

CS 130B—Data Structures and Algorithms II

Discussion Section Week 7

Written Assignment 3

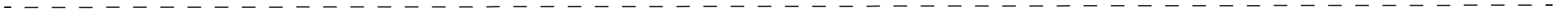
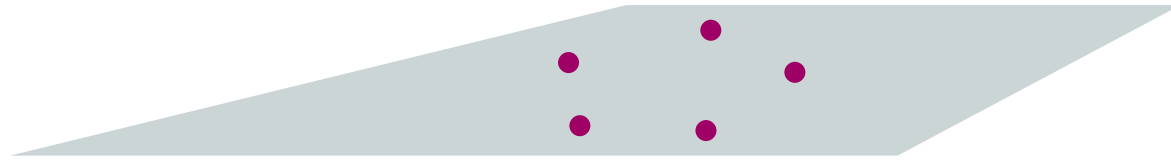
- Due Wednesday May 24th at 4pm

Programming Assignment 3

- Due Wednesday May 31st at 11:59pm

Programming Assignment 3

P

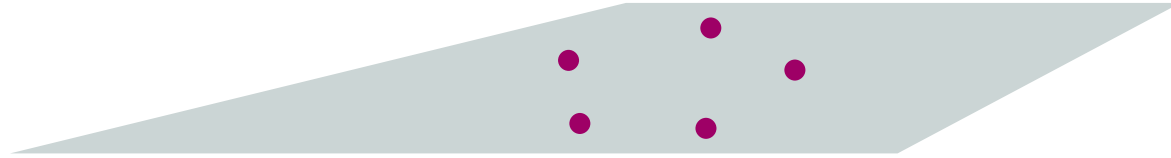


Q



File Format

P



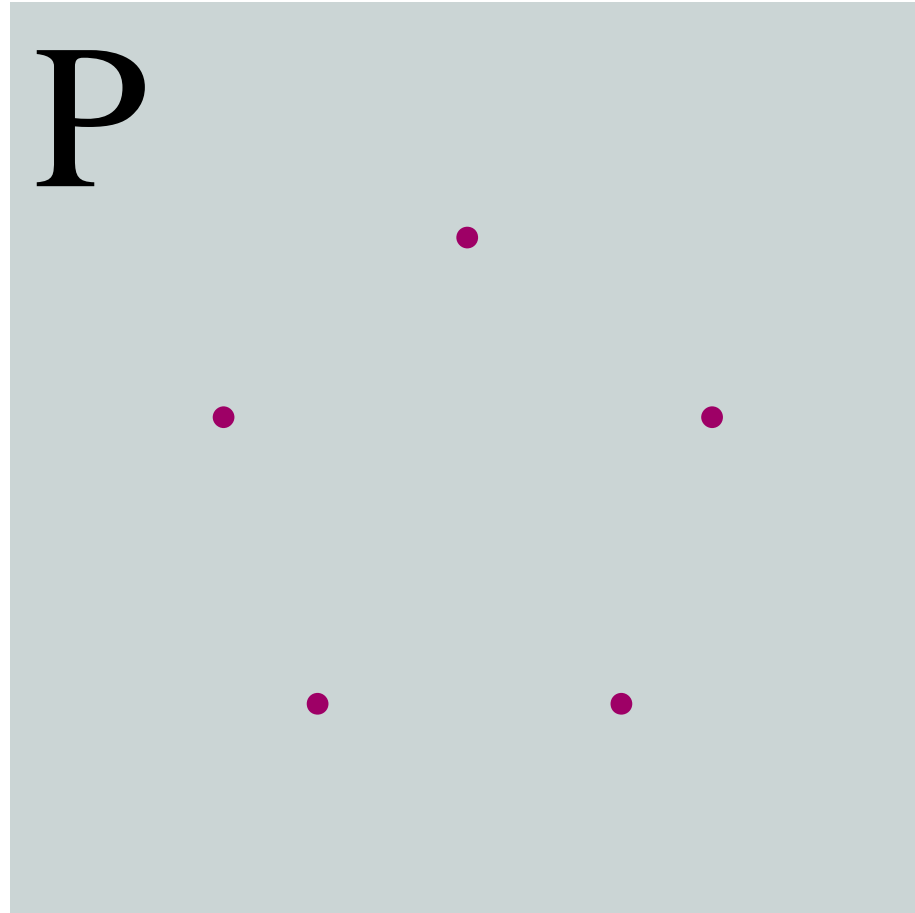
input.txt

```
5 5
x1 y1 z1
x2 y2 z2
x3 y3 z3
x4 y4 z4
x5 y5 z5
x6 y6 z6
x7 y7 z7
x8 y8 z8
x9 y9 z9
x10 y10 z10
```

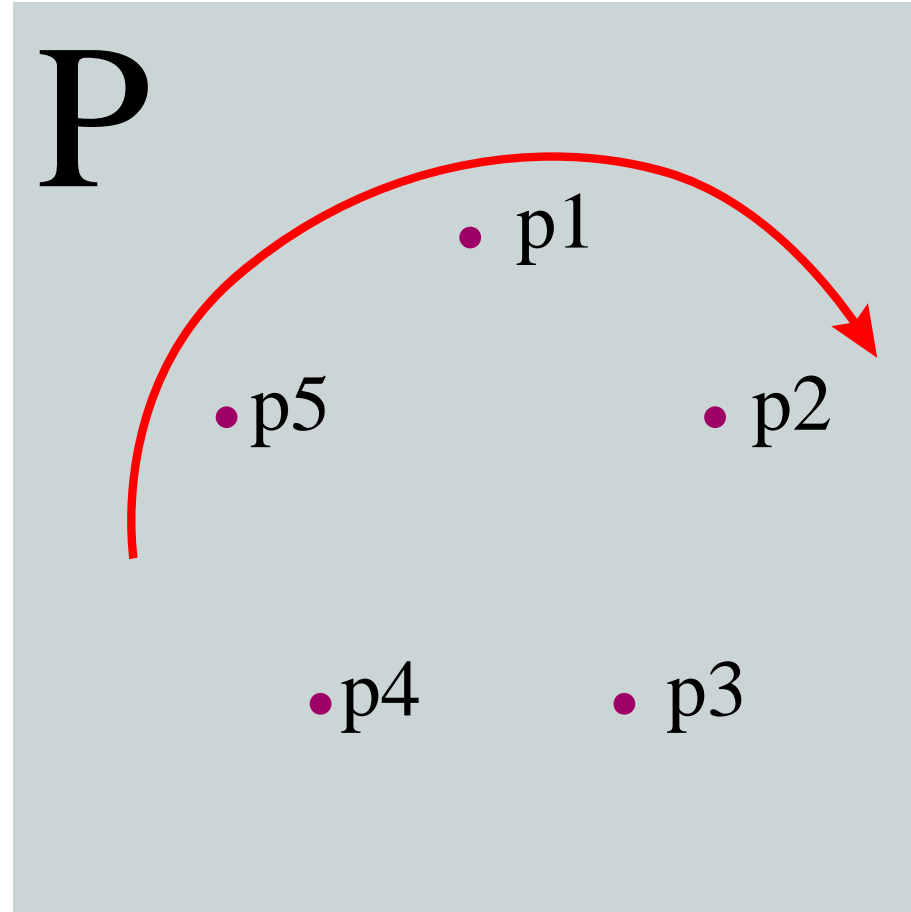
Q



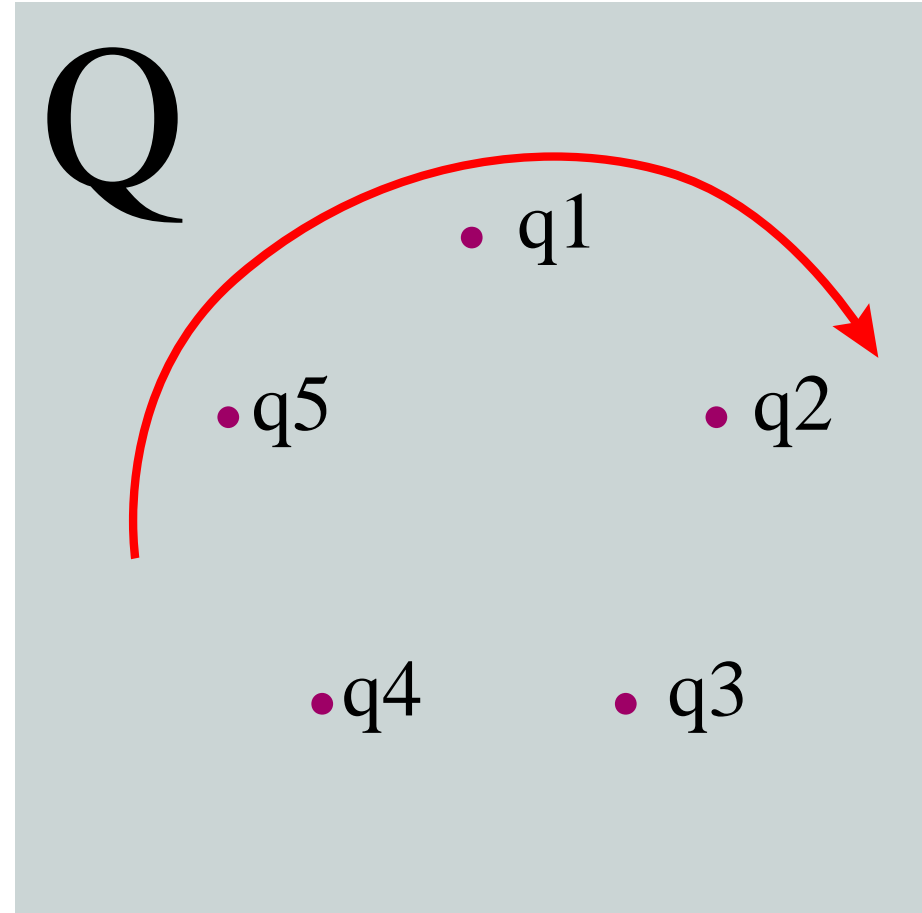
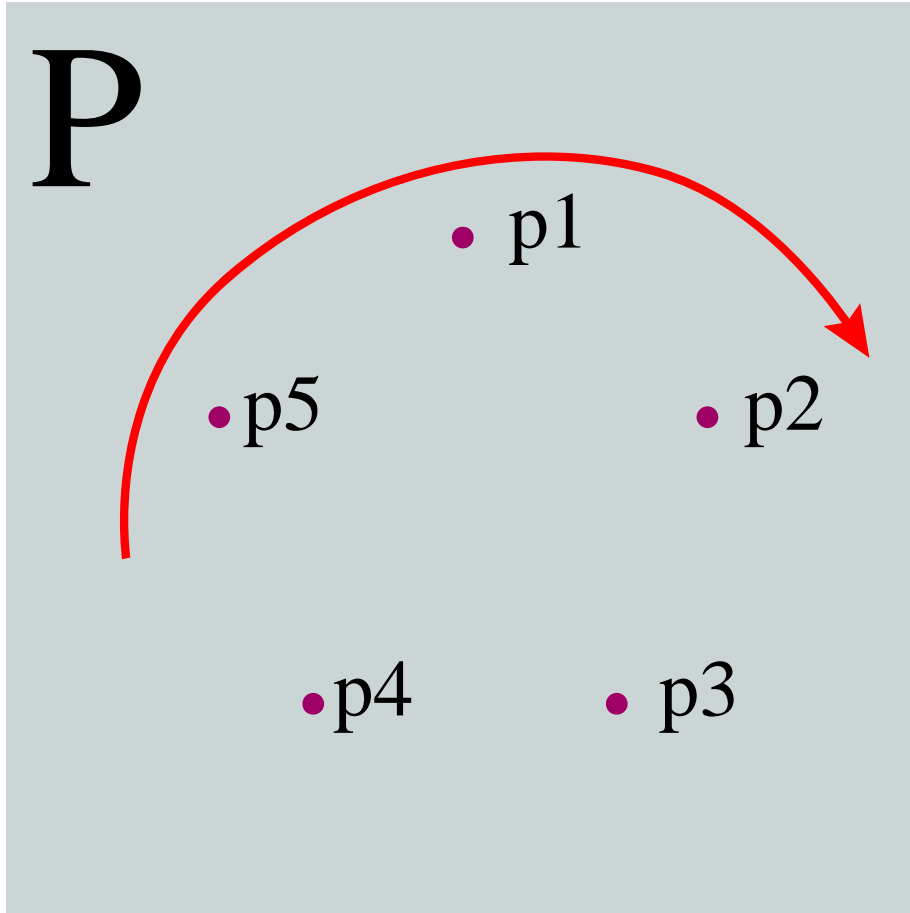
Top View



