1. **Properties of Minkowski Sums and Euler’s theorem:**

* Given sets A, B and C formally prove that =
* What is the Minkowski sum (what geometric object and what can you say about it) of

1. Two points?
2. point and a line?
3. Two lines segments (think of all possible cases)?
4. Two Disks?

* Recall that for the proof of Lemma. 6.2 (complexity of a trapezoidal map) we used the property that in a planar graph we have that

Here E and V are the number of edges and vertices in a planar graph, respectively. Prove Eq. 1 (you can assume that V ≥3).

1. **Exact Motion Planning for a Diamond-Shaped Robot:**
   * 1. **Preprocessing phase (1)—constructing the C-space:**

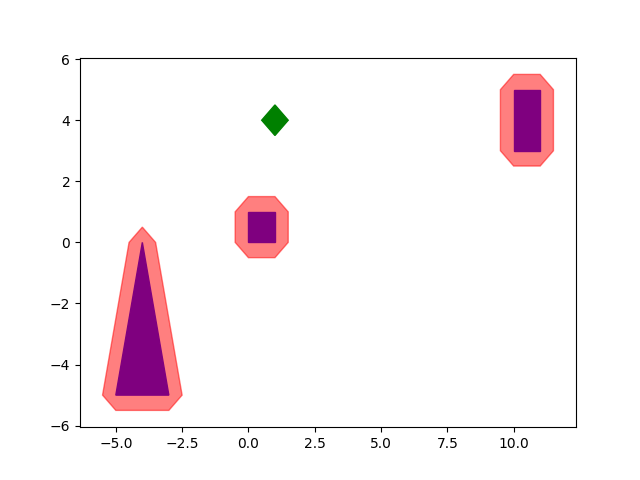
Assumptions**:**

1. All obstacles are convex. ( has vertices)
2. our robot is rotated square-shaped robot . (has vertices)

Computational Complexity:

Since our robot is a rotated square, we have a convex polygon with vertices. Let be an obstacle with vertices, we receive that the time complexity of the Minkowski-sum of those 2 convex polygons is .   
This complexity can be seen in our implementation in the while loop where we iterate over both the obstacle coordinates with and robot coordinates with , each time advancing by one. Such that in the worst case it would take iterations to finish computing the sum.

For the specific data that was provided in the homework, we received the following visualization:



How can non-convex obstacles affect the results? Why? When? Support your claim with examples.

* + 1. **Preprocessing phase (2)—building the visibility diagram:**

Assumptions**:**

1. is the overall number of vertices.

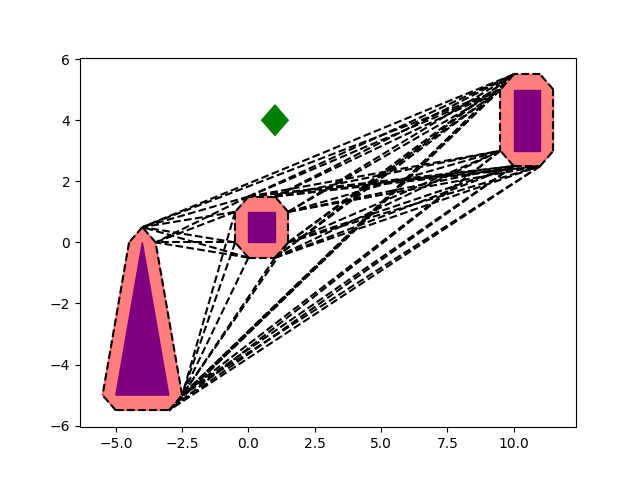
Computational Complexity:

In our implementation of building the visibility graph,, we have a time complexity of because we have a nested for loop where we check for every 2 vertices whether they are visible from each other or not. We know there are pairs of points we need to check because for every vertex it could be connected to any other vertex, and to check visibility requires time in our implementation because we need to check whether the line segment connecting intersects any obstacle or not, therefore because we perform an operation that requires computation time such that it iterates over all the obstacles and determines whether the line segment connecting intersects any of the obstacles.

We perform an operation that takes computation time, different times. Therefore, we receive an overall complexity of for our algorithm.

Note: we saw in class that there exists an implementation in but we chose to implement the naïve implementation for its code simplicity.

For the specific data that was provided in the homework, we received the following visualization:



* + 1. **Query phase—computing shortest paths:**

Assumptions**:**

1. We have a list of the Line-Strings that constitute the visibility graph,.

Computational Complexity:

To compute the shortest path, we first need to build a graph we can run Dijkstra’s algorithm on.

Building the graph, , takes time because we iterate over the line segments from the visibility graph which could reach a number of and for each line we add its end points as vertices to the new graph with a non-directed edge connecting them with a weight equal to the length of the line-segment in time.

After building the graph we can run Dijkstra’s algorithm on it which takes up to and if we plugin , we receive that Dijkstra’s algorithm finishes in . Therefore, the overall complexity of computing the shortest-path given the Line-Strings of the visibility graph is: .

For the specific data that was provided in the homework, we received the following visualization:

