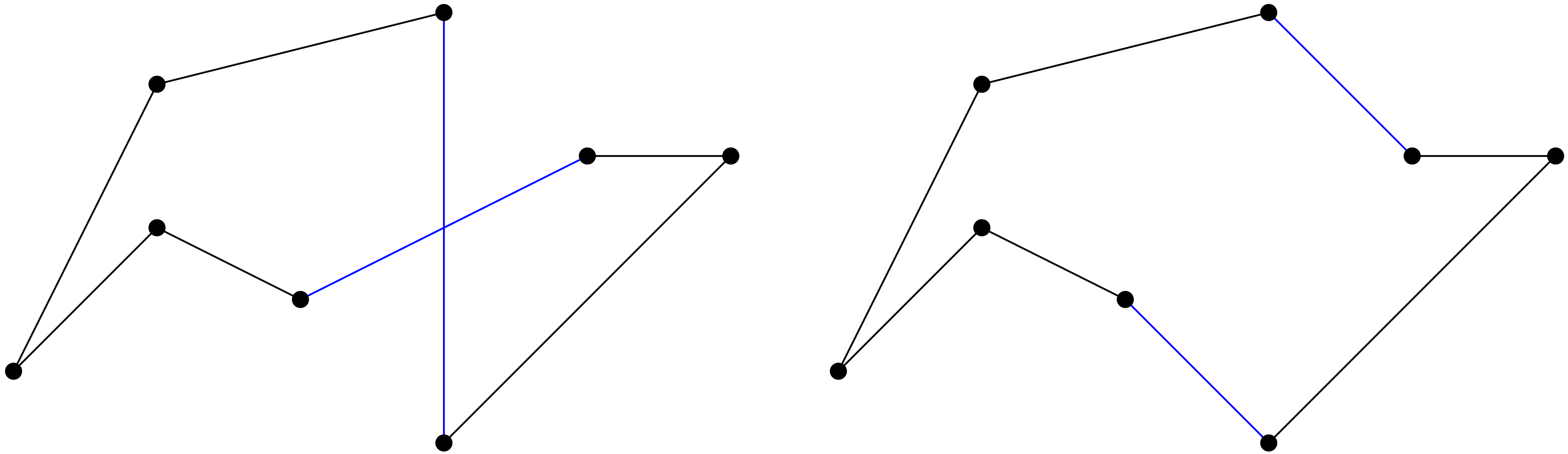
- First algorithm runs in polynomial time when $k = O(1)$.
- Second algorithm runs in polynomial time when $k = O(\log n / \log \log n)$.

Open problem: Improve the bound!

Fact (Flood '56)

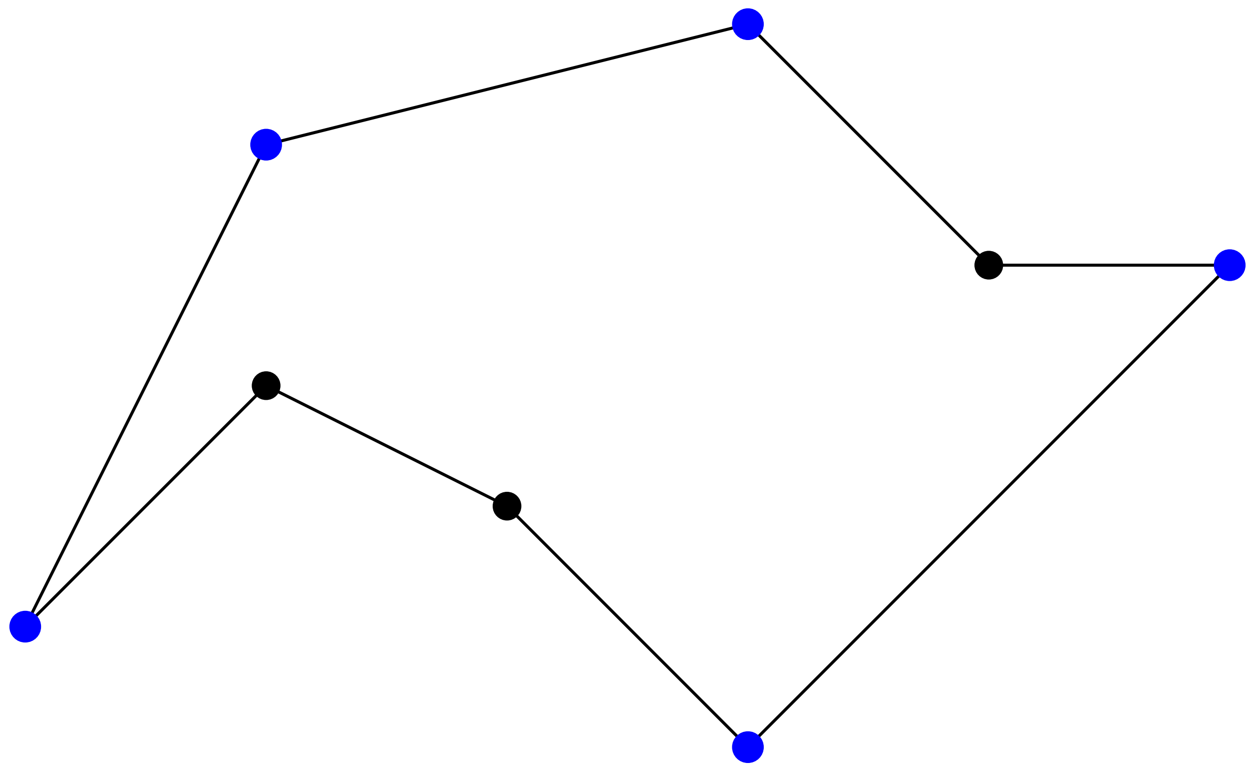
An optimal tour has no self-crossing.

Proof



Corollary

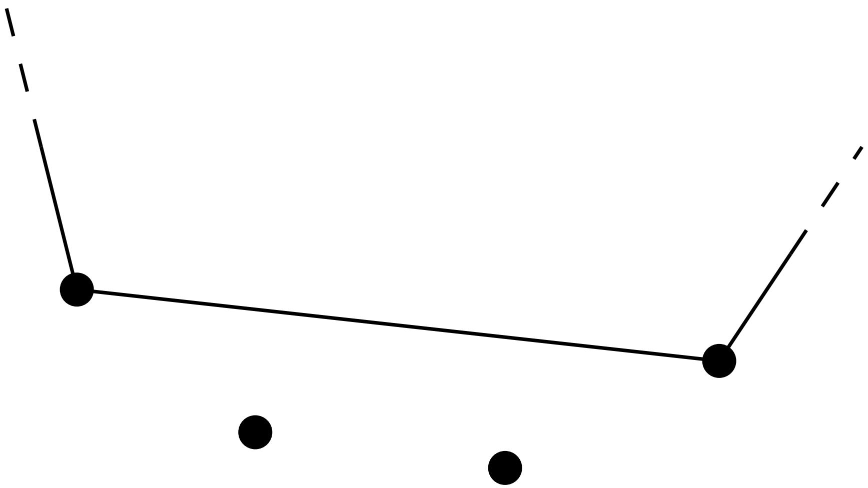
An optimal tour visits the non-inner points in a cyclic order.



Corollary

An optimal tour visits the non-inner points in a cyclic order.

Proof



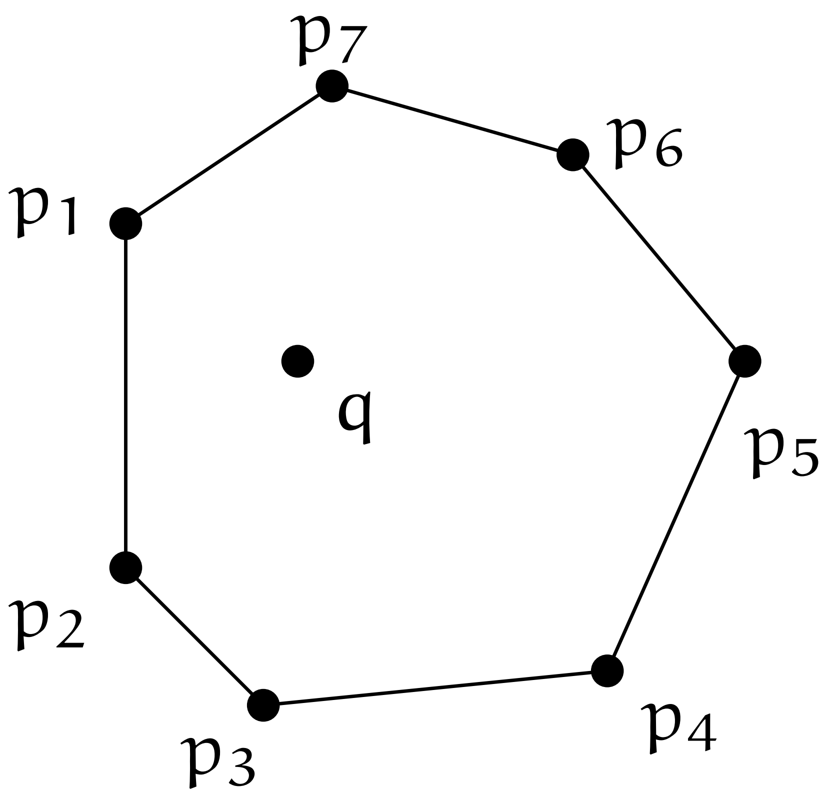
Suppose not.
Then $\exists$ a "skip."
Skipped points must be visited later, which causes a self-crossing.
A contradiction.

Consider the case $k = 1$. ($k := \#$ of inner pts)

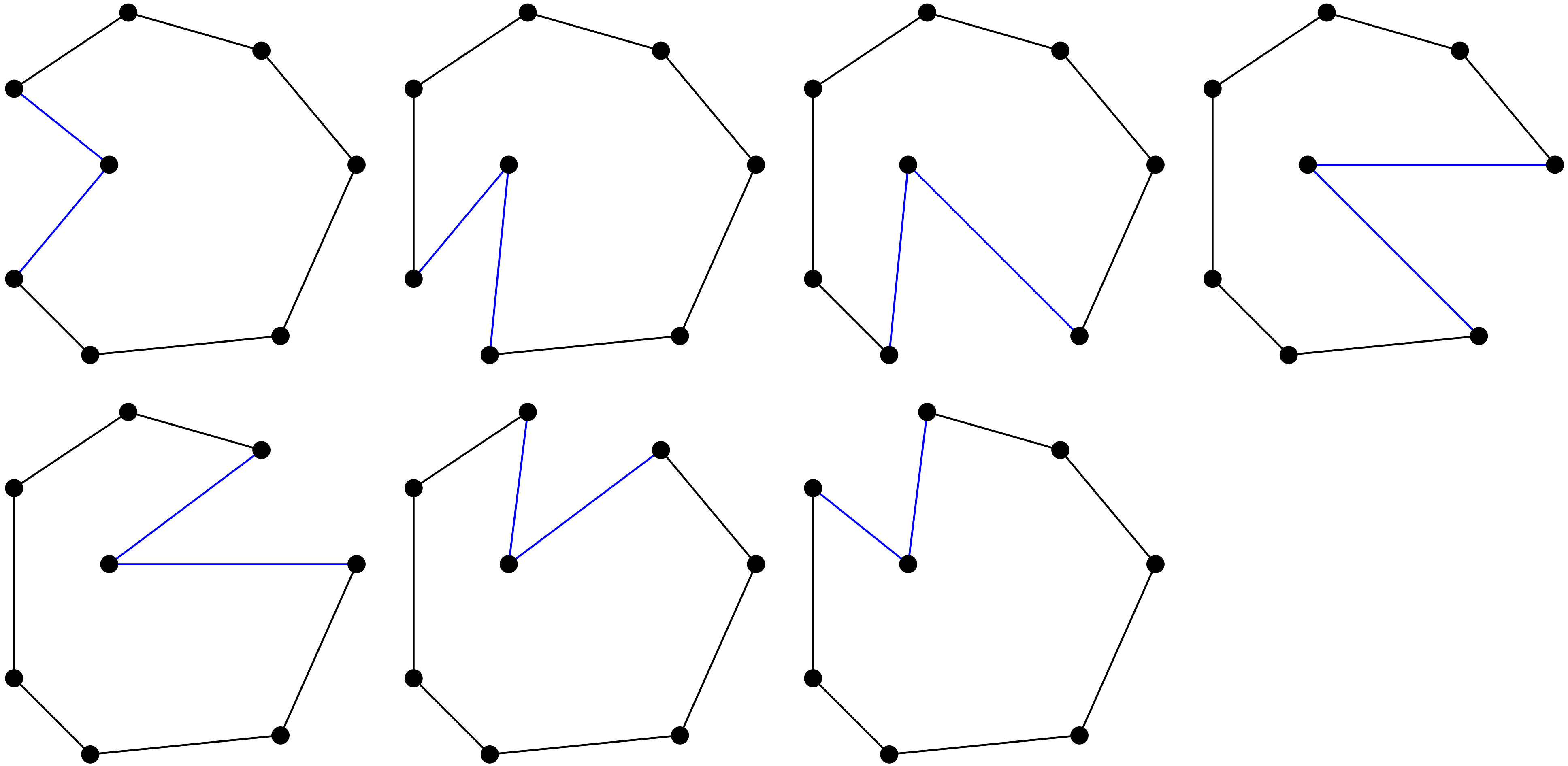
Inner point: q

Non-inner points: $p_1, p_2, \ldots, p_{n-1}$

labeled according to a cyclic order



# of tours which "respect" the cycl. order = $n-1$.



Choose the best one.

(1) Distinguish the inner points and the non-inner points;

(2) Fix a cyclic order on the non-inner points;

(3) For each tour which respects the cyclic order

  (i) Compute the length of the tour;

(4) Choose the best one among them.

(1) Distinguish the inner points and the non-inner points;

(2) Fix a cyclic order on the non-inner points;

(3) For each tour which respects the cyclic order

  (i) Compute the length of the tour;

(4) Choose the best one among them.

(1) Distinguish the inner points and the non-inner points;

(2) Fix a cyclic order on the non-inner points;

(3) For each tour which respects the cyclic order

  (i) Compute the length of the tour;

(4) Choose the best one among them.

(1) Distinguish the inner points and the non-inner points;

(2) Fix a cyclic order on the non-inner points;

(3) For each tour which respects the cyclic order

  (i) Compute the length of the tour;

(4) Choose the best one among them.

(1) Distinguish the inner points and the non-inner points;

(2) Fix a cyclic order on the non-inner points;

(3) For each tour which respects the cyclic order

  (i) Compute the length of the tour;

(4) Choose the best one among them.

(1) Distinguish the inner points and the non-inner points;

(2) Fix a cyclic order on the non-inner points;

(3) For each tour which respects the cyclic order

  (i) Compute the length of the tour;

(4) Choose the best one among them.

Running time

There are k inner points.

- \# of tours which "respect" the cycl. order $= O(k!n^k)$.
- They can be enumerated in $O(1)$ time per tour.
- The length of each tour can be computed in $O(n)$ time.

The running time $= \underbrace{O(n \log n)}_{\text{convex hull computation}} + O(k!n^{k+1})$.

When k is a constant, this is polynomial in n.

There are k inner points.

- # of tours which "respect" the cycl. order $= O(k!n^k)$.
- They can be enumerated in $O(1)$ time per tour.
- The length of each tour can be computed in $O(n)$ time.

The running time $= \underbrace{O(n \log n)}_{\text{convex hull computation}} + O(k!n^{k+1})$.

When k is a constant, this is polynomial in n.

Running time

There are k inner points.

- \# of tours which "respect" the cycl. order $= O(k!n^k)$.
- They can be enumerated in $O(1)$ time per tour.
- The length of each tour can be computed in $O(n)$ time.

The running time $= \underbrace{O(n \log n)}_{\text{convex hull computation}} + O(k!n^{k+1})$.

When k is a constant, this is polynomial in n.

There are k inner points.

- $\#$ of tours which "respect" the cycl. order $= O(k!n^k)$.
- They can be enumerated in $O(1)$ time per tour.
- The length of each tour can be computed in $O(n)$ time.

The running time $= \underbrace{O(n\log n)}_{\text{convex hull computation}} + O(k!n^{k+1})$.

When k is a constant, this is polynomial in n.

Running time

There are k inner points.

- $\#$ of tours which "respect" the cycl. order $= O(k!n^k)$.
- They can be enumerated in $O(1)$ time per tour.
- The length of each tour can be computed in $O(n)$ time.

The running time $= \underbrace{O(n \log n)}_{\text{convex hull computation}} + O(k!n^{k+1})$.

When k is a constant, this is polynomial in n.

There are k inner points.

- # of tours which "respect" the cycl. order $= O(k!n^k)$.
- They can be enumerated in $O(1)$ time per tour.
- The length of each tour can be computed in $O(n)$ time.

The running time $= \underbrace{O(n \log n)}_{\text{convex hull computation}} + O(k!n^{k+1})$.

When k is a constant, this is polynomial in n.

Result

We give two simple algorithms.

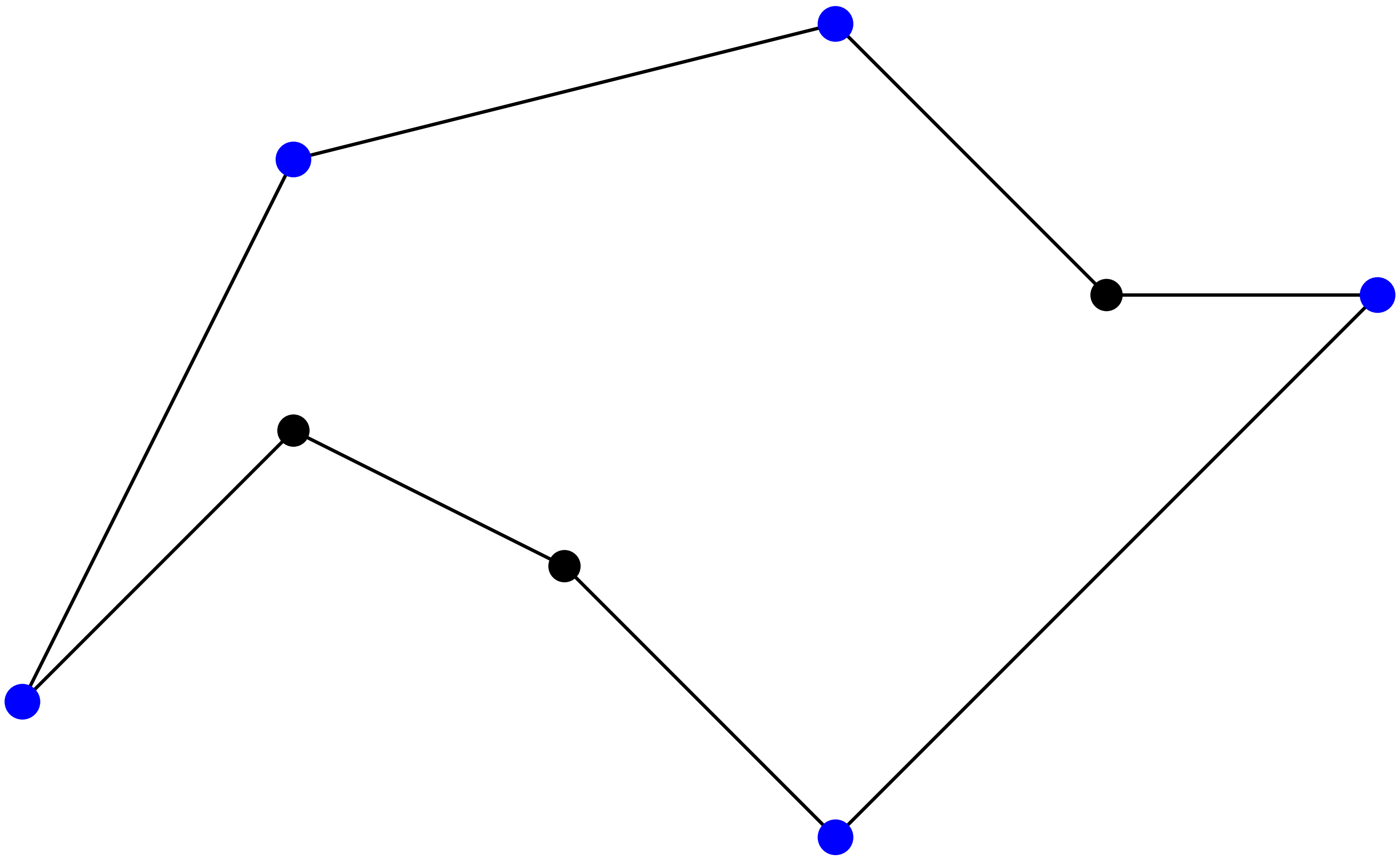
$n :=$ the total number of points
$k :=$ the number of inner points