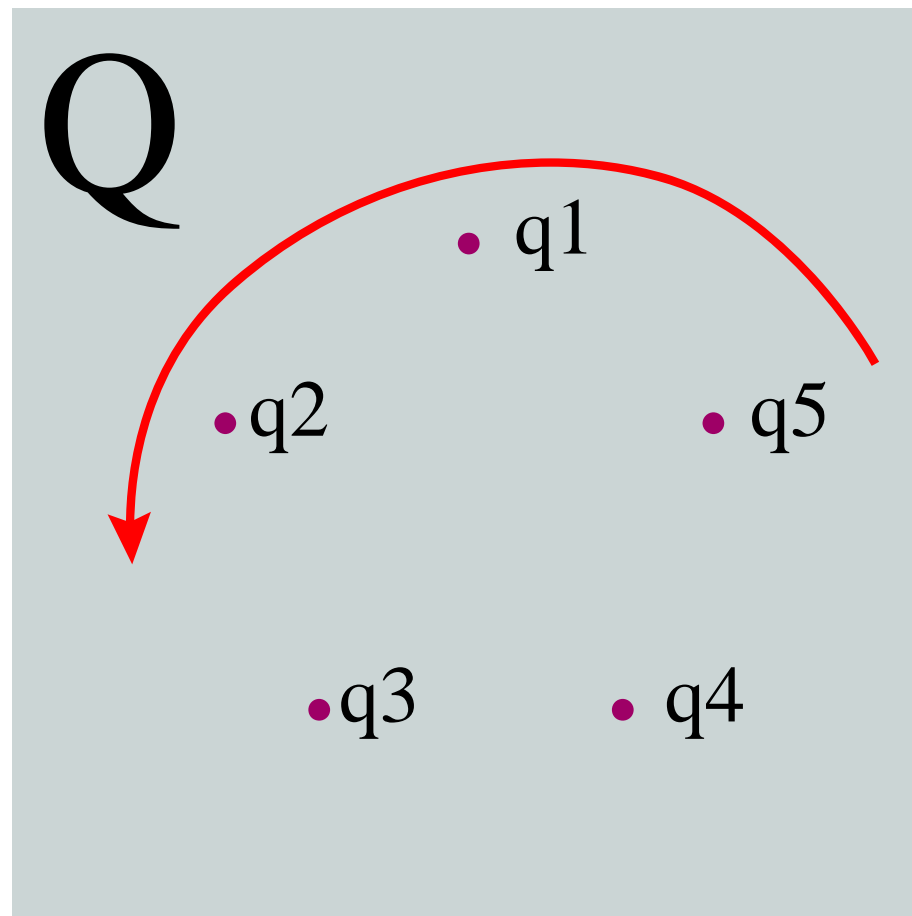
