Motion Planning

O’Rourke, Chapter 8

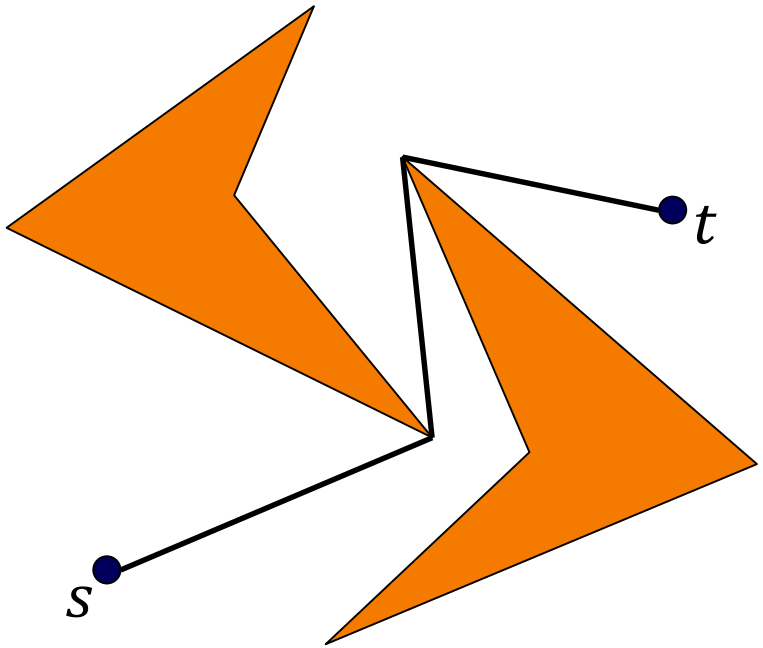
Outline

- Translating a polygon
- Moving a ladder

Shortest Path (Point-to-Point)

Goal:

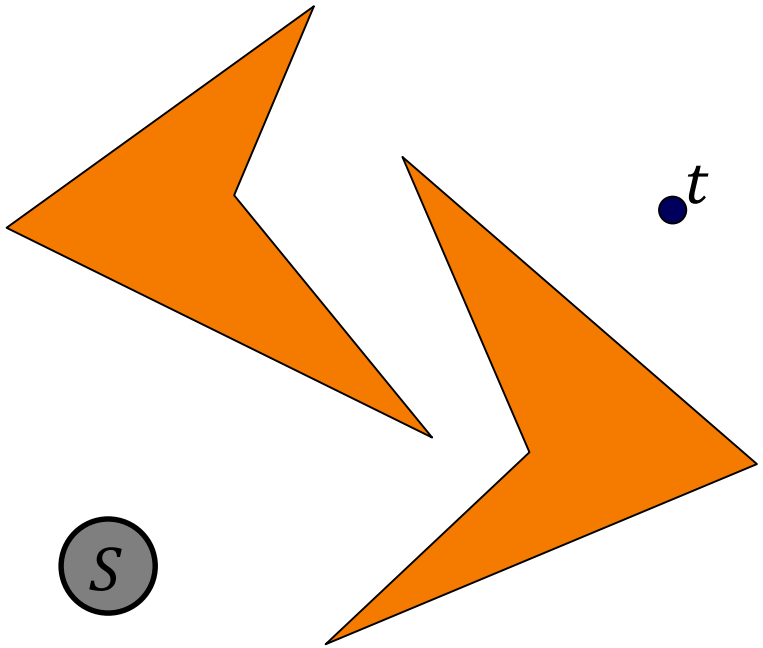
Given disjoint polygons in the plane, and given positions s and t, find the **shortest** path from s to t that avoids the polygons.



Any Path (Shape-to-Point)

Goal:

Given disjoint polygons in the plane, and given a shape S and a position t, determine if there is **any** path taking S to t avoiding the polygons.



Any Path (Shape-to-Point)

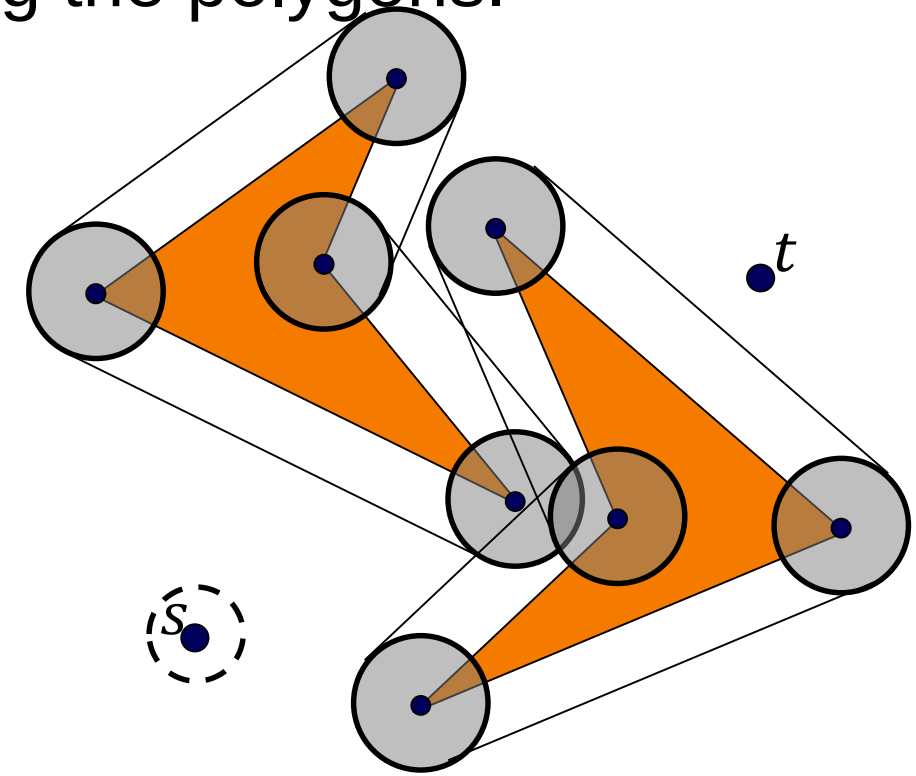
Goal:

Given disjoint polygons in the plane, and given a shape S and a position t, determine if there is **any** path taking S to t avoiding the polygons.

Approach:

Dilate the polygons by S to transform this into a point-to-point problem.

If s can reach t while avoiding the dilated polygons, S can reach t.



Any Path (Shape-to-Point)

Goal:

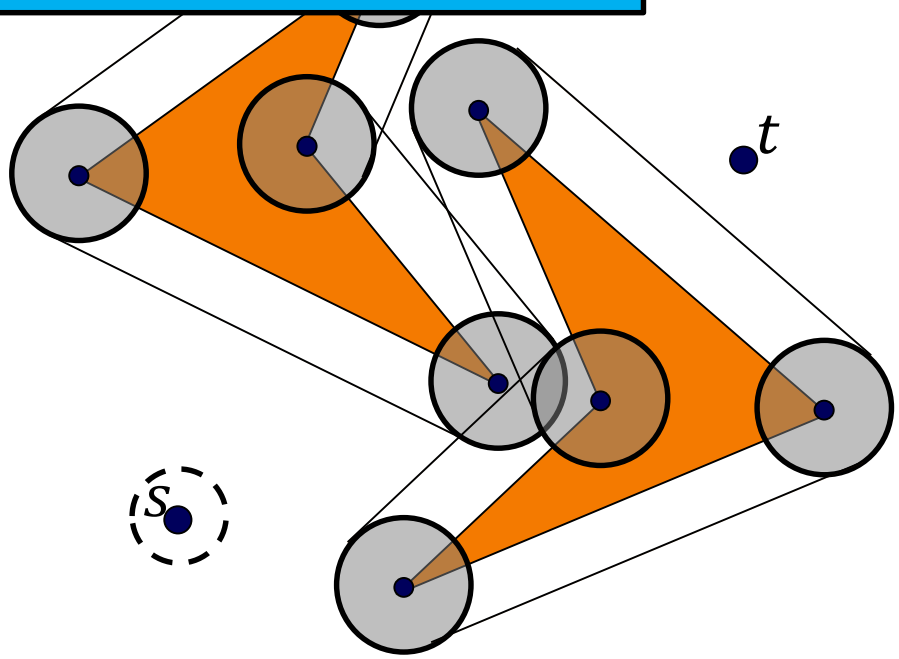
Given[illegible]en a shape[illegible]s **any** path t[illegible]

Appro[illegible]

Note that in this case:
- S is symmetric.
- s is the center of S.
- The traced out boundary self-intersects.

Dilate the polygons by S to transform this into a point-to-point problem.

If s can reach t while avoiding the dilated polygons, S can reach t.



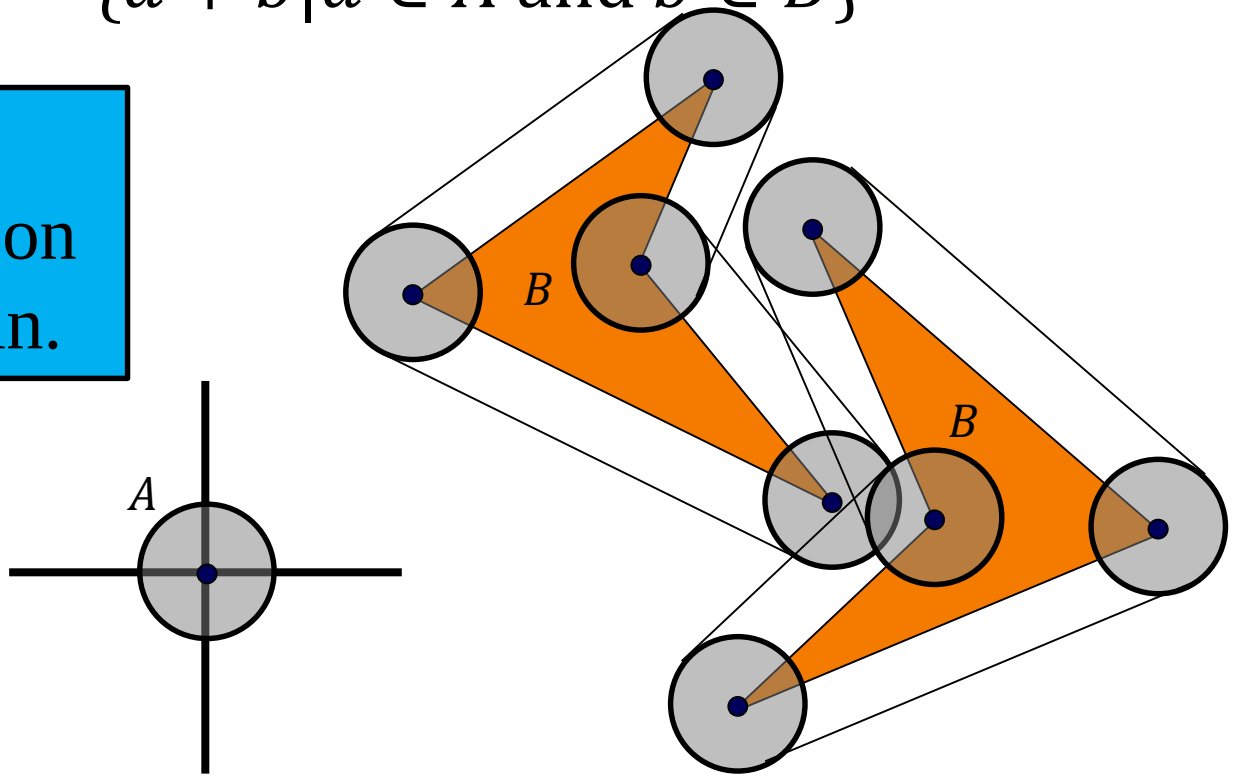
Minkowski Sums

Definition:

Given two point-sets in the plane, A and B, the *Minkowski Sum* is the set of points:

$$A \oplus B = \{a + b | a \in A \text{ and } b \in B\}$$

Note:
The sum depends on the choice of origin.

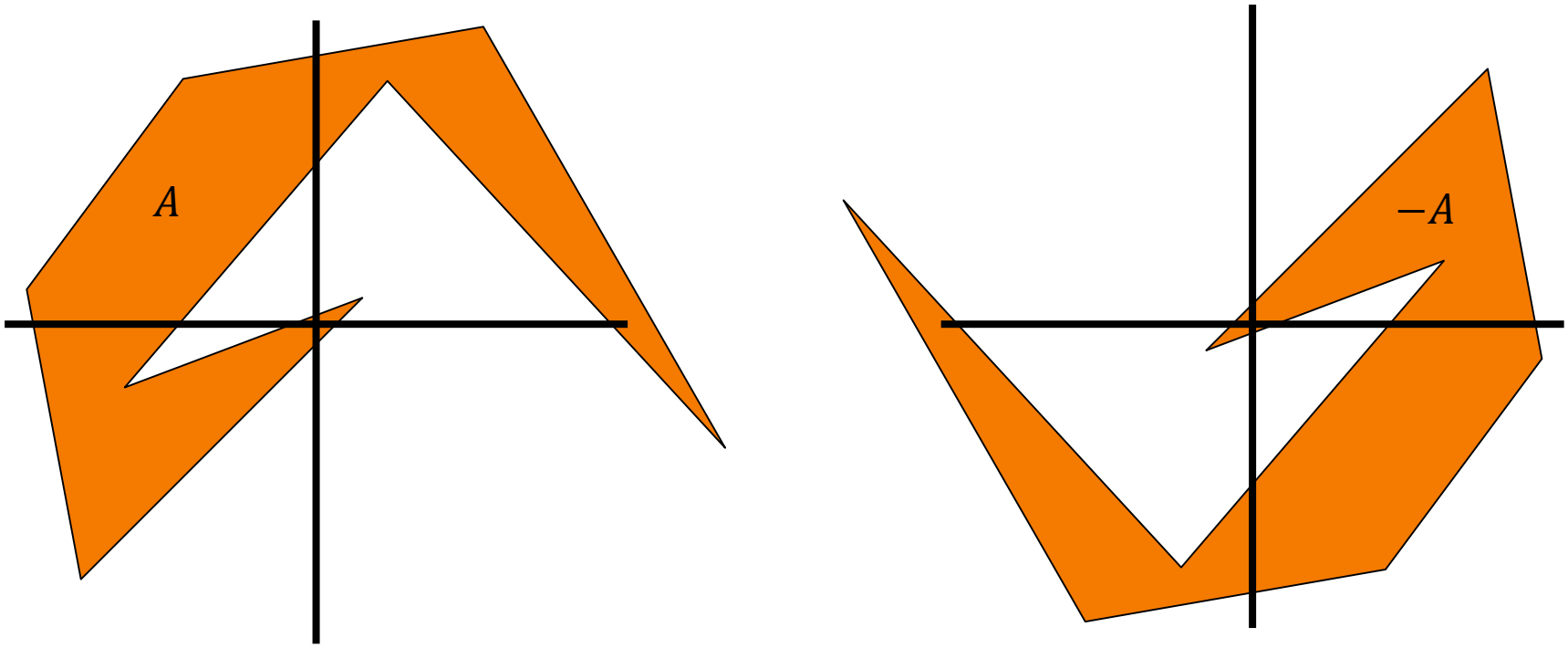


Minkowski Sums

Definition:

Given a point-set A in the plane, its *negation* is the set of points:

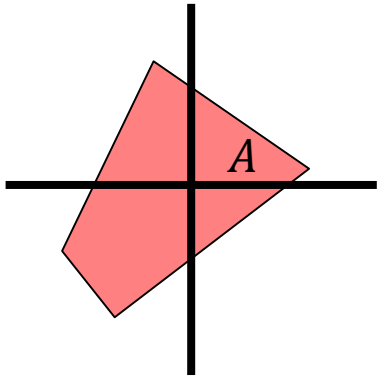
$$-A = \{-a | a \in A\}$$

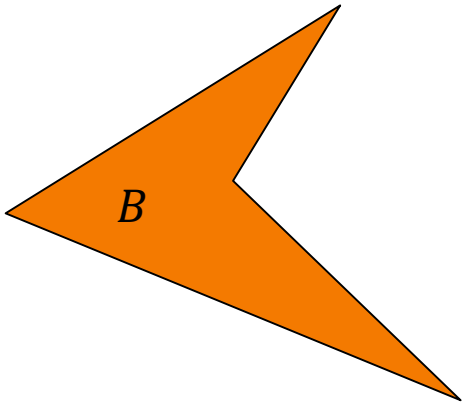


Motion Planning

Claim:

Given shapes A and B, the translation of A by τ intersects B if and only if $\tau \in B \oplus (-A)$.

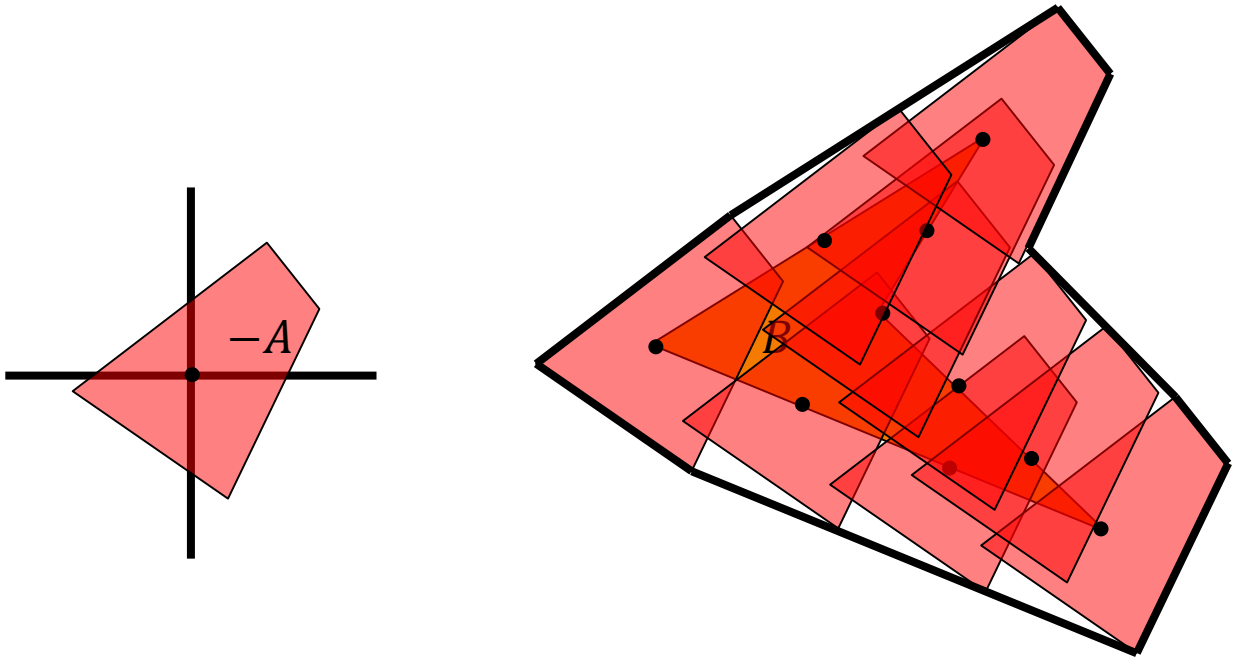




Motion Planning

Claim:

Given shapes A and B, the translation of A by τ intersects B if and only if $\tau \in B \oplus (-A)$.



Motion Planning

Claim:

Given shapes A and B, the translation of A by τ intersects B if and only if $\tau \in B \oplus (-A)$.

Proof:

The translation of A by τ intersects B.

$\Leftrightarrow \exists a \in A, b \in B$ such that $a + \tau = b$.

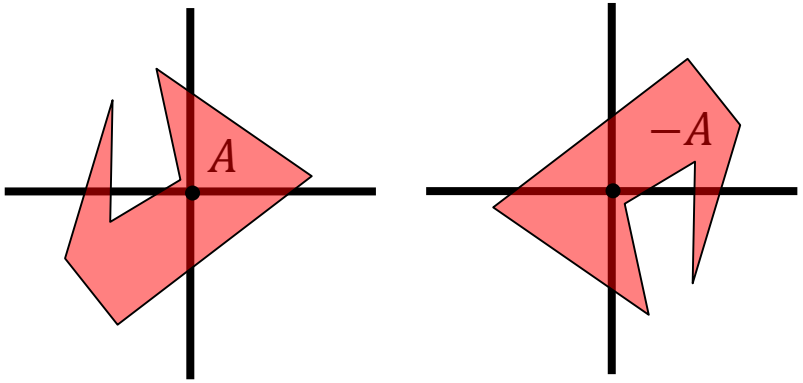
$\Leftrightarrow \exists a \in A, b \in B$ such that $\tau = b - \mathrm{a}$.

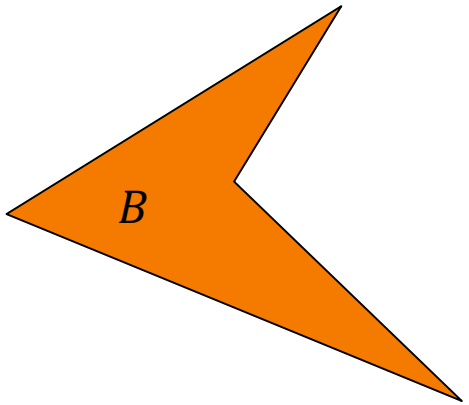
$\Leftrightarrow \tau \in B \oplus (-A)$.

Motion Planning

MinkowskiTrace(A , B)

- Place $-A$ at a vertex of $b \in B$.
- for each $a \in$ Extremal(b)
  - » for $b \in B$:
    - Trace a along edge $(b, b+1)$
    - $a \leftarrow$ TraceNextLocalExtremum($b+1$)

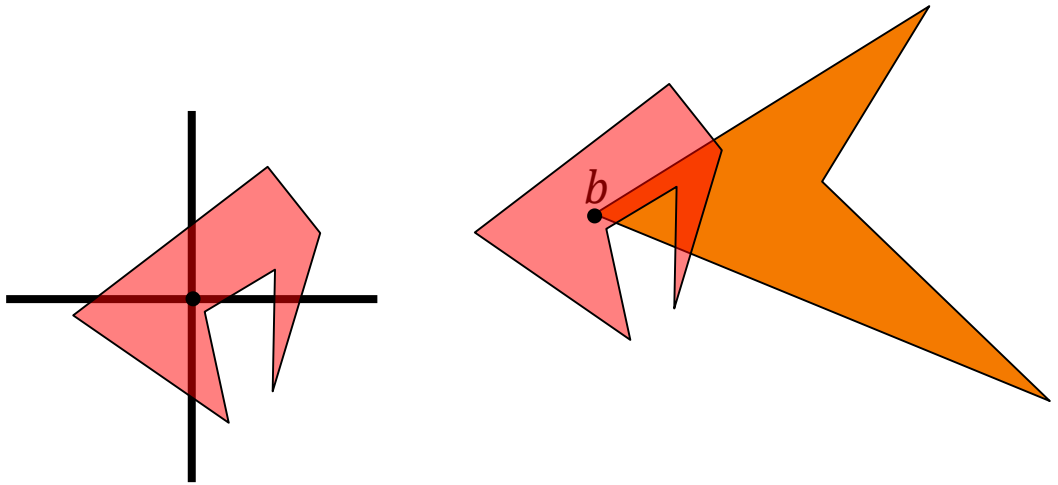




Motion Planning

MinkowskiTrace(A , B)

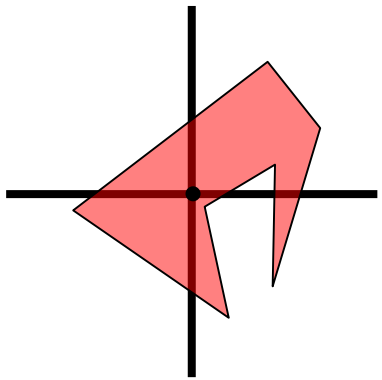
- **Place $-A$ at a vertex of $b \in B$.**
- for each $a \in$ Extremal(b)
  - » for $b \in B$:
    - Trace a along edge $(b, b+1)$
    - $a \leftarrow$ TraceNextLocalExtremum($b+1$)

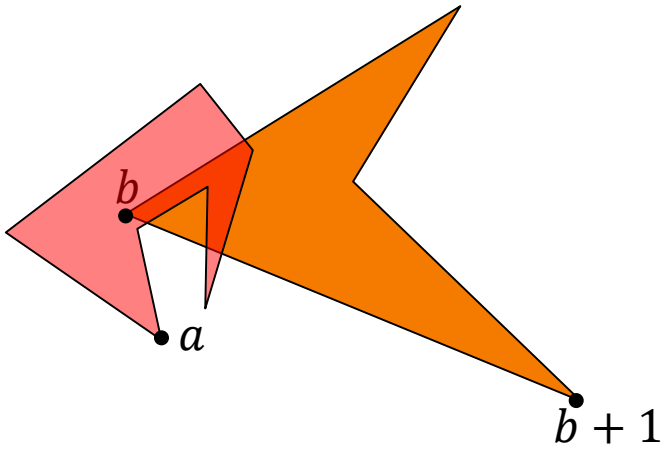


Motion Planning

MinkowskiTrace(A , B)

- Place $-A$ at a vertex of $b \in B$.
- **for each $a \in$ Extremal(b)**
  - » for $b \in B$:
    - Trace a along edge $(b, b+1)$
    - $a \leftarrow$ TraceNextLocalExtremum($b+1$)

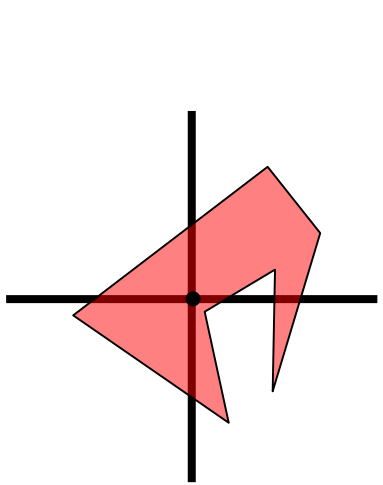


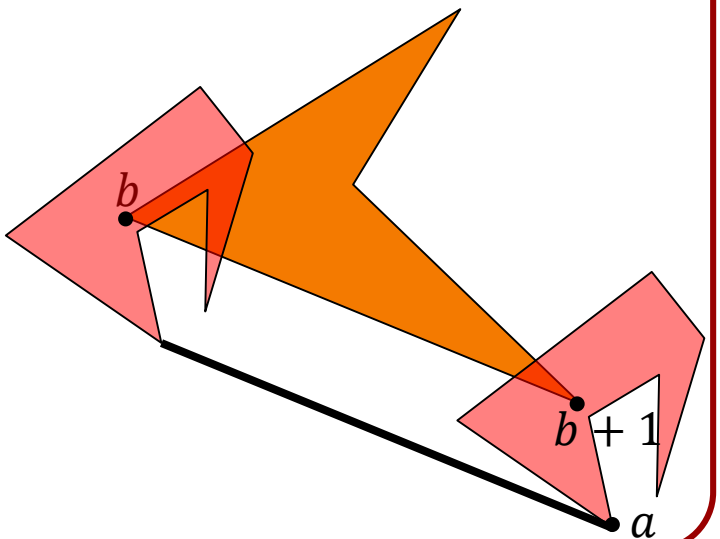


Motion Planning

MinkowskiTrace(A , B)

- Place $-A$ at a vertex of $b \in B$.
- for each $a \in$ Extremal(b)
  - » for $b \in B$:
    - **Trace a along edge $(b, b+1)$**
    - $a \leftarrow$ TraceNextLocalExtremum($b+1$)