- First algorithm runs in polynomial time when $k = O(1)$.
- Second algorithm runs in polynomial time when $k = O(\log n / \log\log n)$.

Open problem: Improve the bound!

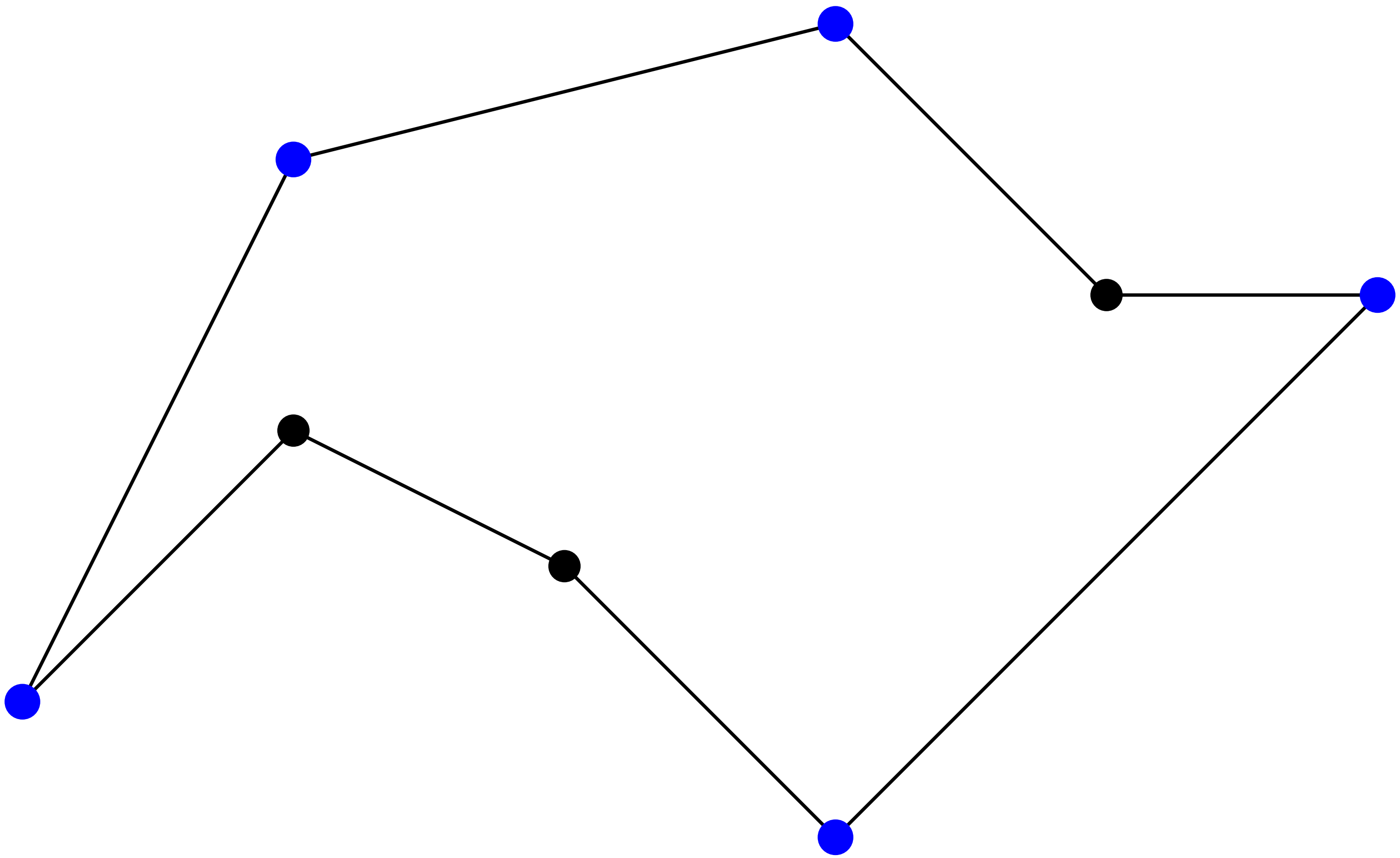
Fact we already saw

An optimal tour respects
a cyclic order on the non-inner points.



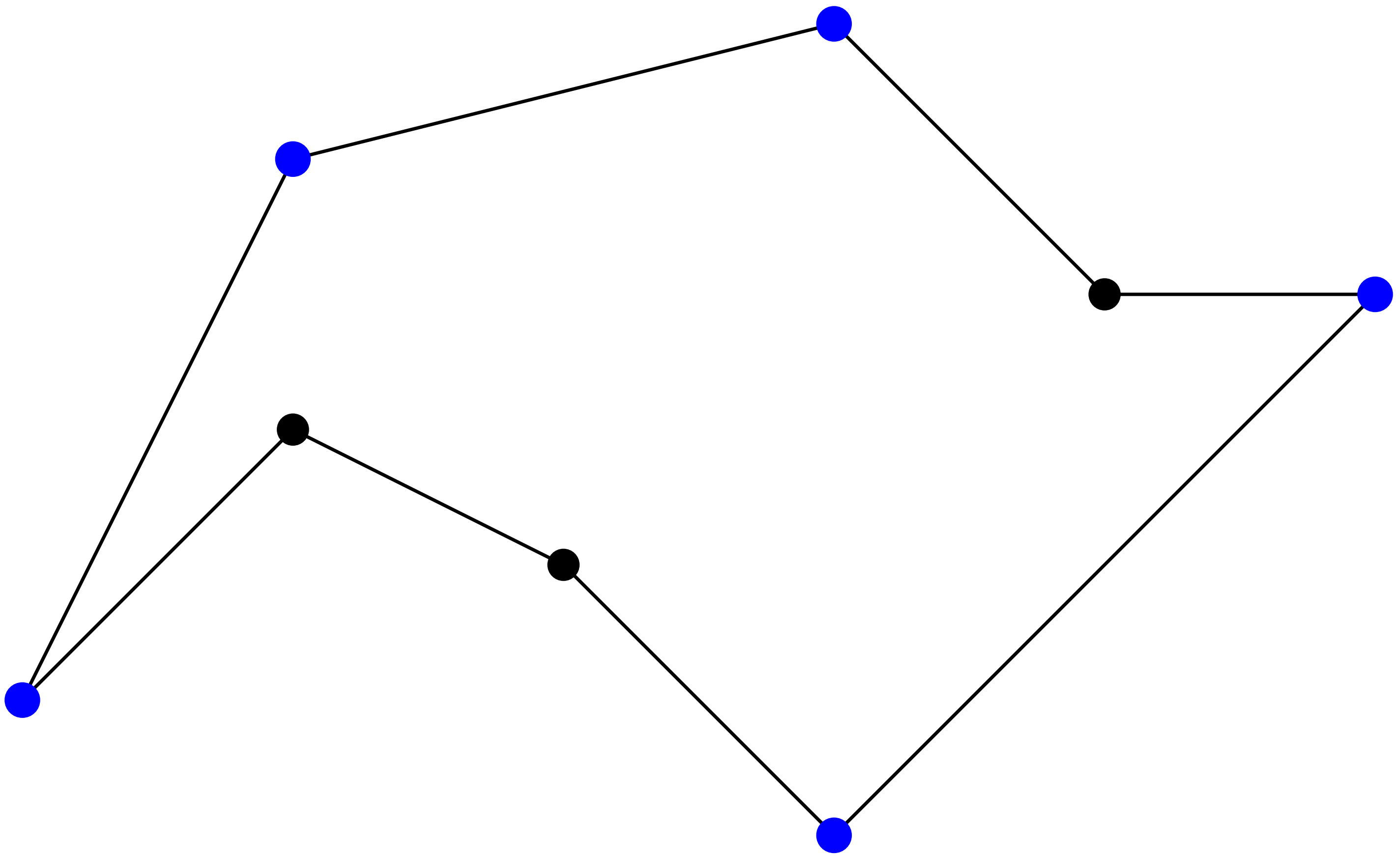
Another fact

An optimal tour respects
some linear order on the inner points.