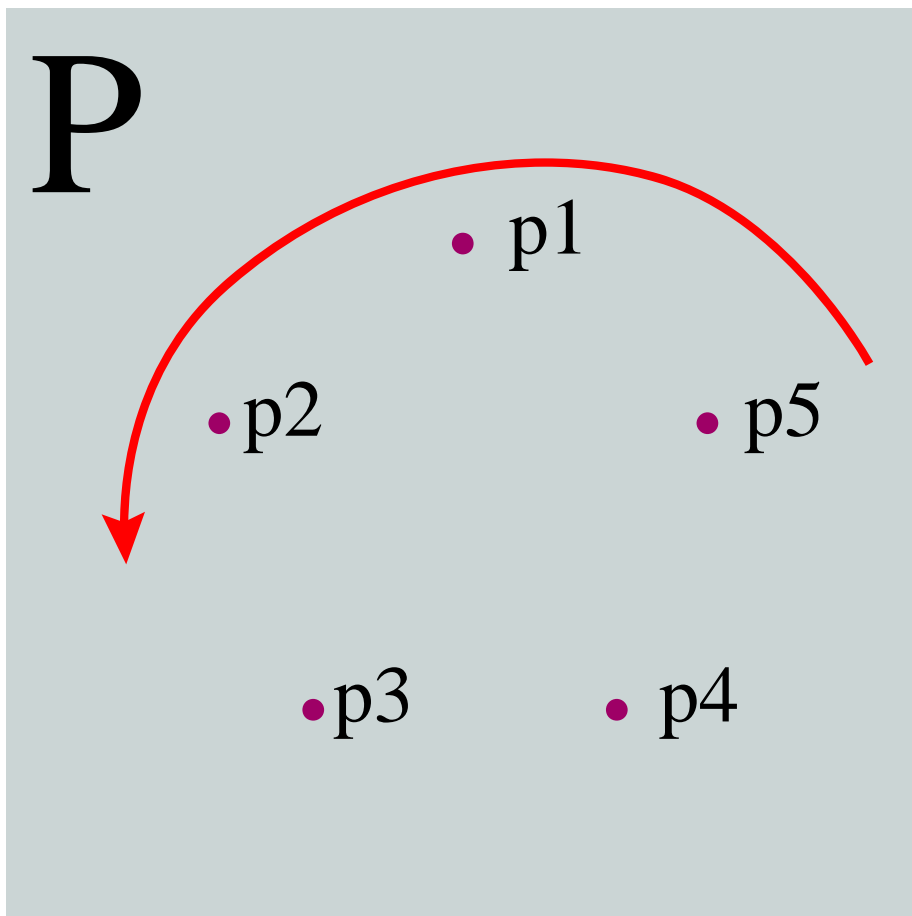
Points Are In Order