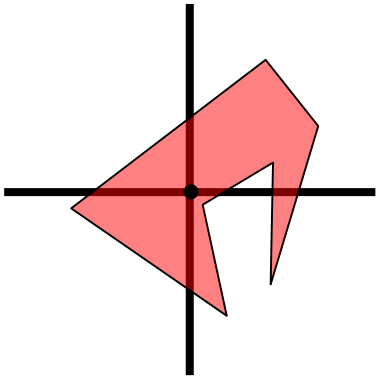


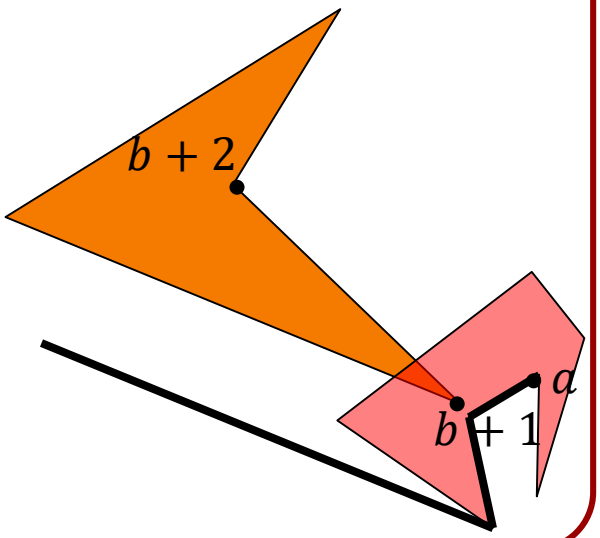


Motion Planning

MinkowskiTrace(A , B)

- Place $-A$ at a vertex of $b \in B$.
- for each $a \in$ Extremal(b)
  - » for $b \in B$:
    - Trace a along edge $(b, b+1)$
    - $a \leftarrow$ **TraceNextLocalExtremum($b+1$)**

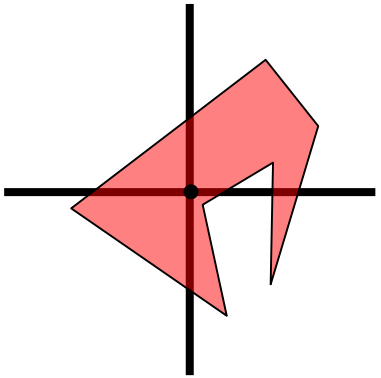


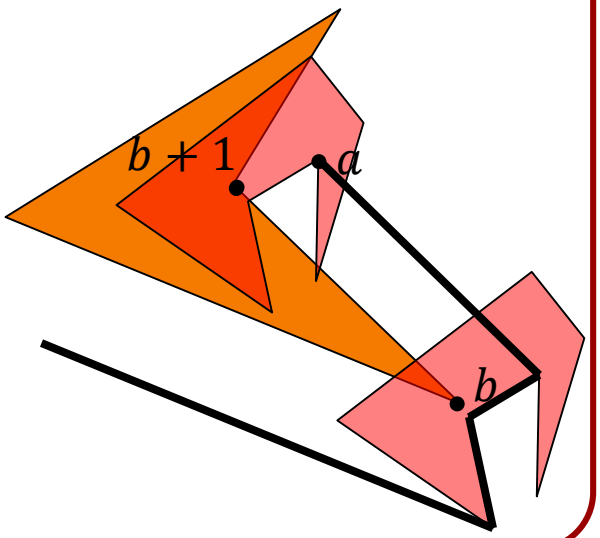


Motion Planning

MinkowskiTrace(A , B)

- Place $-A$ at a vertex of $b \in B$.
- for each $a \in$ Extremal(b)
  - » for $b \in B$:
    - **Trace a along edge $(b, b+1)$**
    - $a \leftarrow$ TraceNextLocalExtremum($b+1$)

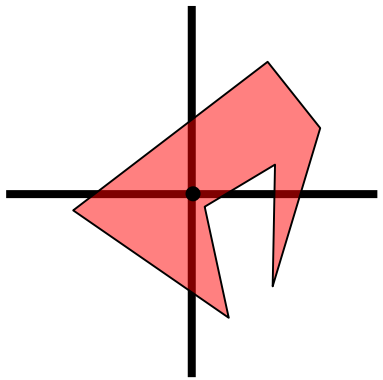


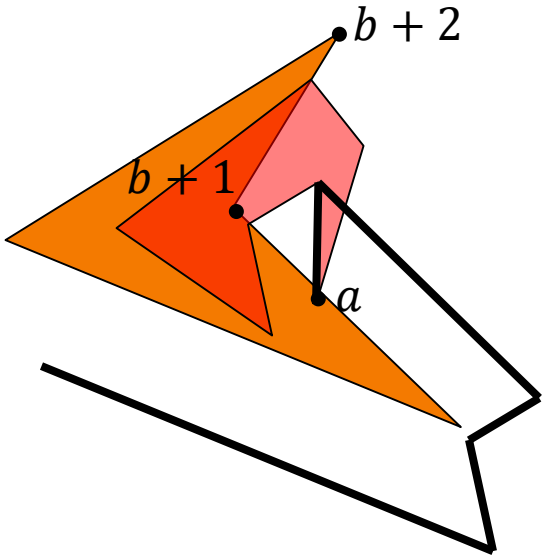


Motion Planning

MinkowskiTrace(A , B)

- Place $-A$ at a vertex of $b \in B$.
- for each $a \in$ Extremal(b)
  - » for $b \in B$:
    - Trace a along edge $(b, b+1)$
    - $a \leftarrow$ **TraceNextLocalExtremum($b+1$)**

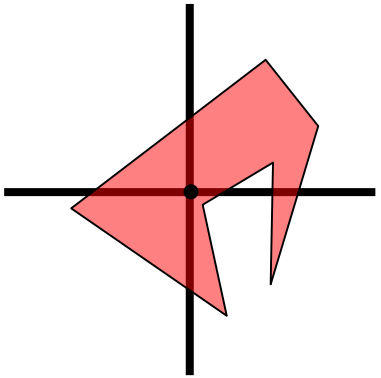


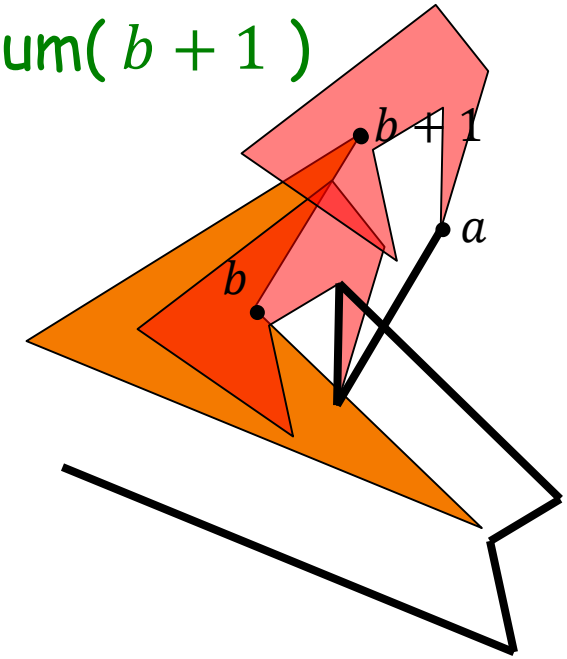


Motion Planning

MinkowskiTrace(A , B)

- Place $-A$ at a vertex of $b \in B$.
- for each $a \in$ Extremal(b)
  - » for $b \in B$:
    - **Trace a along edge $(b, b+1)$**
    - $a \leftarrow$ TraceNextLocalExtremum($b+1$)

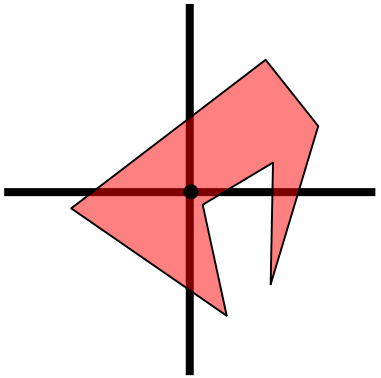


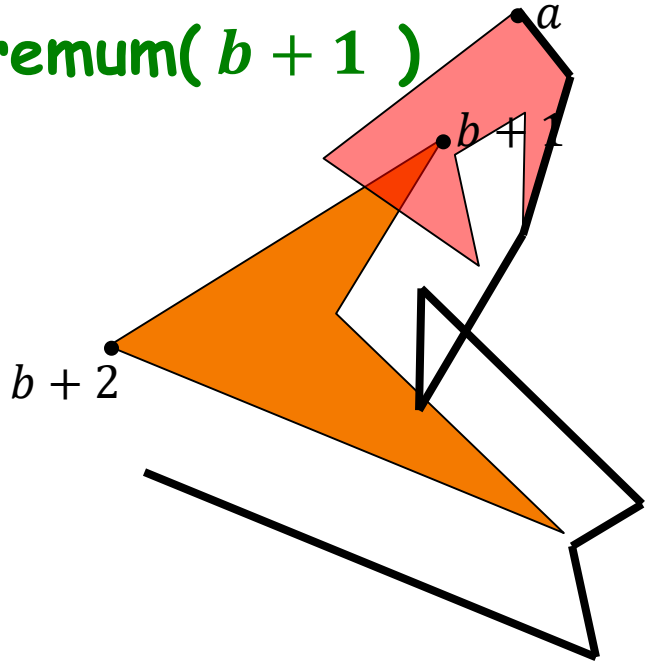


Motion Planning

MinkowskiTrace(A , B)

- Place $-A$ at a vertex of $b \in B$.
- for each $a \in$ Extremal(b)
  - » for $b \in B$:
    - Trace a along edge $(b, b+1)$
    - $a \leftarrow$ **TraceNextLocalExtremum($b+1$)**

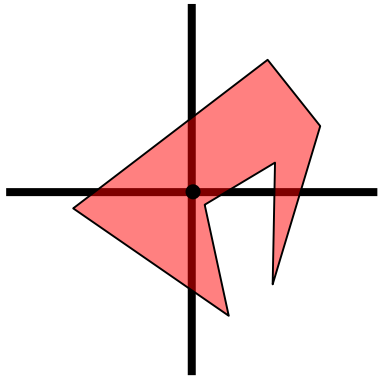


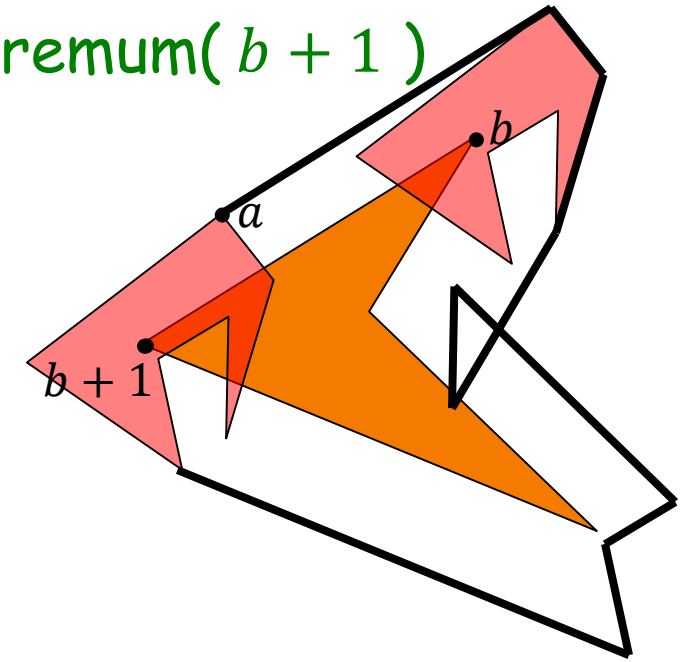


Motion Planning

MinkowskiTrace(A , B)

- Place $-A$ at a vertex of $b \in B$.
- for each $a \in$ Extremal(b)
  - » for $b \in B$:
    - **Trace a along edge $(b, b+1)$**
    - $a \leftarrow$ TraceNextLocalExtremum($b+1$)

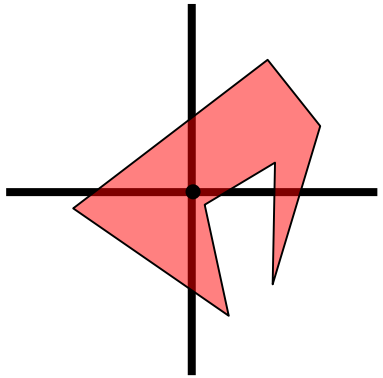


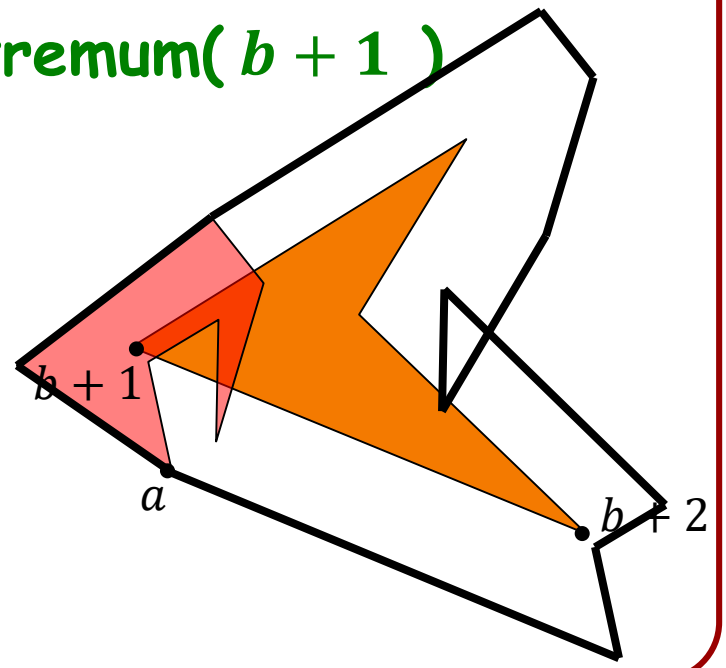


Motion Planning

MinkowskiTrace(A , B)

- Place $-A$ at a vertex of $b \in B$.
- for each $a \in$ Extremal(b)
  - » for $b \in B$:
    - Trace a along edge $(b, b+1)$
    - $a \leftarrow$ **TraceNextLocalExtremum($b+1$)**

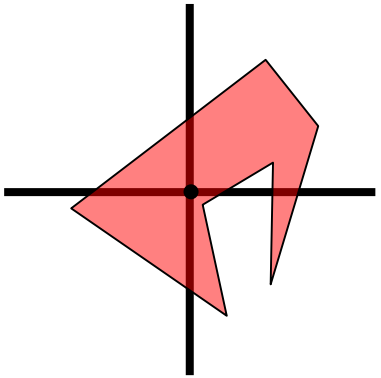


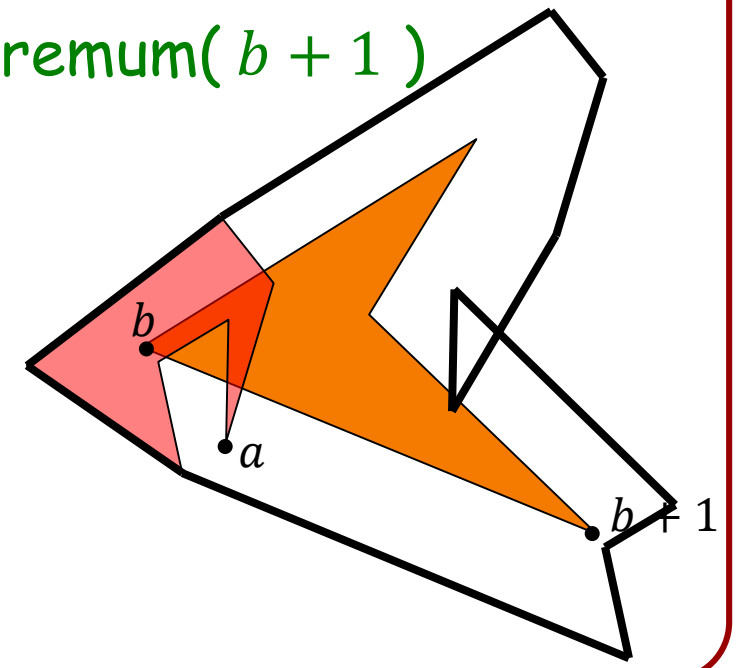


Motion Planning

MinkowskiTrace(A , B)

- Place $-A$ at a vertex of $b \in B$.
- **for each $a \in$ Extremal(b)**
  - » for $b \in B$:
    - Trace a along edge $(b, b+1)$
    - $a \leftarrow$ TraceNextLocalExtremum($b+1$)

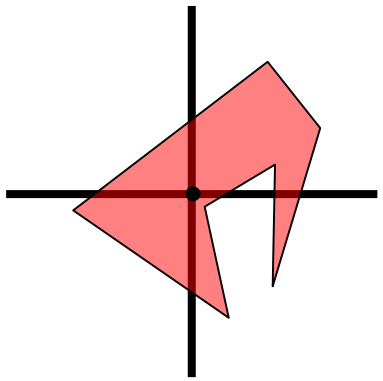


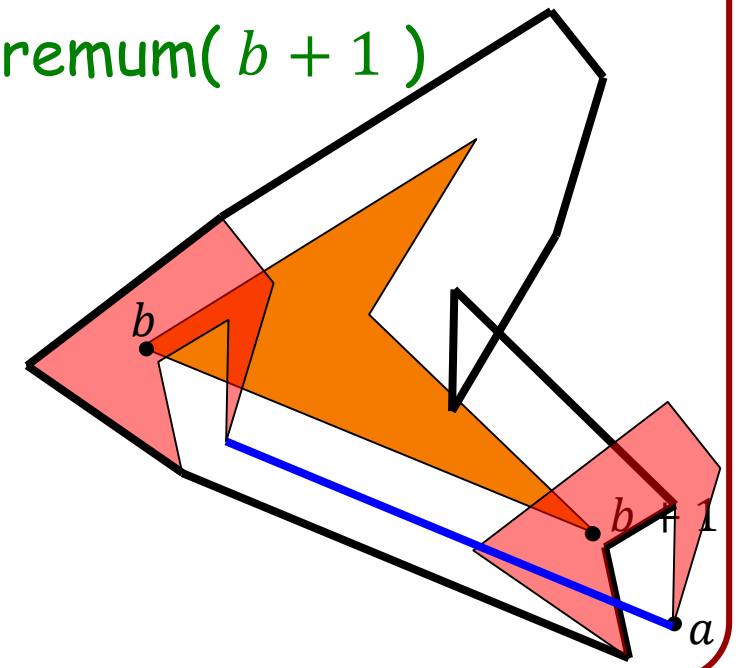


Motion Planning

MinkowskiTrace(A , B)

- Place $-A$ at a vertex of $b \in B$.
- for each $a \in$ Extremal(b)
  - » for $b \in B$:
    - **Trace a along edge $(b, b+1)$**
    - $a \leftarrow$ TraceNextLocalExtremum($b+1$)

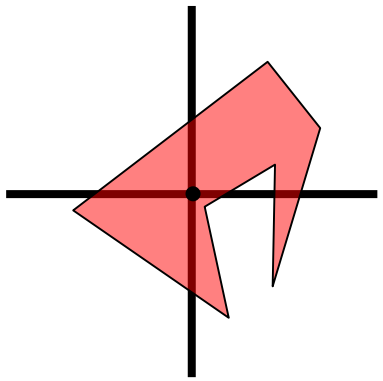


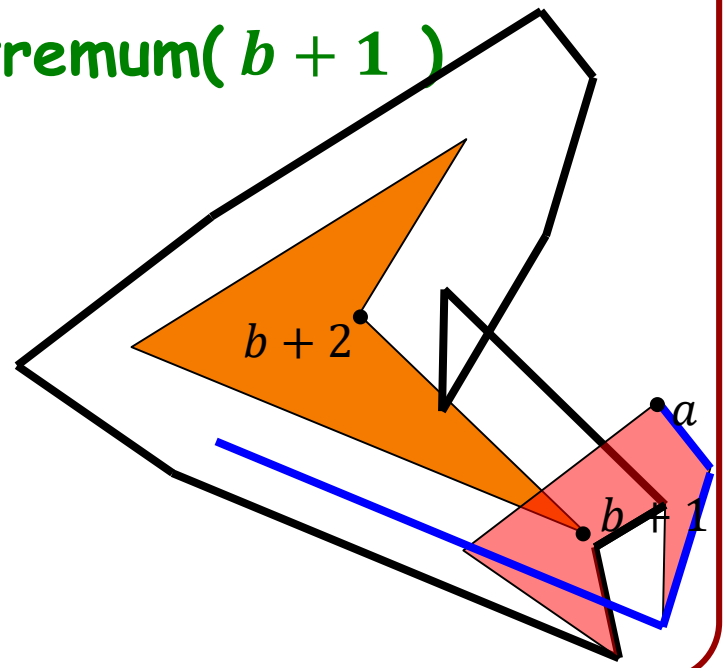


Motion Planning

MinkowskiTrace(A , B)

- Place $-A$ at a vertex of $b \in B$.
- for each $a \in$ Extremal(b)
  - » for $b \in B$:
    - Trace a along edge $(b, b+1)$
    - $a \leftarrow$ **TraceNextLocalExtremum($b+1$)**

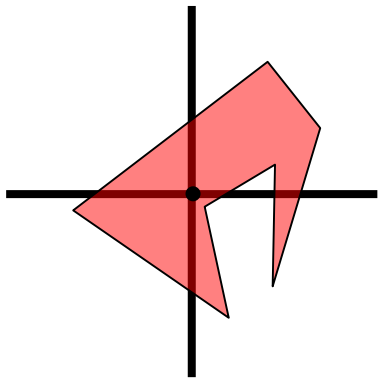


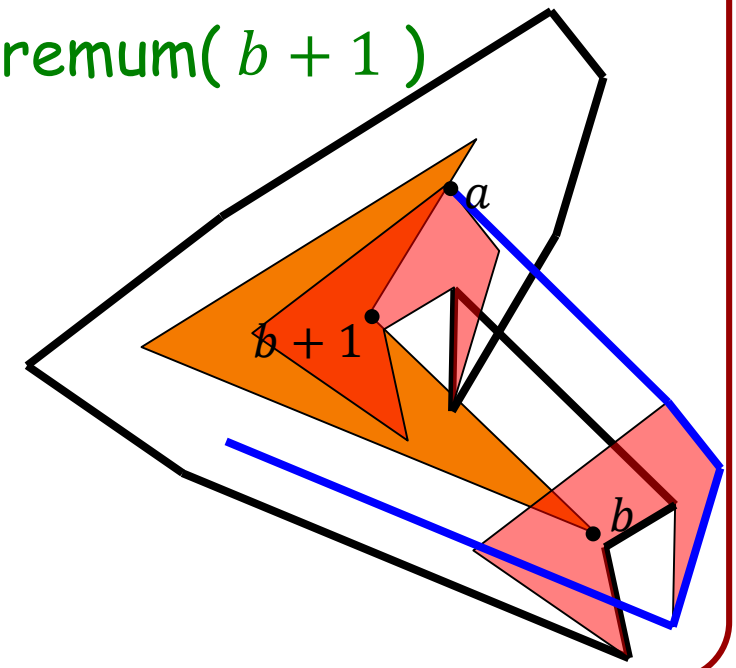


Motion Planning

MinkowskiTrace(A , B)

- Place $-A$ at a vertex of $b \in B$.
- for each $a \in$ Extremal(b)
  - » for $b \in B$:
    - **Trace a along edge $(b, b+1)$**
    - $a \leftarrow$ TraceNextLocalExtremum($b+1$)

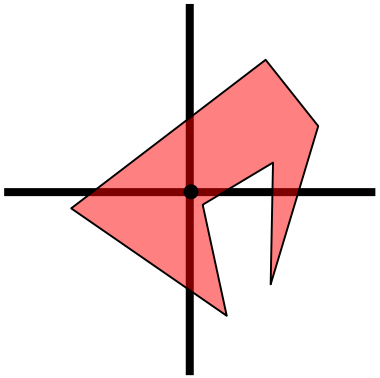


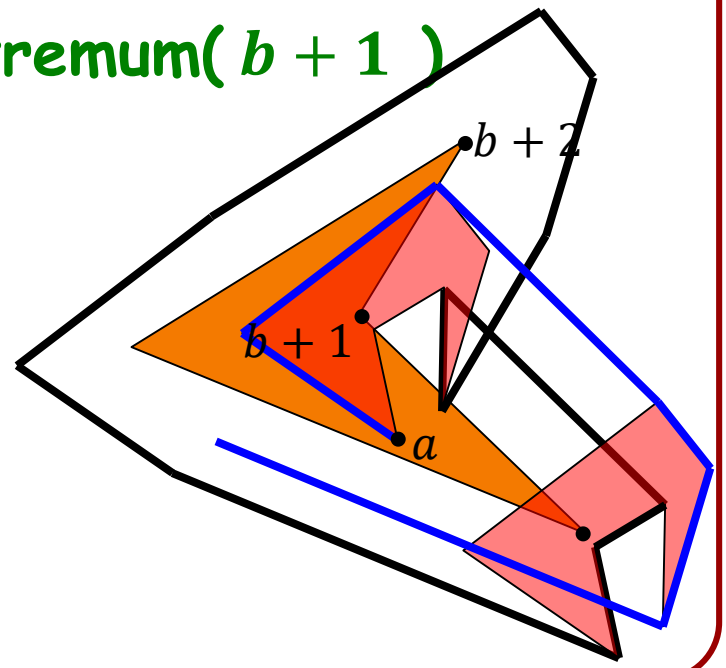


Motion Planning

MinkowskiTrace(A , B)

- Place $-A$ at a vertex of $b \in B$.
- for each $a \in$ Extremal(b)
  - » for $b \in B$:
    - Trace a along edge $(b, b+1)$
    - $a \leftarrow$ **TraceNextLocalExtremum($b+1$)**

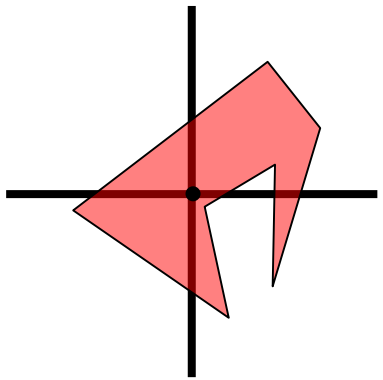


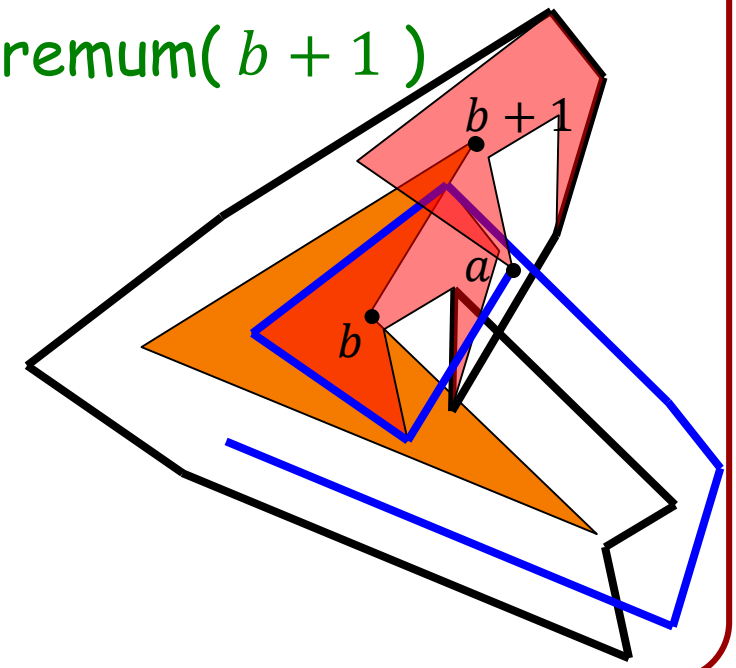


Motion Planning

MinkowskiTrace(A , B)