Idea

Try all linear orders on the inner points.



Outline of the second algorithm

(1) Distinguish the inner points and the non-inner points;

(2) Fix a cyclic order on the non-inner points;

(3) For each linear order on the inner points

- **(i)** Compute an optimal tour among those which respect these two orders;

(4) Choose the best one among them.

(1) Distinguish the inner points and the non-inner points;

(2) Fix a cyclic order on the non-inner points;

(3) For each linear order on the inner points

  (i) Compute an optimal tour among those which respect these two orders;

(4) Choose the best one among them.

Outline of the second algorithm

(1) Distinguish the inner points and the non-inner points;

(2) Fix a cyclic order on the non-inner points;

(3) For each linear order on the inner points

  (i) Compute an optimal tour among those which respect these two orders;

(4) Choose the best one among them.

Outline of the second algorithm

(1) Distinguish the inner points and the non-inner points;

(2) Fix a cyclic order on the non-inner points;

(3) For each linear order on the inner points

- **(i)** Compute an optimal tour among those which respect these two orders;

(4) Choose the best one among them.

(1) Distinguish the inner points and the non-inner points;

(2) Fix a cyclic order on the non-inner points;

(3) For each linear order on the inner points

  (i) Compute an optimal tour among those which respect these two orders;

(4) Choose the best one among them.

Outline of the second algorithm

(1) Distinguish the inner points and the non-inner points;

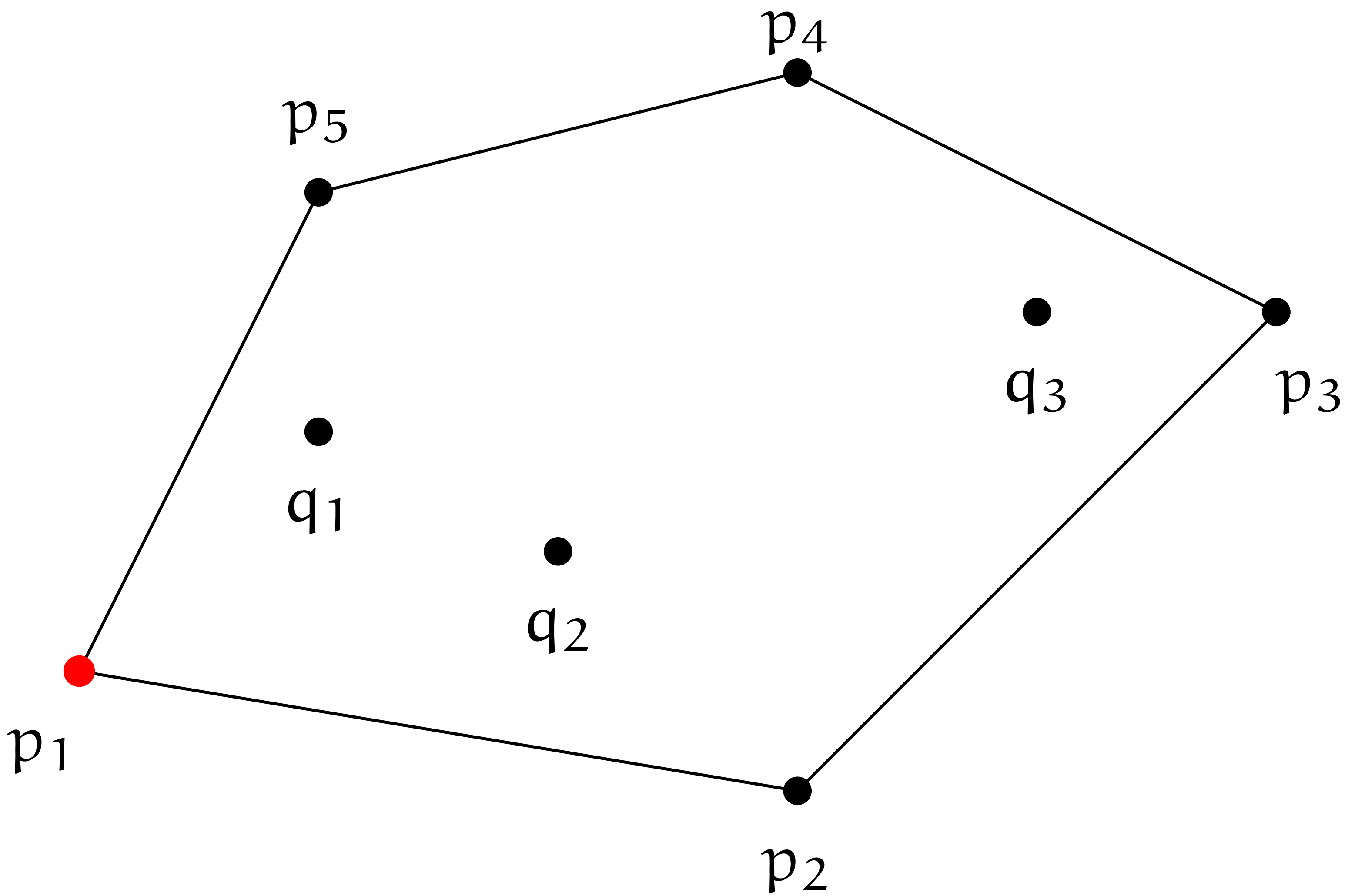
(2) Fix a cyclic order on the non-inner points;

(3) For each linear order on the inner points

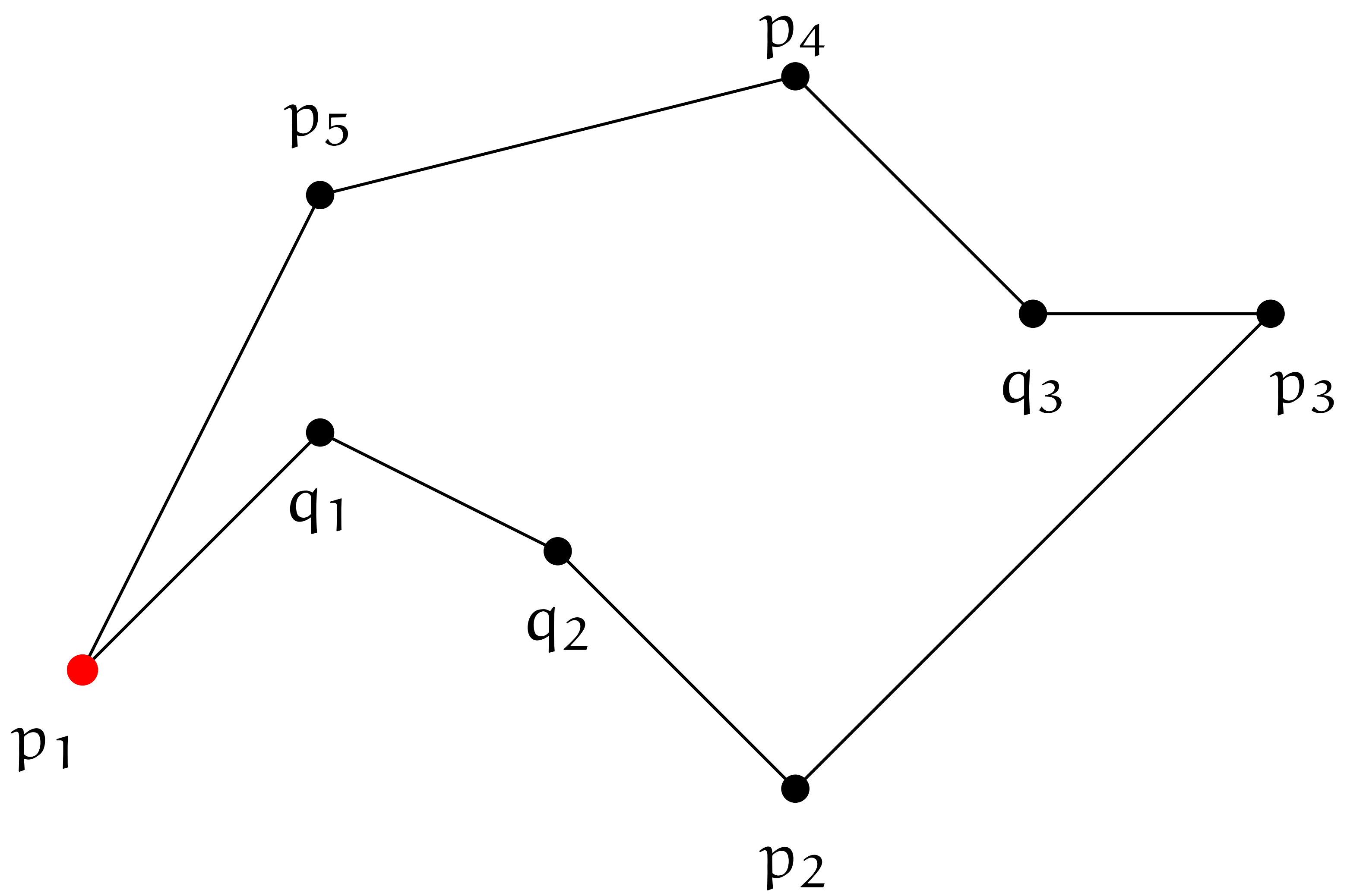
- **(i)** Compute an optimal tour among those which respect these two orders;

(4) Choose the best one among them.