- Place $-A$ at a vertex of $b \in B$.
- for each $a \in$ Extremal(b)
  - » for $b \in B$:
    - **Trace a along edge $(b, b+1)$**
    - $a \leftarrow$ TraceNextLocalExtremum($b+1$)

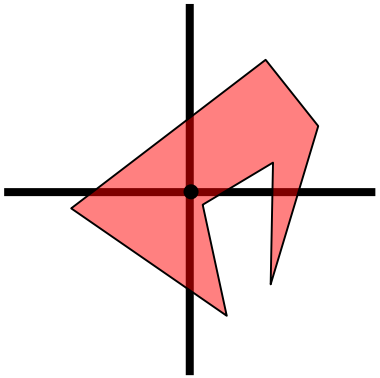


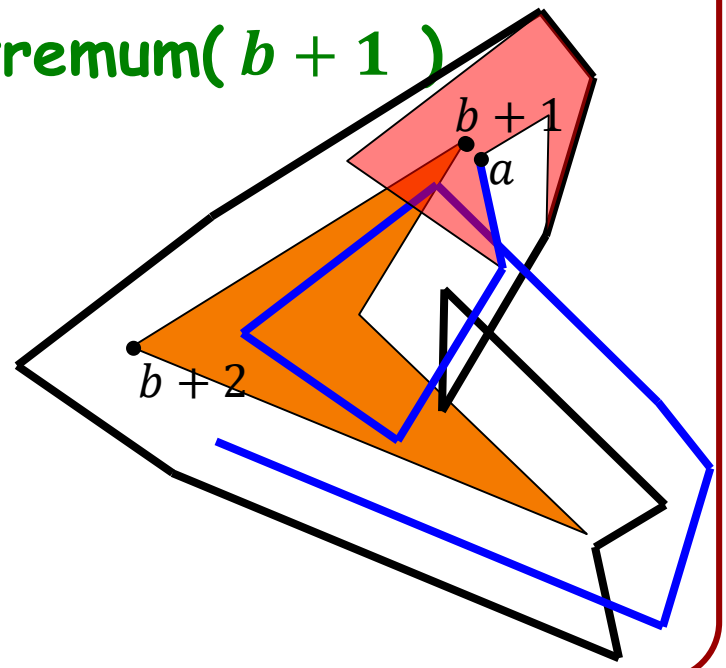


Motion Planning

MinkowskiTrace(A , B)

- Place $-A$ at a vertex of $b \in B$.
- for each $a \in$ Extremal(b)
  - » for $b \in B$:
    - Trace a along edge $(b, b+1)$
    - $a \leftarrow$ **TraceNextLocalExtremum($b+1$)**

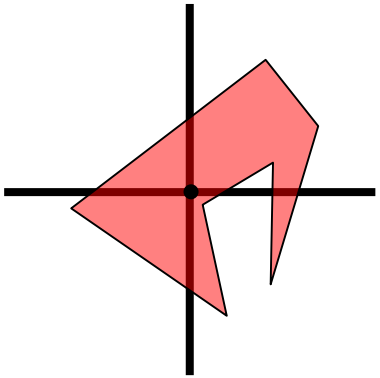


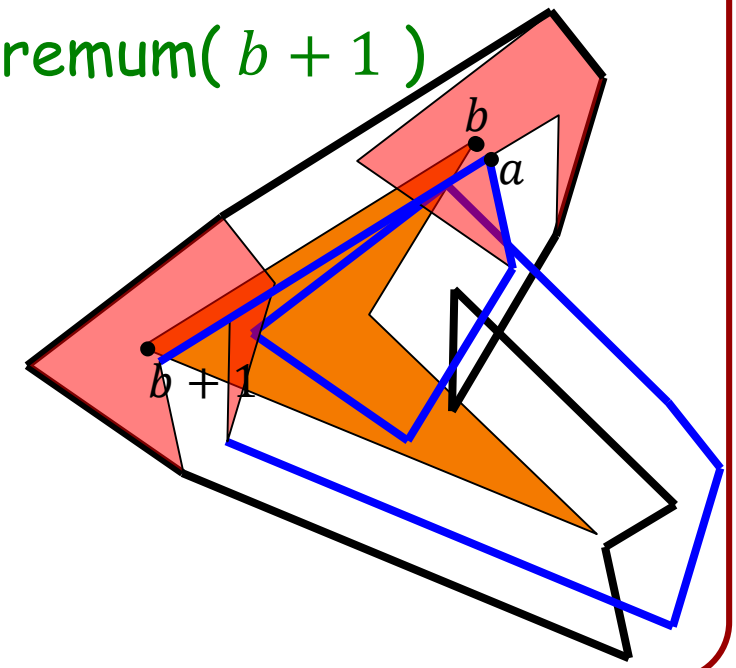


Motion Planning

MinkowskiTrace(A , B)

- Place $-A$ at a vertex of $b \in B$.
- for each $a \in$ Extremal(b)
  - » for $b \in B$:
    - **Trace a along edge $(b, b+1)$**
    - $a \leftarrow$ TraceNextLocalExtremum($b+1$)

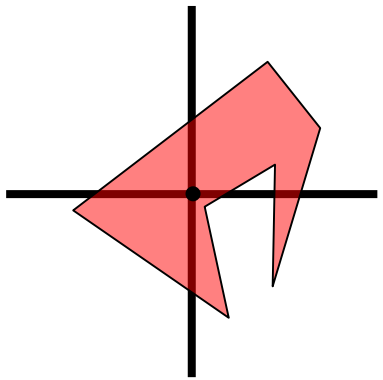


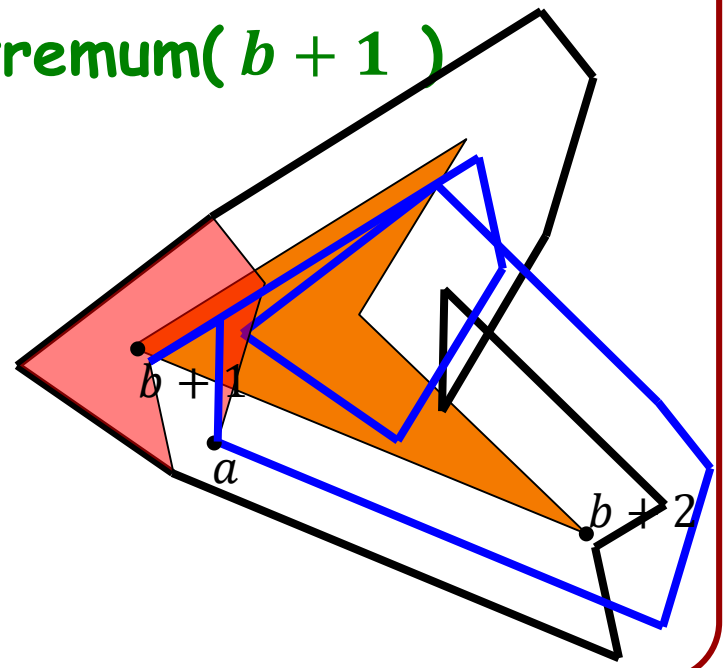


Motion Planning

MinkowskiTrace(A , B)

- Place $-A$ at a vertex of $b \in B$.
- for each $a \in$ Extremal(b)
  - » for $b \in B$:
    - Trace a along edge $(b, b+1)$
    - $a \leftarrow$ **TraceNextLocalExtremum($b+1$)**

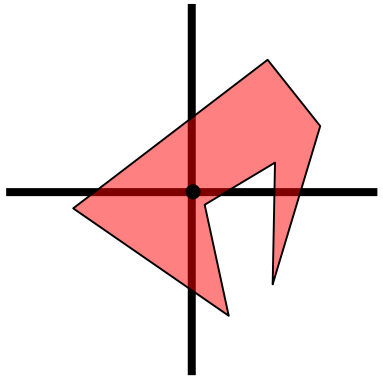


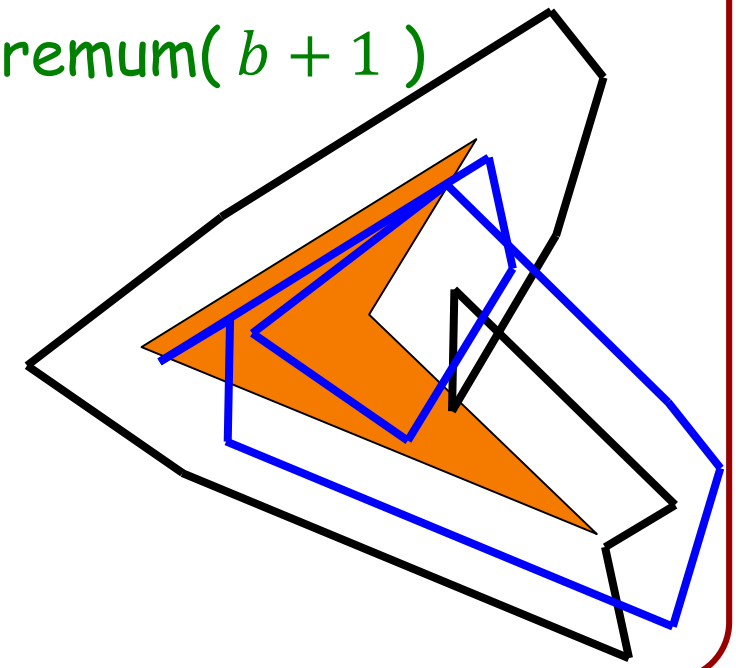


Motion Planning

MinkowskiTrace(A , B)

- Place $-A$ at a vertex of $b \in B$.
- for each $a \in$ Extremal(b)
  - » for $b \in B$:
    - Trace a along edge $(b, b+1)$
    - $a \leftarrow$ TraceNextLocalExtremum($b+1$)





Motion Planning

MinkowskiTrace(A , B)

- Place $-A$ at a vertex of $b \in B$.
- for each $a \in$ Extremal(b)
  - » for $b \in B$:
    - Trace a along edge $(b, b+1)$
    - $a \leftarrow$ TraceNextLocalExtremum($b+1$)

Note:

- If A is convex, the for loop cycles around the polygon B once.
- Otherwise we can cycle as many times as there are different extremal vertices in A -- $O(|A|)$.

Motion Planning

MinkowskiTrace(A , B)

- Place $-A$ at a vertex of $b \in B$.
- for each $a \in$ Extremal(b)
  - » for $b \in B$:
    - Trace a along edge $(b, b+1)$
    - $a \leftarrow$ TraceNextLocalExtremum($b+1$)

Note:

- In the worst case, we can trace the next extremal vertex in $O(|A|)$ time.
- If A and B are convex, then in all these tracings of, we cycle around A once -- $O(1)$ time.

Motion Planning

MinkowskiTrace(A , B)

- Place $-A$ at a vertex of $b \in B$.
- for each $a \in$ Extremal(b)
  - » for $b \in B$:
    - Trace a along edge $(b, b+1)$
    - $a \leftarrow$ TraceNextLocalExtremum($b+1$)

Complexity:

- If A and B convex: $O(|A| + |B|)$
- Else if A convex: $O(|A| \cdot |B|)$
- Else: $O(|A|^2 \cdot |B|)$ [I think.]

Motion Planning

MinkowskiTrace(A , B)

- Place $-A$ at a vertex of $b \in B$.
- for each $a \in$ Extremal(b)
  - » for $b \in B$:

If A or B is not convex, to get the Minkowski Sum, we need to remove the "internal" polygons.

We can do this, e.g., by constructing a trapezoidalization, considering the dual graph, and using the winding-number/parity-test to mark the external cells.

- Linking the dual cells that are external, we need to determine if s and t are in the same connected component.

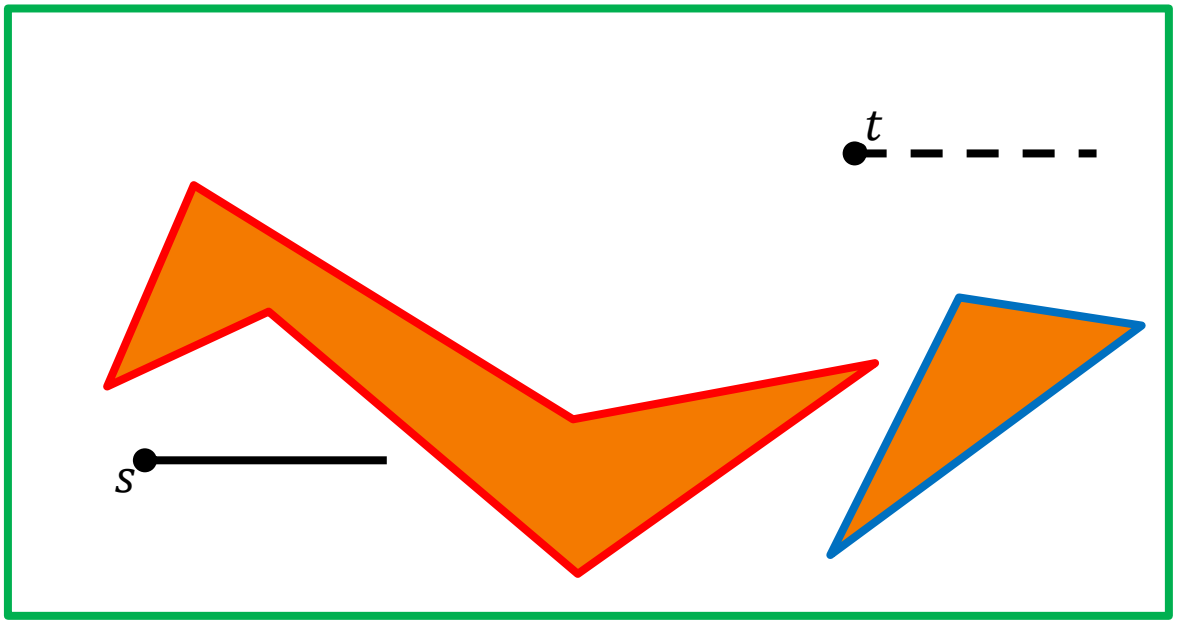
Outline

- Translating a polygon
- Moving a ladder

Moving a Ladder

Goal:

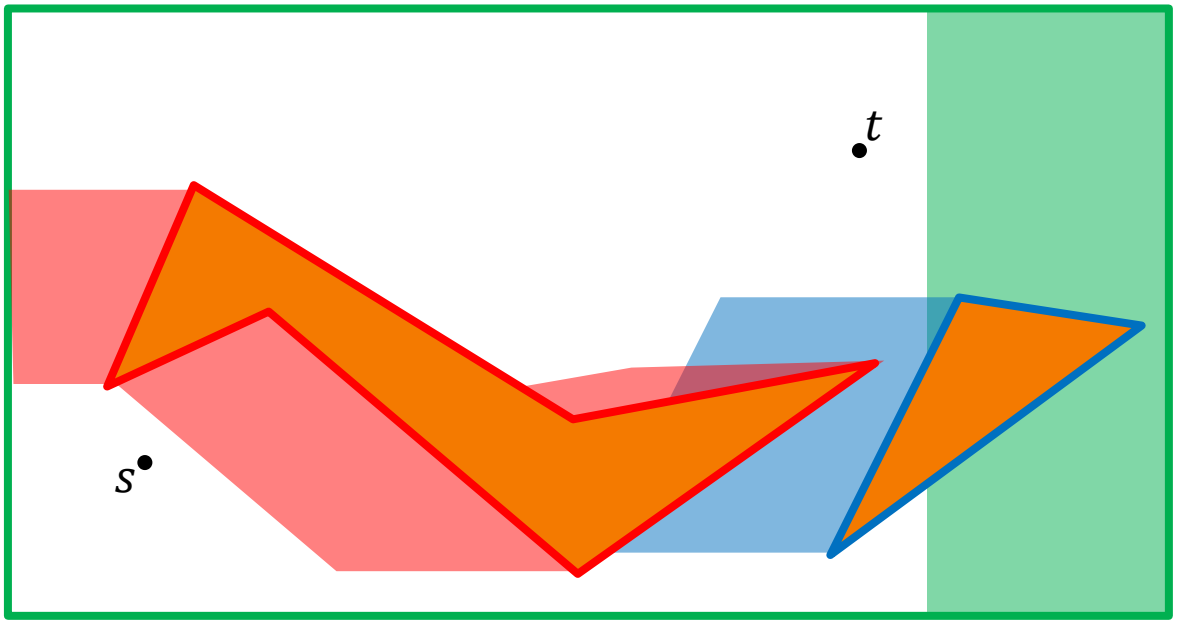
Given a line segment at position s, a set of polygons P, and a target position t, determine if the ladder can be moved (translated and rotated) to t while avoiding the polygonal obstacles.



Moving a Ladder

General Approach:

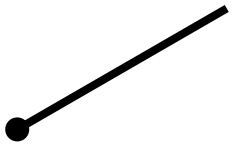
- Compute the Minkowski Sum of P with l.

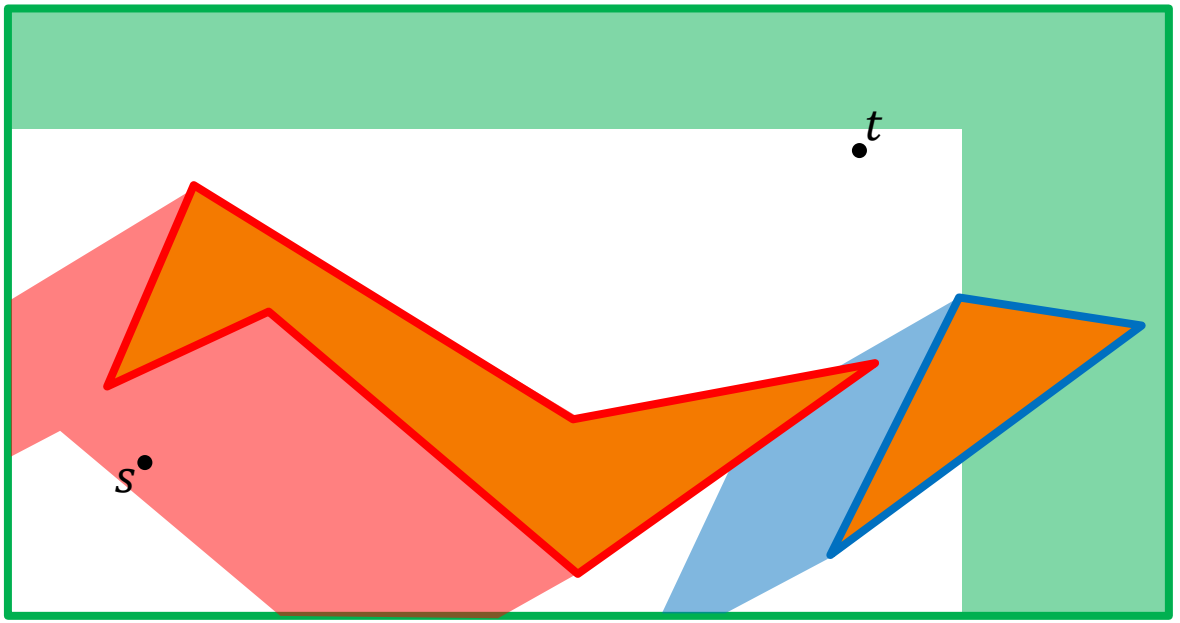


Moving a Ladder

General Approach:

- Compute the Minkowski Sum of P with the ladder.
- Do this for every rotation of the ladder.

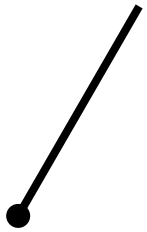


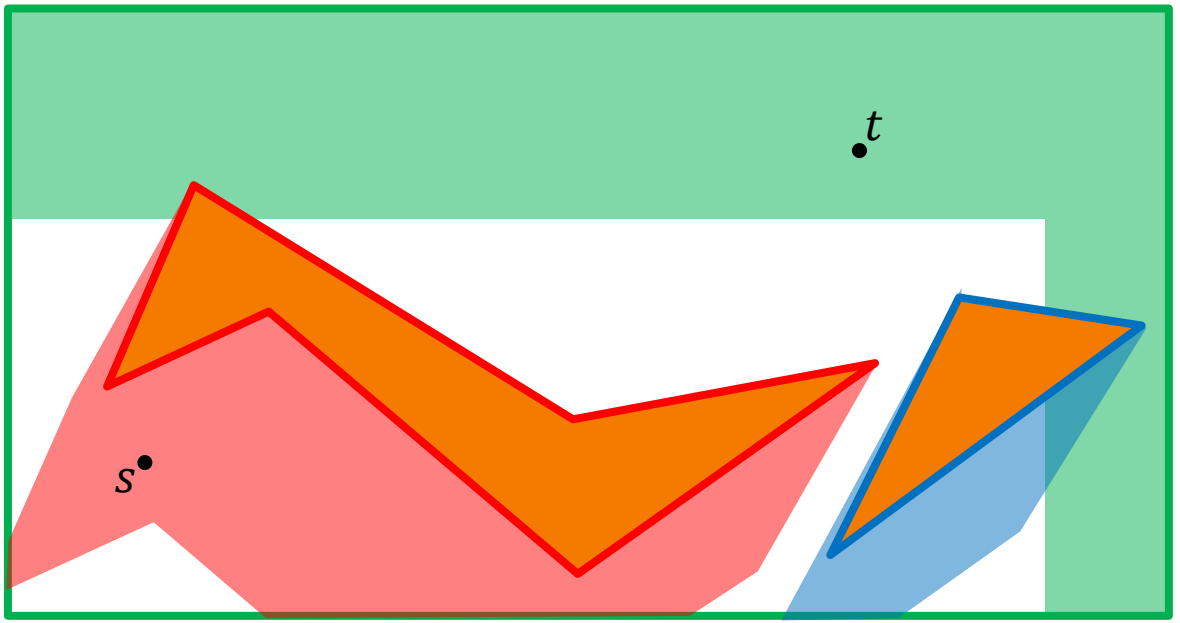


Moving a Ladder

General Approach:

- Compute the Minkowski Sum of P with the ladder.
- Do this for every rotation of the ladder.

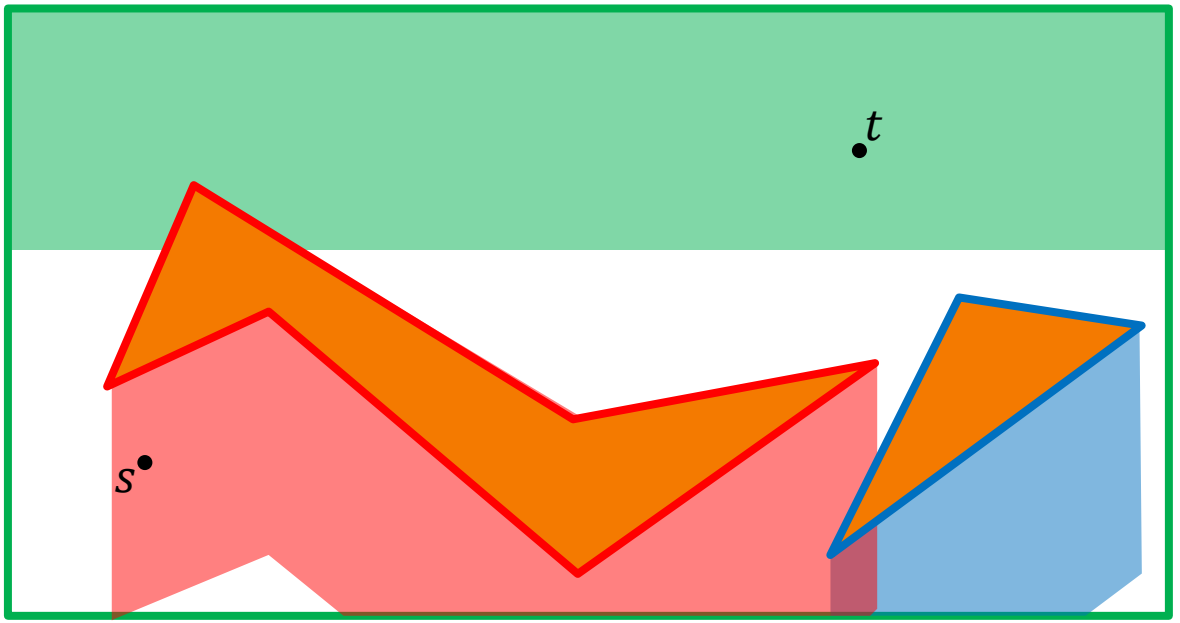




Moving a Ladder

General Approach:

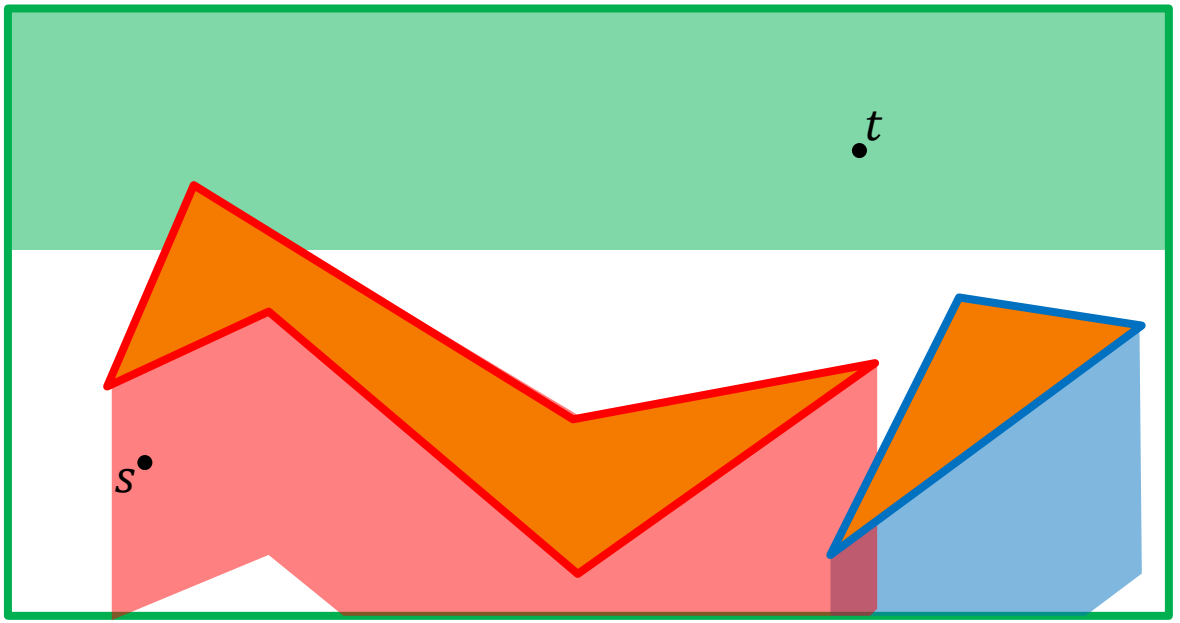
- Compute the Minkowski Sum of P with the ladder.
- Do this for every rotation of the ladder.