Example

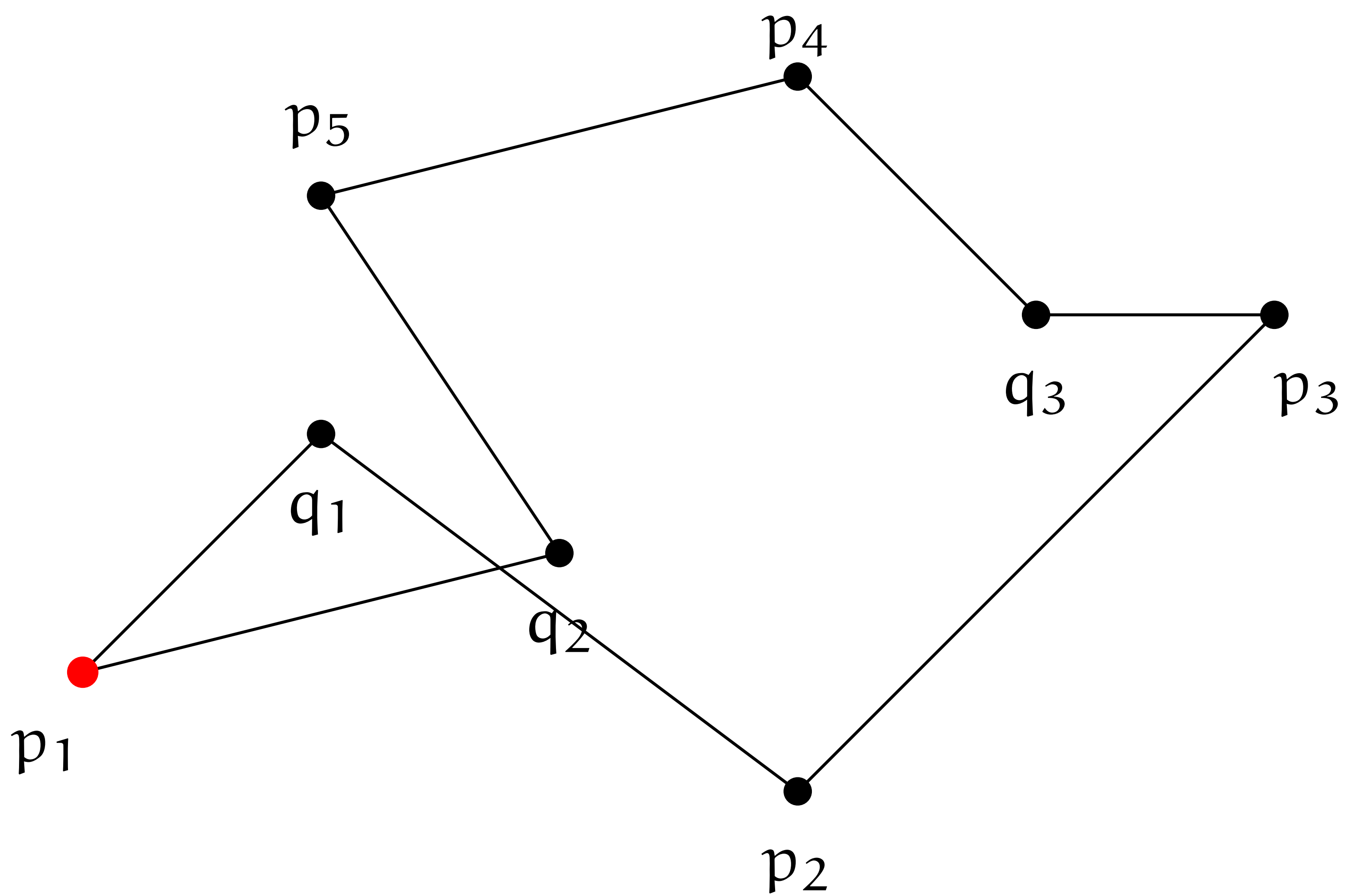


Example



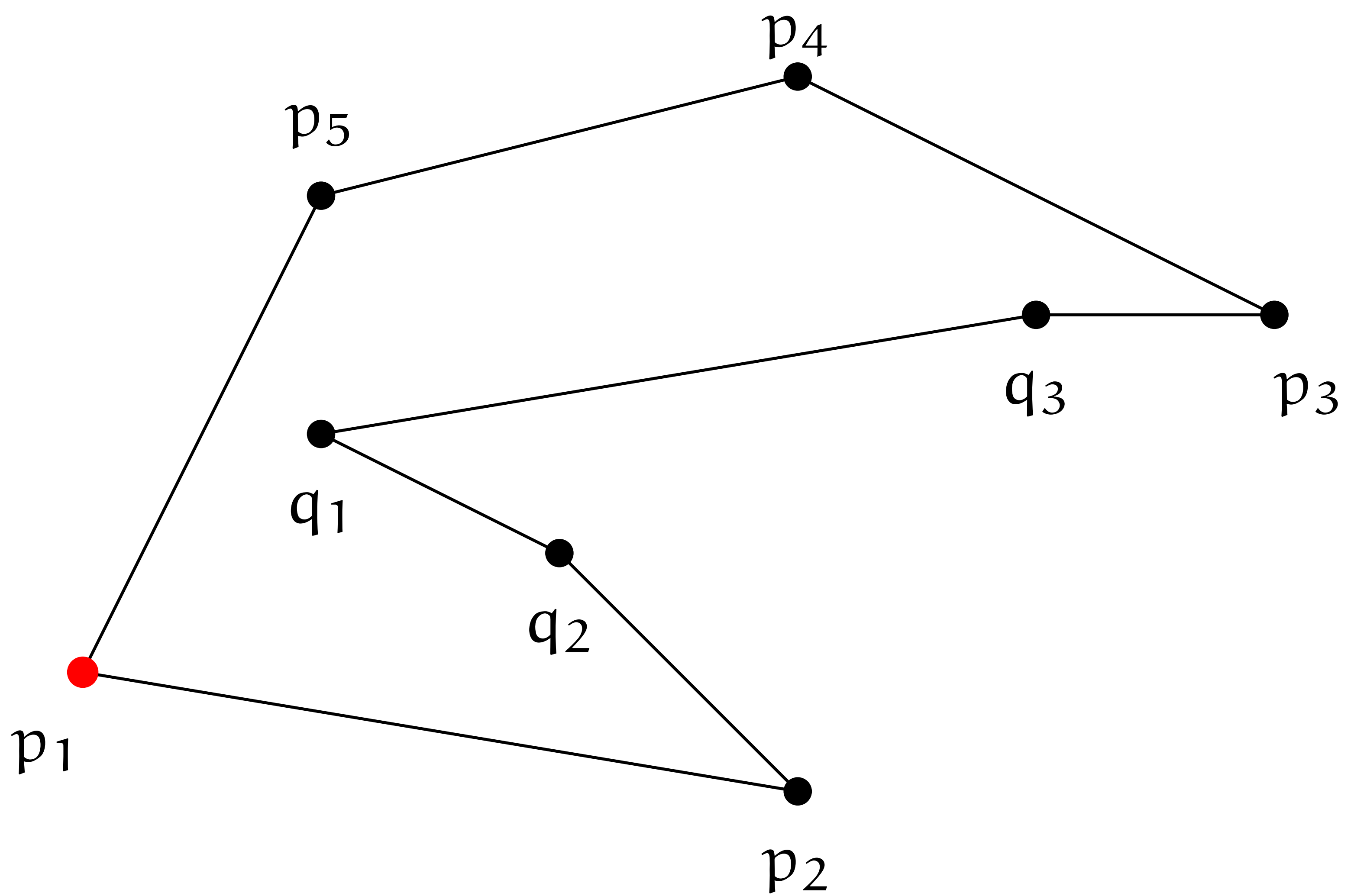
Optimal tour among those which respect the cyclic order and the order “1–2–3.”