Moving a Ladder

General Approach:

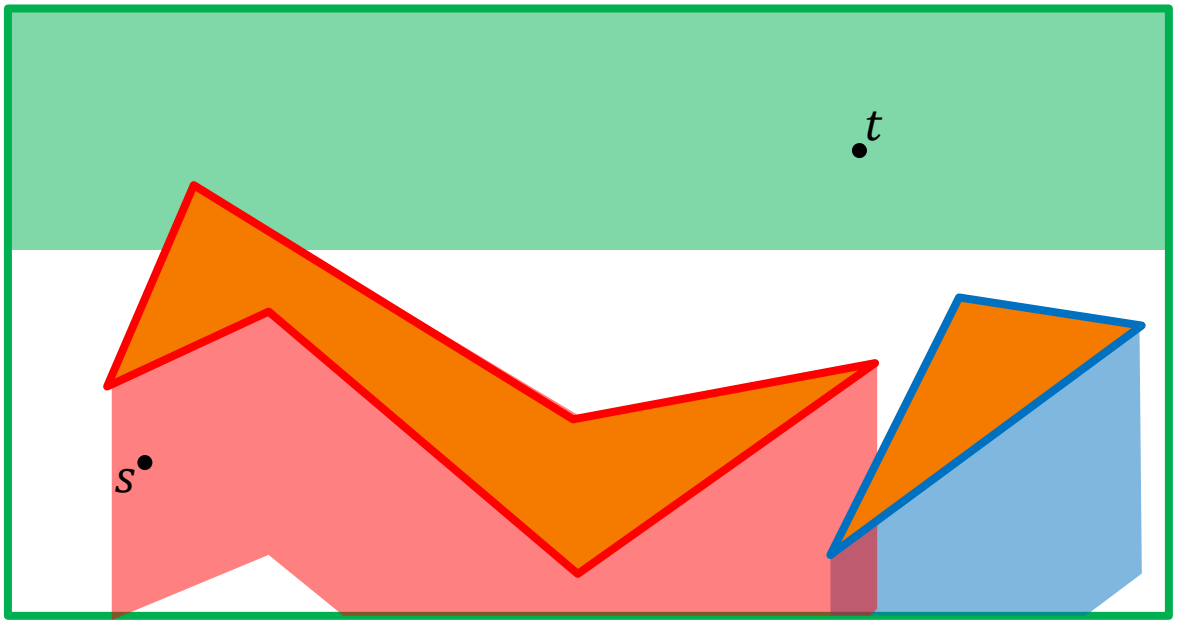
- Compute the Minkowski Sum of P with the ladder.
- Do this for every rotation of the ladder.
- Stack into a 3D volume and test if there is a path from s to t, in the 3D space, that avoids all the polygons.



Moving a Ladder

General Approach:

- Compute the Minkowski Sum of P with the ladder.
- [illegible]
  - There are infinitely many angles.
  - The boundaries of the 3D sums are not planar.
- Stack into a 3D volume and test if there is a path from s to t, in the 3D space, that avoids all the polygons.



Moving a Ladder

Implementation:

Partition empty region into trapezoids:

- Base aligned with the ladder direction.
- Sides defined by the edges of the polygons.

In general, as we rotate the ladder, the shape of the cells change but the topology of the partition (i.e. dual graph) does not.

At discrete events, cells may be inserted, deleted, or merged.

Moving a Ladder

Implementation:

Partition empty region into trapezoids:

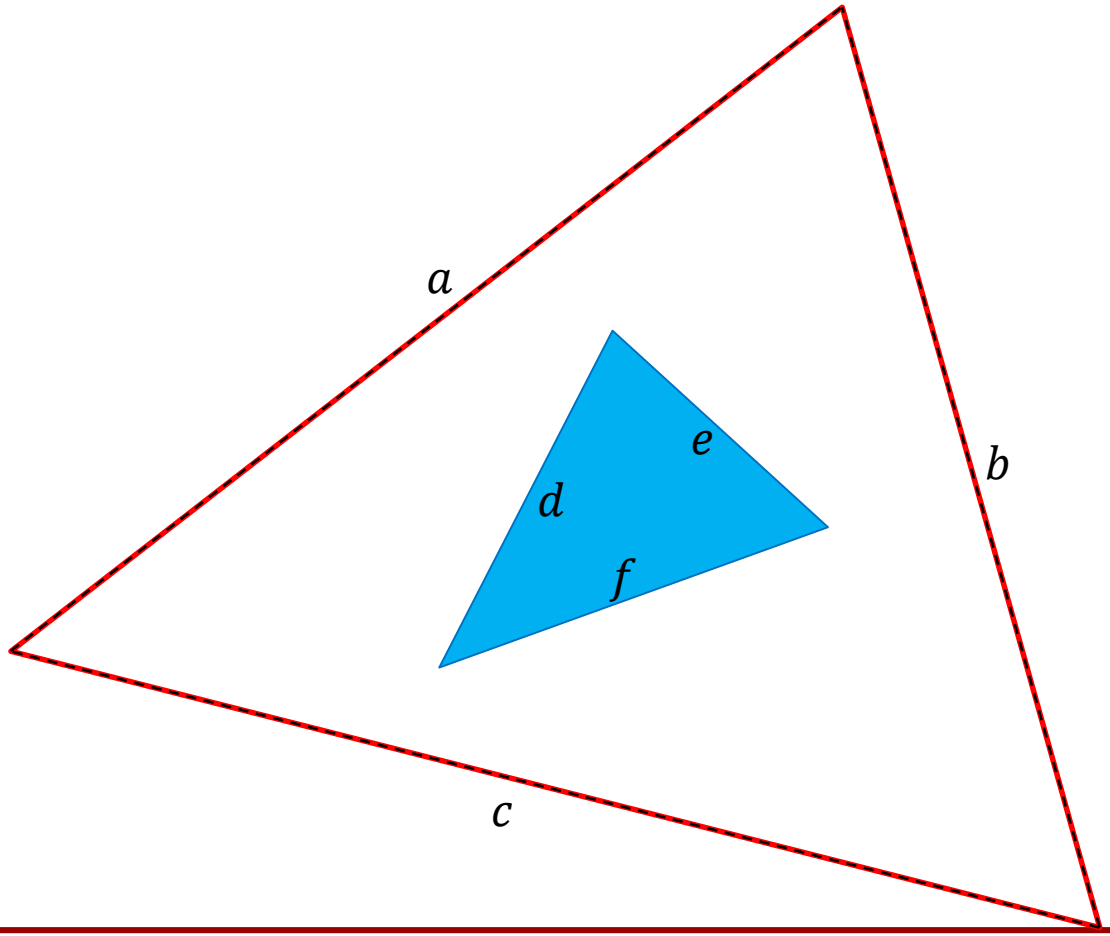
These events can only occur when the rotated ladder aligns with a segment between two vertices of the polygons.

Instead of trying all possible rotations, we only need to consider the $O(|P|^2)$ cases.

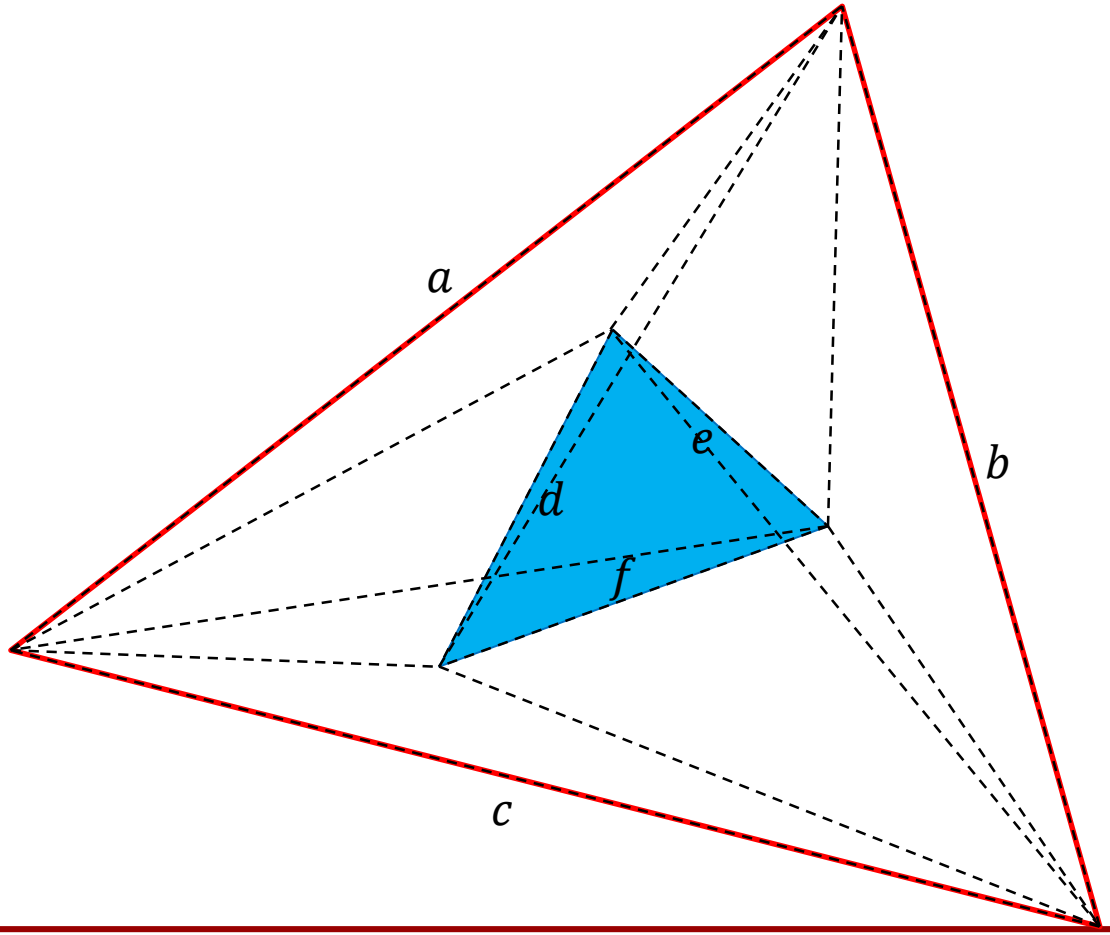
In general, as we rotate the ladder, the shape of the cells change but the topology of the partition (i.e. dual graph) does not.

At discrete events, cells may be inserted, deleted, or merged.

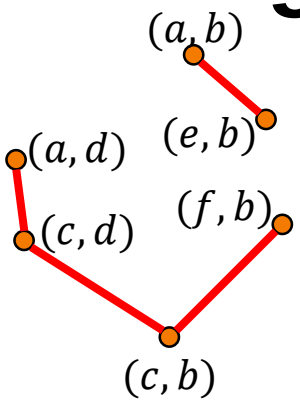
Moving a Ladder

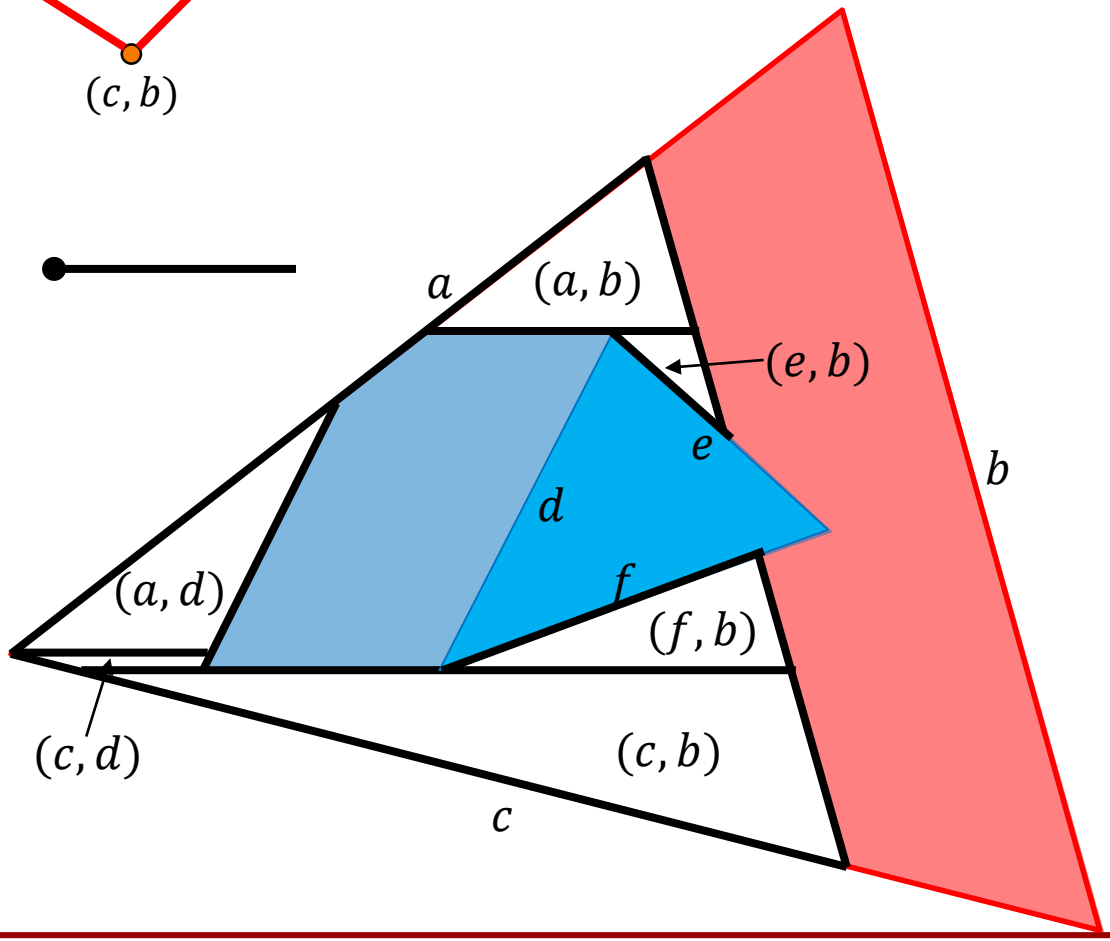


Moving a Ladder

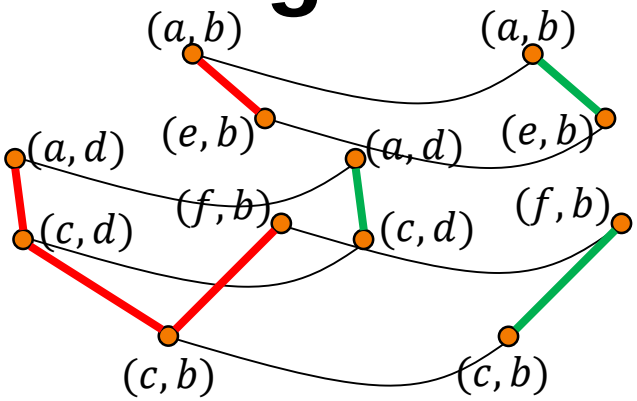


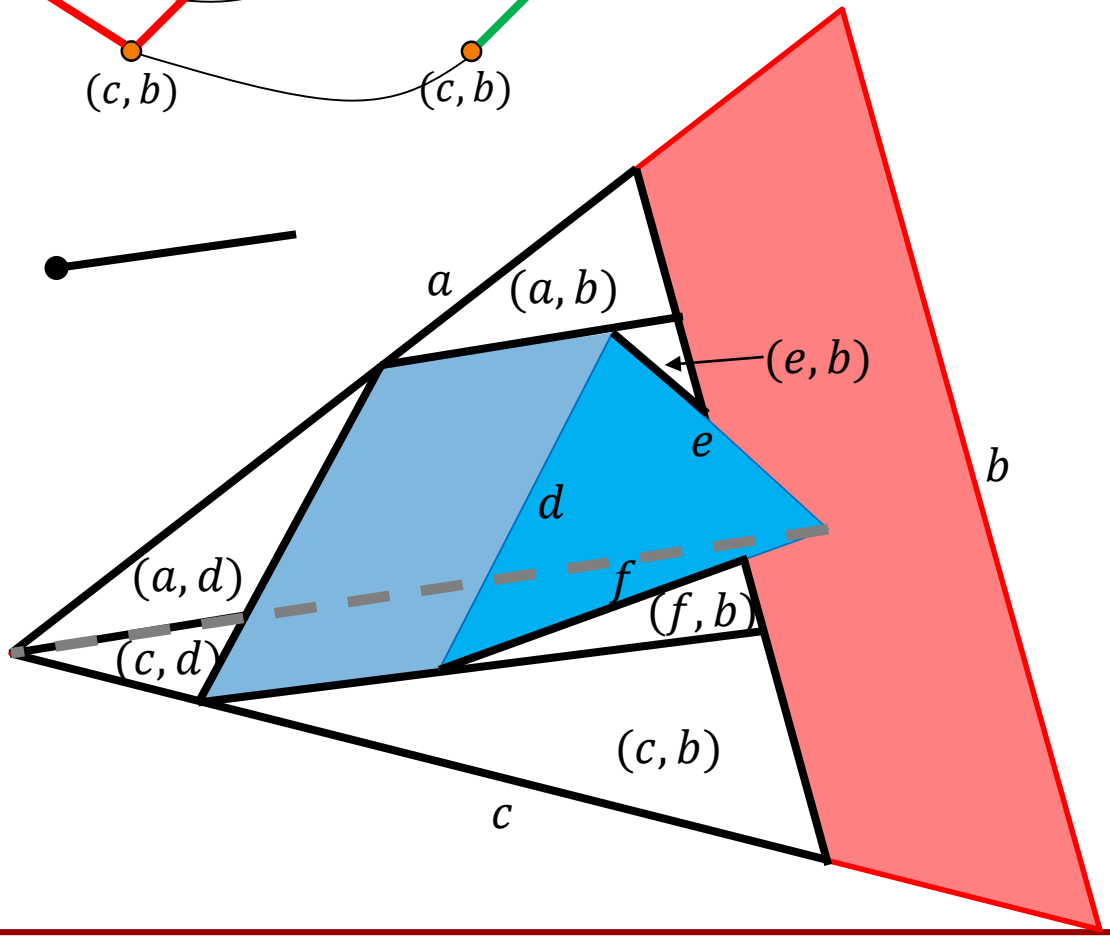
Moving a Ladder



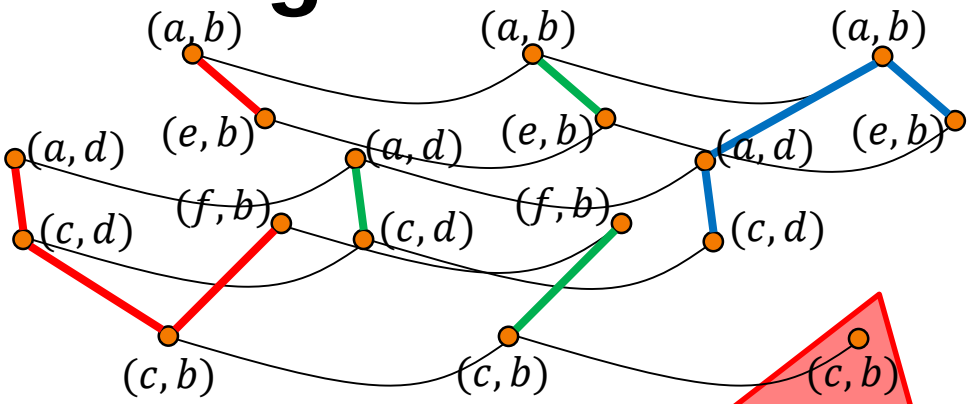


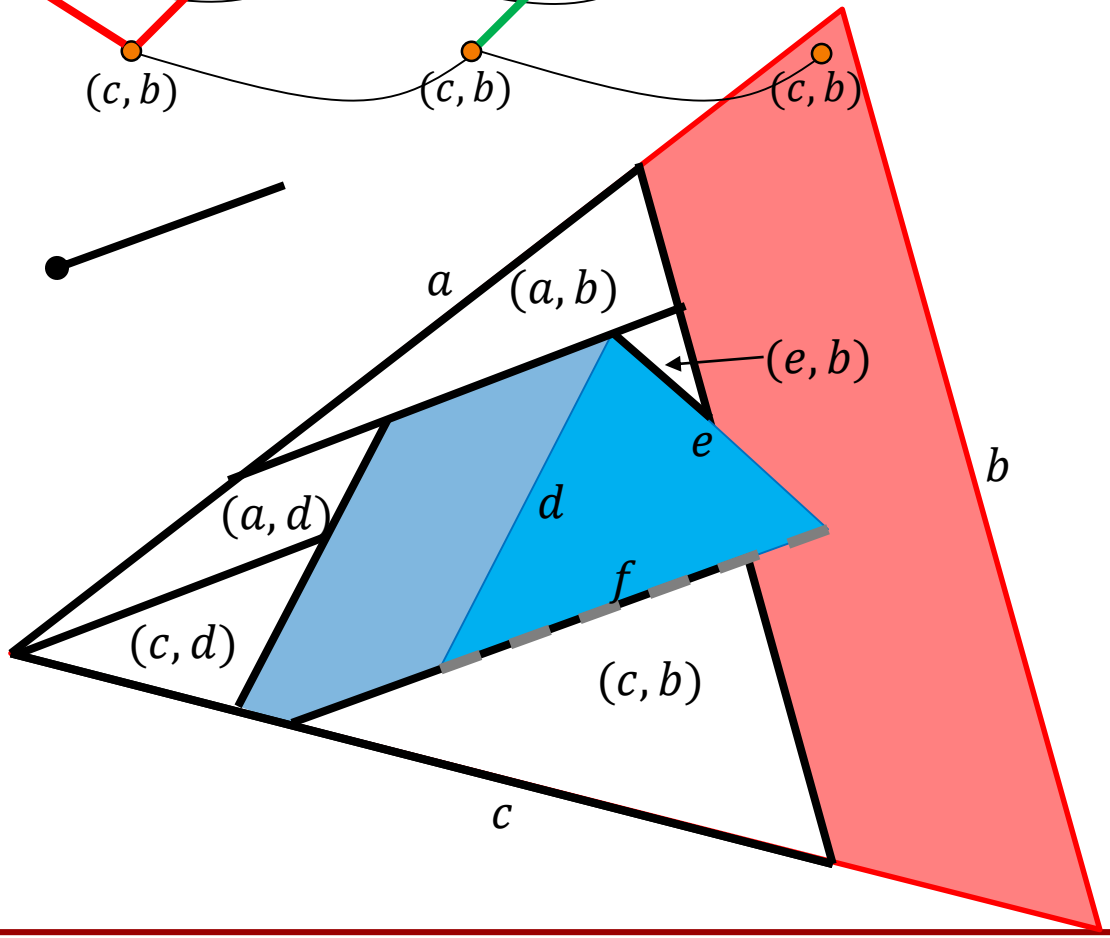
Moving a Ladder



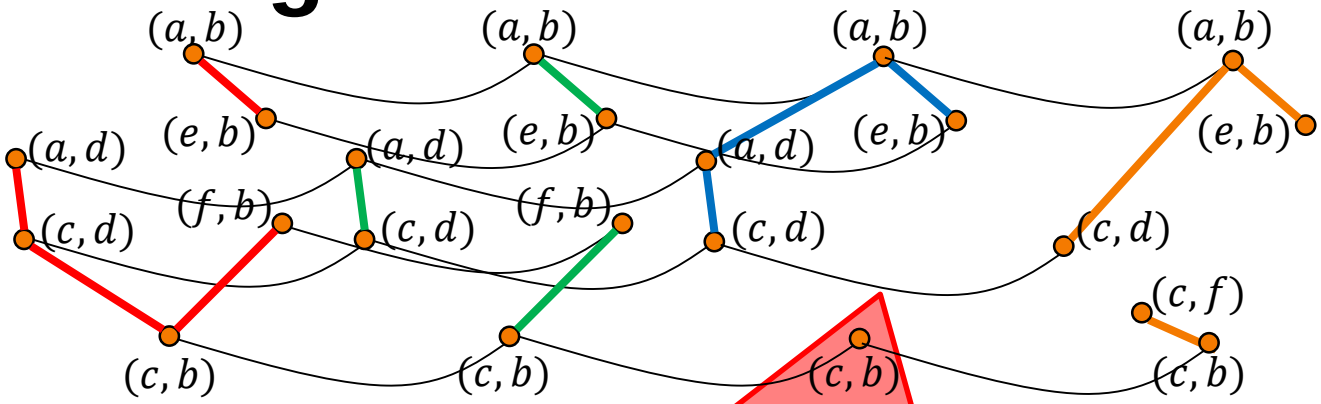


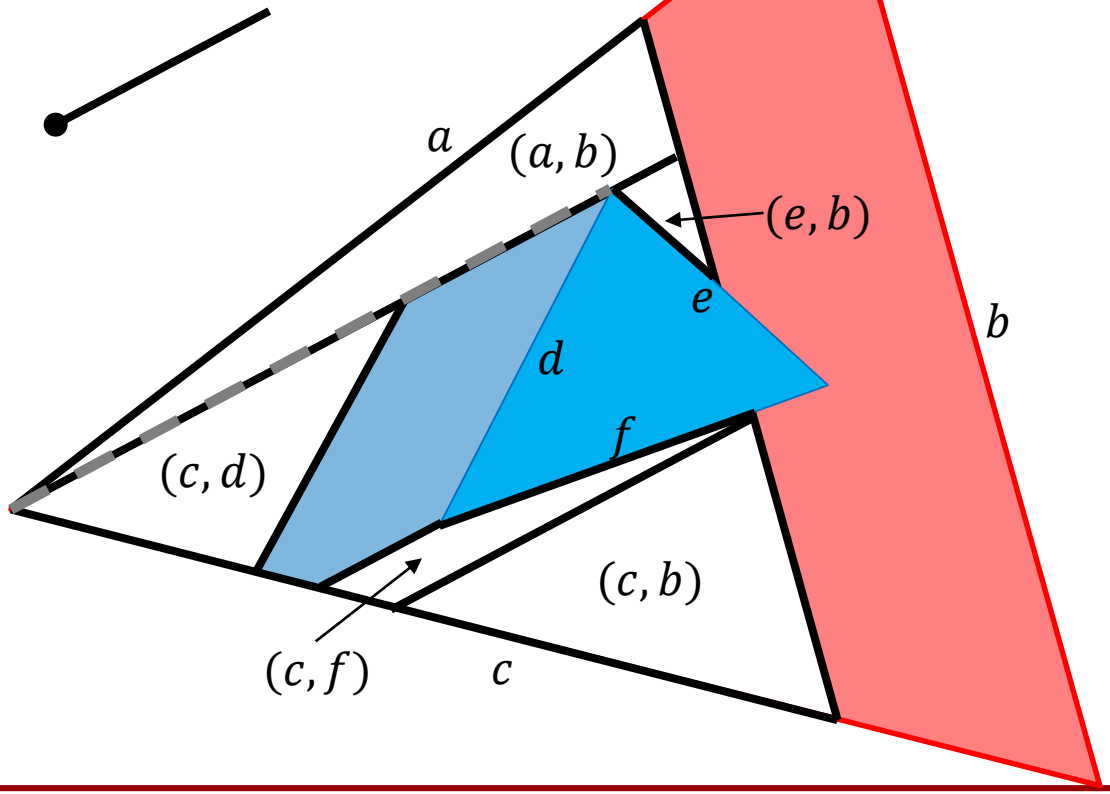
Moving a Ladder





Moving a Ladder





Moving a Ladder

Approach:

- Set $\Theta = \{\theta_0 = 0, \theta_1, \ldots, \theta_n\}$ to be sorted angles, with θ_i the angle of the i-th segment.
- For $0 \leq i \leq n$:
  - » Compute the Minkowski Sum of P with the rotation of the ladder by angle $\theta_i + \varepsilon$.
  - » Compute the dual graph of the Minkowski Sum.
  - » Link the $(i-1)$-st and i-th dual graphs.
- Test if there is a path between the cells containing s and t (at $\theta = 0$) in the linked graph.