Example

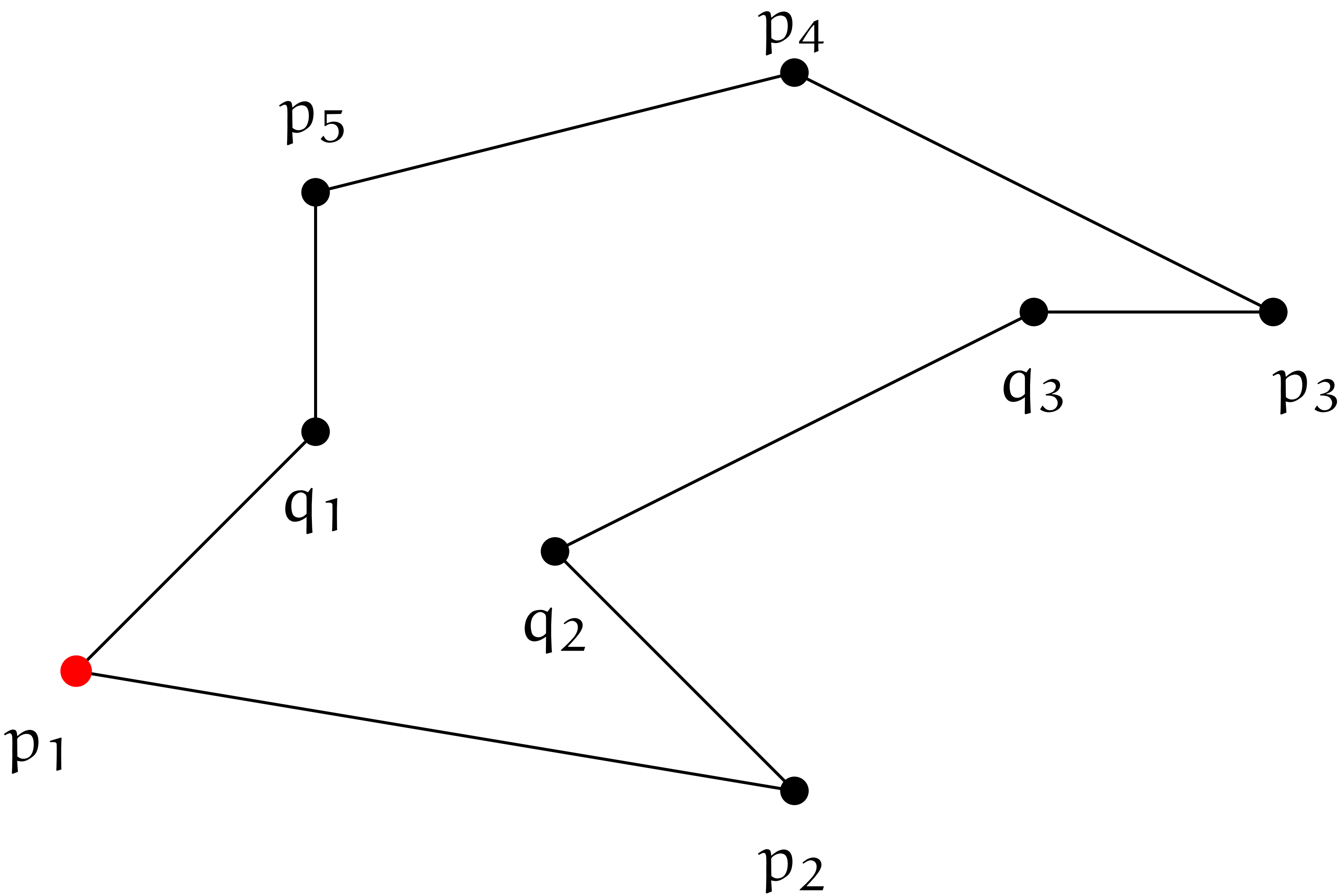


Optimal tour among those which respect the cyclic order and the order "1–3–2."

Example

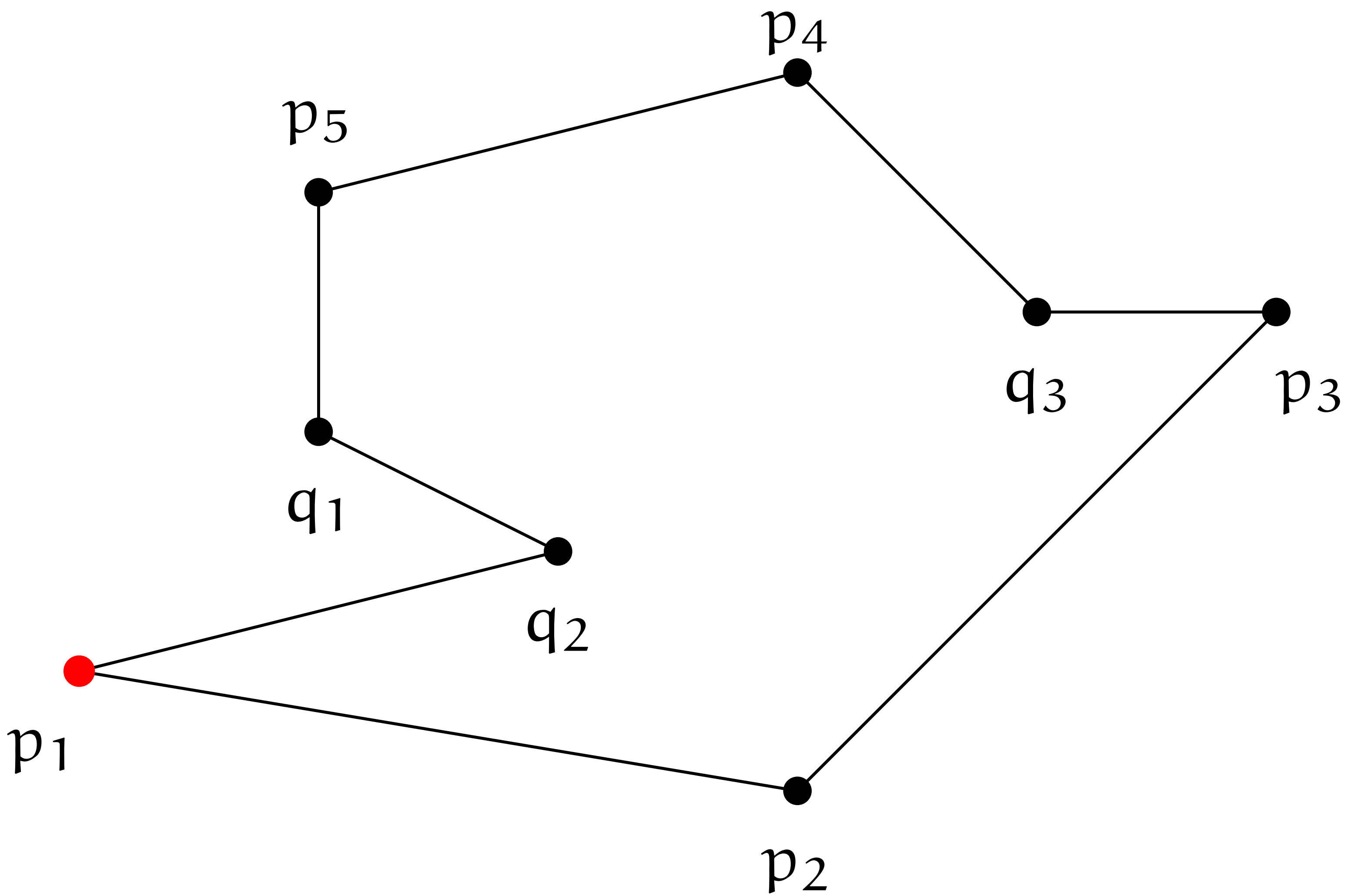


Optimal tour among those which respect the cyclic order and the order “2–1–3.”

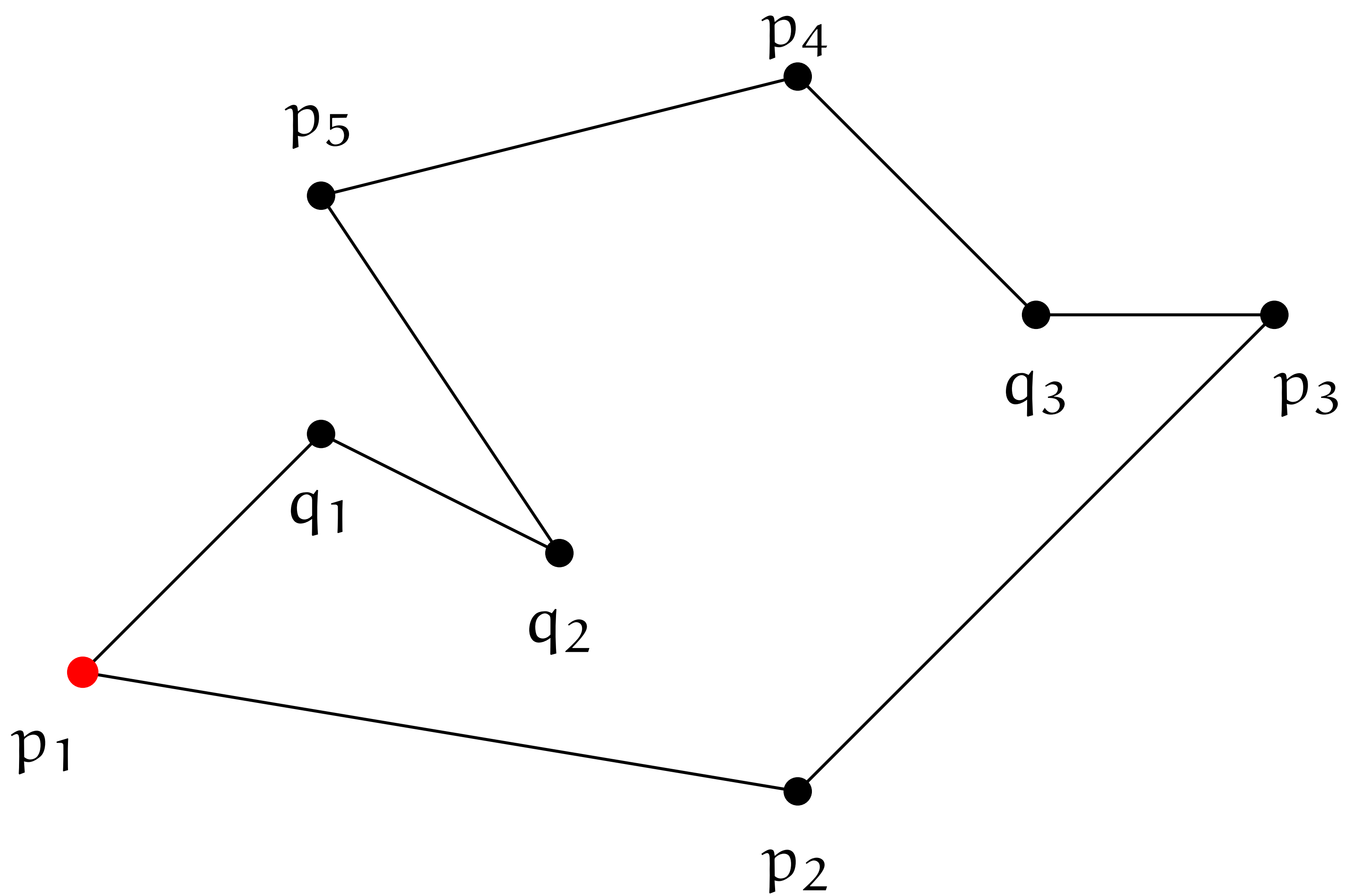


Optimal tour among those which respect the cyclic order and the order “2–3–1.”

Example

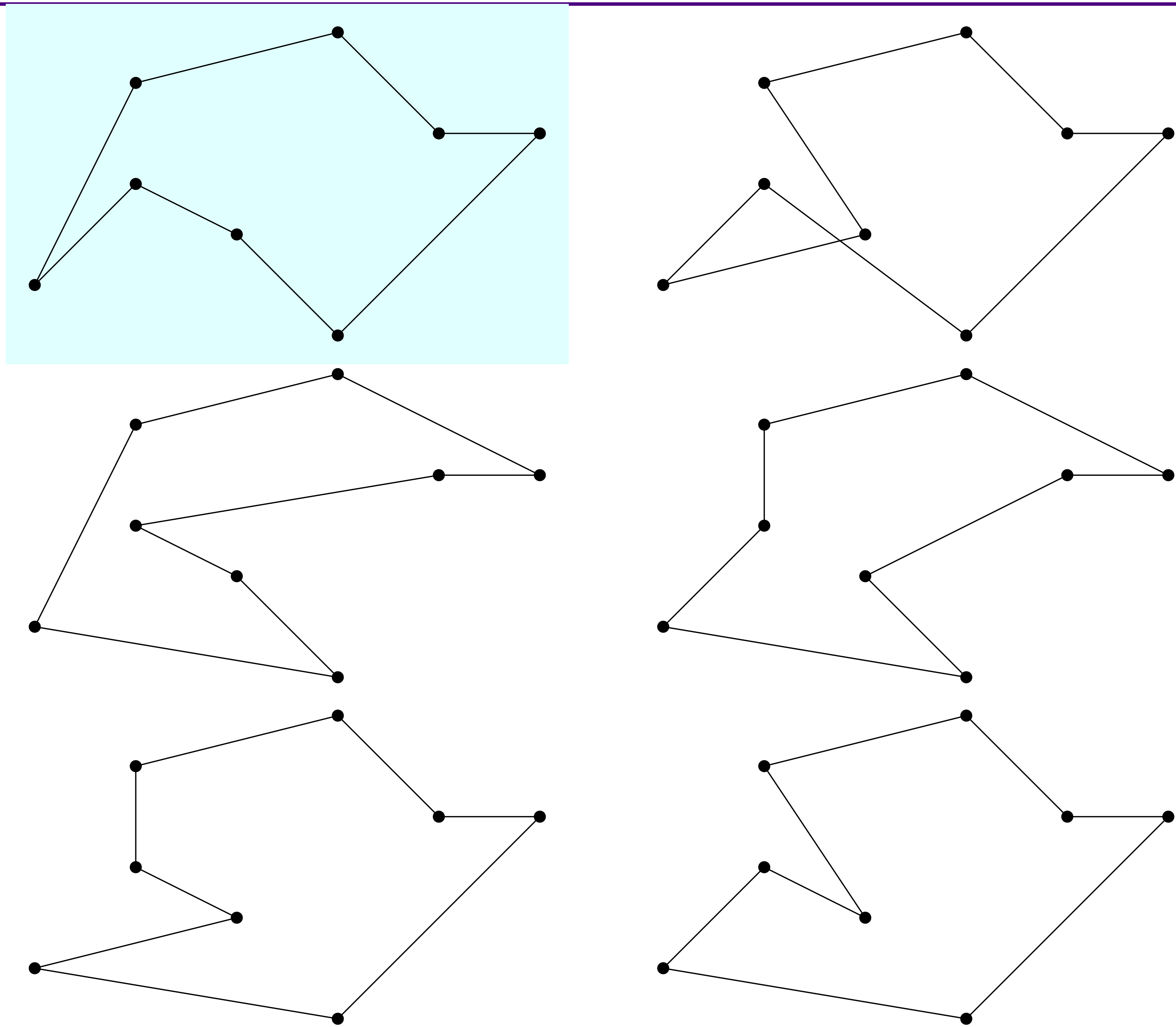


Optimal tour among those which respect the cyclic order and the order “3–1–2.”



Optimal tour among those which respect the cyclic order and the order “3–2–1.”

Choose the best one



Outline of the second algorithm

(1) Distinguish the inner points and the non-inner points;

(2) Fix a cyclic order on the non-inner points;

(3) For each linear order on the inner points

- **(i)** Compute an optimal tour among those which respect these two orders;

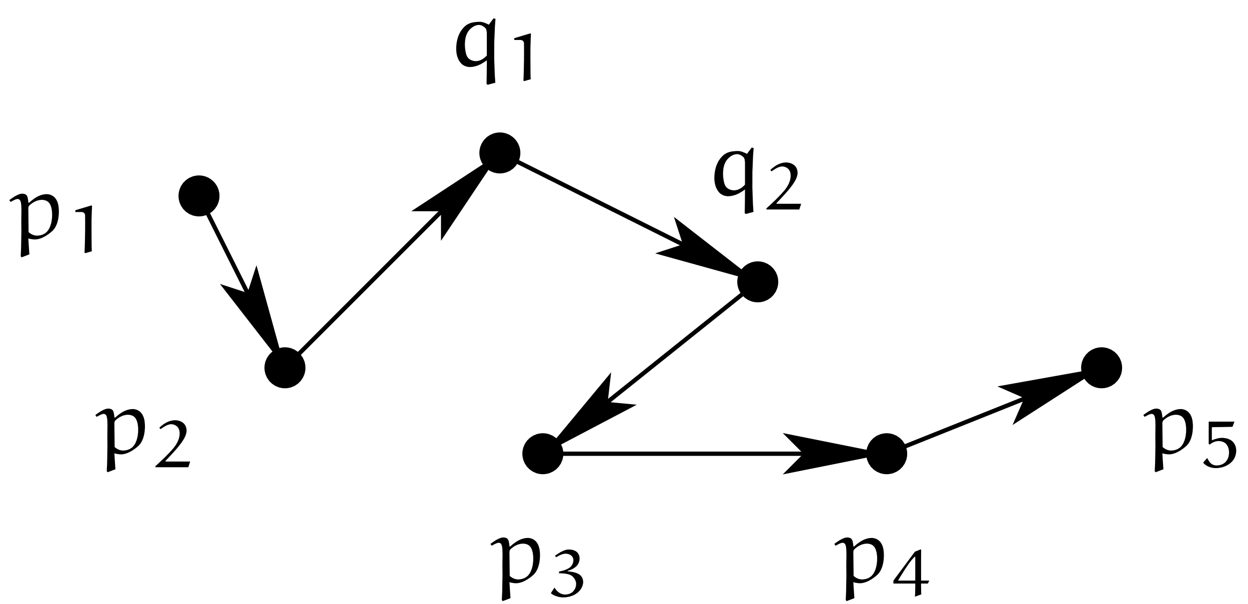
(4) Choose the best one among them.

Not yet clear: How to do Step (3i)??

$p_1, \ldots, p_{n-k}$ a cycl. order on the non-inner pts

$q_1, \ldots, q_k$ a linear order on the inner pts

$F(\underline{i}, j) :=$ the length of a shortest path from p_1 to p_i via $p_1, \ldots, p_i$ and $q_1, \ldots, q_j$ which respects these two orders

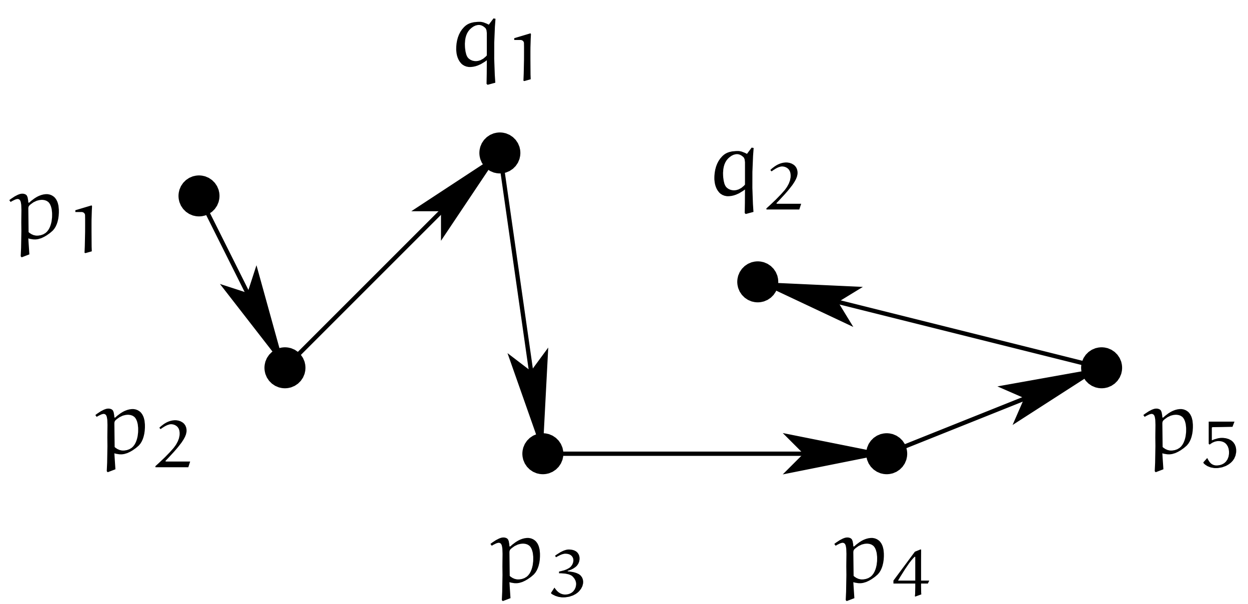


$(i = 5,\ j = 2)$

$p_1, \ldots, p_{n-k}$ a cycl. order on the non-inner pts

$q_1, \ldots, q_k$ a linear order on the inner pts

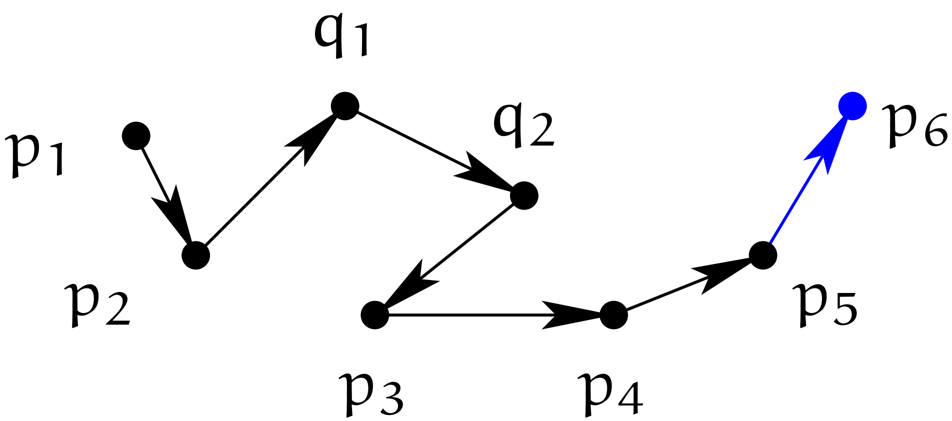
$F(i, \underline{j}) :=$ the length of a shortest path from p_1 to q_j via $p_1, \ldots, p_i$ and $q_1, \ldots, q_j$ which respects these two orders

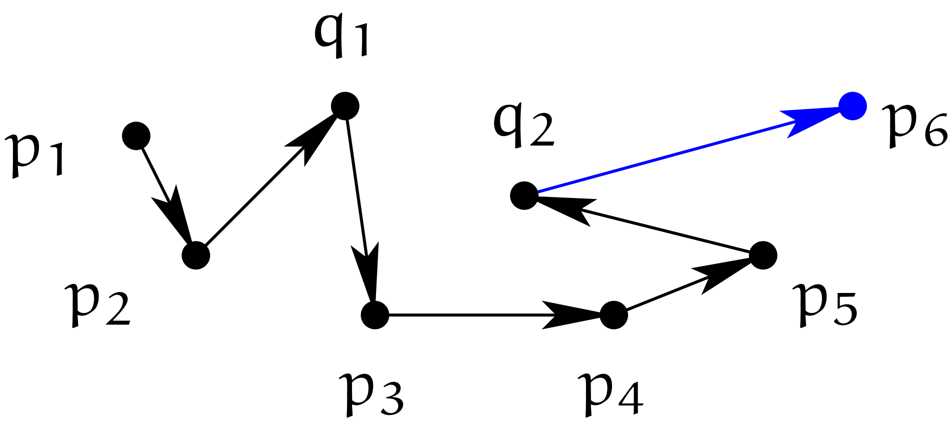


$(i = 5,\ j = 2)$

It holds that

$$F(\underline{i+1}, j) \;=\; \begin{array}{l} \text{minimum of} \\ F(\underline{i}, j) + d(p_i, p_{i+1}) \text{ and} \\ F(i, \underline{j}) + d(q_j, p_{i+1}). \end{array}$$





$(i = 5,\ j = 2)$

Main recurrence

It holds that

$$
\begin{aligned}
F(\underline{i+1}, j) \quad = \quad & \text{minimum of} \\
& F(\underline{i}, j) + d(p_i, p_{i+1}) \text{ and} \\
& F(i, \underline{j}) + d(q_j, p_{i+1}). \\
F(i, \underline{j+1}) \quad = \quad & \text{minimum of} \\
& F(\underline{i}, j) + d(p_i, q_{j+1}) \text{ and} \\
& F(i, \underline{j}) + d(q_j, q_{j+1}).
\end{aligned}
$$

- By the dynamic programming technique, $F(\underline{n-k}, k)$ and $F(n-k, \underline{k})$ can be computed in $O(kn)$ time.

- The length of a shortest tour which respects these two orders is the minimum of $F(\underline{n-k}, k) + d(p_{n-k}, p_1)$ and $F(n-k, \underline{k}) + d(q_k, p_1)$.

- By the dynamic programming technique,
  $F(\underline{n-k}, k)$ and $F(n-k, \underline{k})$ can be computed in $O(kn)$ time.

- The length of a shortest tour which respects these two orders is
  the minimum of
  $F(\underline{n-k}, k) + d(p_{n-k}, p_1)$ and
  $F(n-k, \underline{k}) + d(q_k, p_1)$.

Outline of the second algorithm

(1) Distinguish the inner points and the non-inner points;

(2) Fix a cyclic order on the non-inner points;

(3) For each linear order on the inner points

   (i) Compute an optimal tour among those which respect these two orders;

(4) Choose the best one among them.

What remains: the analysis of the running time

Running time

There are k inner points.

- $\#$ of linear orders on k points $= k!$.
- They can be enumerated in $O(1)$ time per order.
- The length of an optimal tour which respects the two orders can be computed in $O(kn)$ time.

The running time $= \underbrace{O(n \log n)}_{\text{convex hull computation}} + O(k!kn)$.

When $k = O(\log n / \log\log n)$, this is poly. in n.

There are k inner points.

- $\#$ of linear orders on k points $= k!$.
- They can be enumerated in $O(1)$ time per order.
- The length of an optimal tour which respects the two orders can be computed in $O(kn)$ time.

The running time $= \underbrace{O(n \log n)}_{\text{convex hull computation}} + O(k!kn)$.

When $k = O(\log n / \log \log n)$, this is poly. in n.

There are k inner points.

- $\#$ of linear orders on k points $= k!$.
- They can be enumerated in $O(1)$ time per order.
- The length of an optimal tour which respects the two orders can be computed in $O(kn)$ time.

The running time $= \underbrace{O(n\log n)}_{\text{convex hull computation}} + O(k!kn)$.

When $k = O(\log n/\log\log n)$, this is poly. in n.

Running time

There are k inner points.

- $\#$ of linear orders on k points $= k!$.
- They can be enumerated in $O(1)$ time per order.
- The length of an optimal tour which respects the two orders can be computed in $O(kn)$ time.

The running time $= \underbrace{O(n \log n)}_{\text{convex hull computation}} + O(k!kn)$.

When $k = O(\log n / \log \log n)$, this is poly. in n.

There are k inner points.

- $\#$ of linear orders on k points $= k!$.
- They can be enumerated in $O(1)$ time per order.
- The length of an optimal tour which respects the two orders can be computed in $O(kn)$ time.

The running time $= \underbrace{O(n \log n)}_{\text{convex hull computation}} + O(k!kn)$.

When $k = O(\log n / \log \log n)$, this is poly. in n.

There are k inner points.

- $\#$ of linear orders on k points $= k!$.
- They can be enumerated in $O(1)$ time per order.
- The length of an optimal tour which respects the two orders can be computed in $O(kn)$ time.

The running time $= \underbrace{O(n \log n)}_{\text{convex hull computation}} + O(k!kn)$.

When $k = O(\log n / \log \log n)$, this is poly. in n.

Result

We gave two simple algorithms.

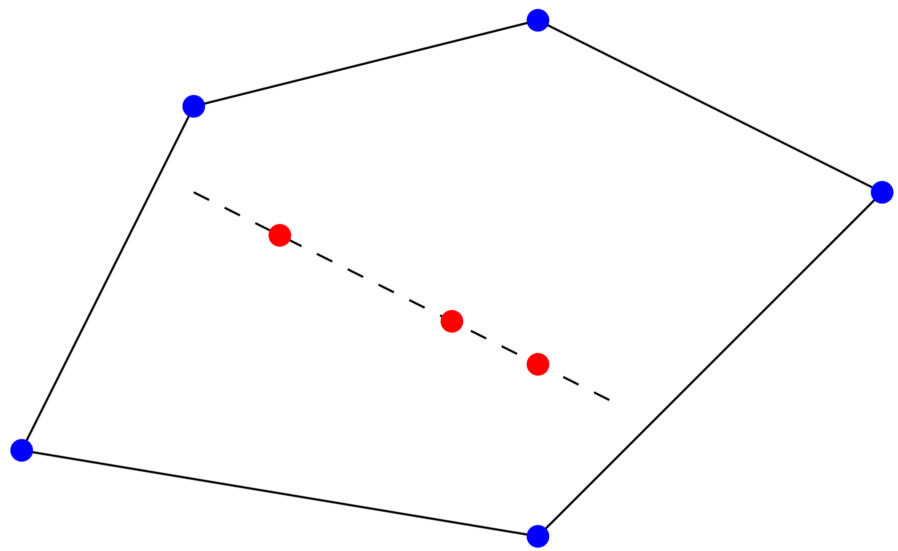
$n :=$ the total number of points
$k :=$ the number of inner points

- First algorithm runs in $O(k!n^{k+1})$ time which is poly. when $k = O(1)$.
- Second algorithm runs in $O(k!kn)$ time which is poly. when $k = O(\log n / \log\log n)$.

Open problem: Improve the bound!

Thm (Deĭneko, van Dal & Rote '96)

2DTSP can be solved in $O(kn)$ time when the inner points lie on a line.



Our work $\left\{ \begin{array}{l} \text{deals with the most general case.} \\ \text{still runs in linear time in } n. \end{array} \right.$

The same strategy works for other problems.

- The prize-collecting TSP
- The partial TSP

Result

The 2D versions of these problems
with k inner points
can be solved in polynomial time
when $k = O(\log n / \log\log n)$.