Same Order Applies Top and Bottom

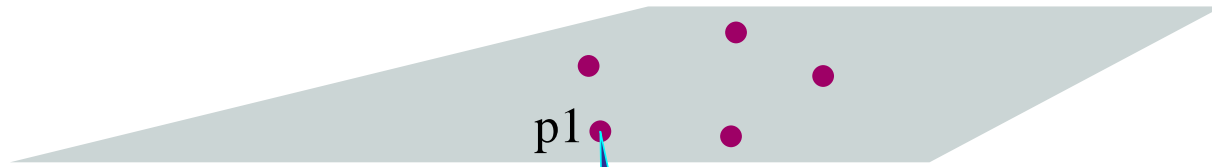


Or..

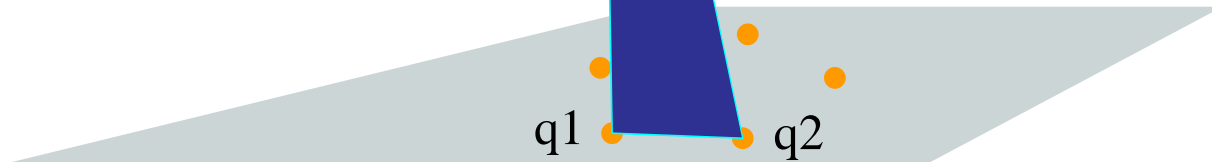


Finding Triangles

P

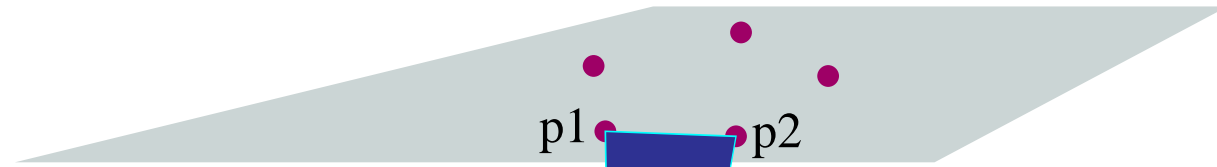


Q

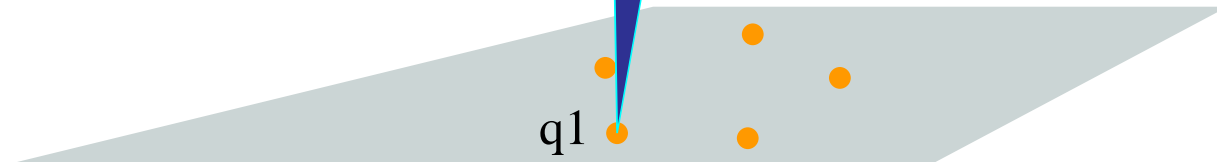


Need to Minimize Area of Surface

P



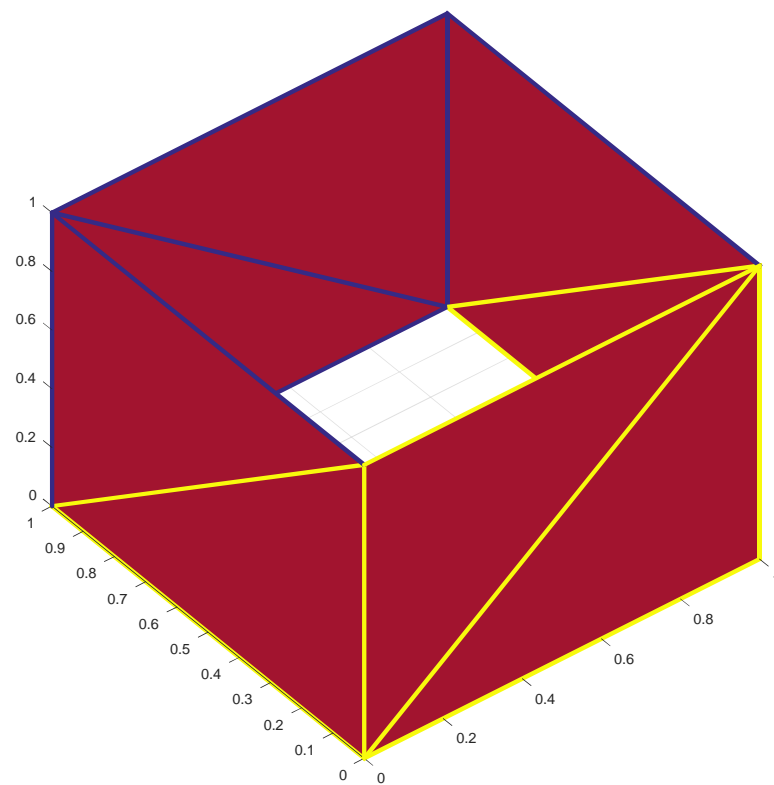
Q



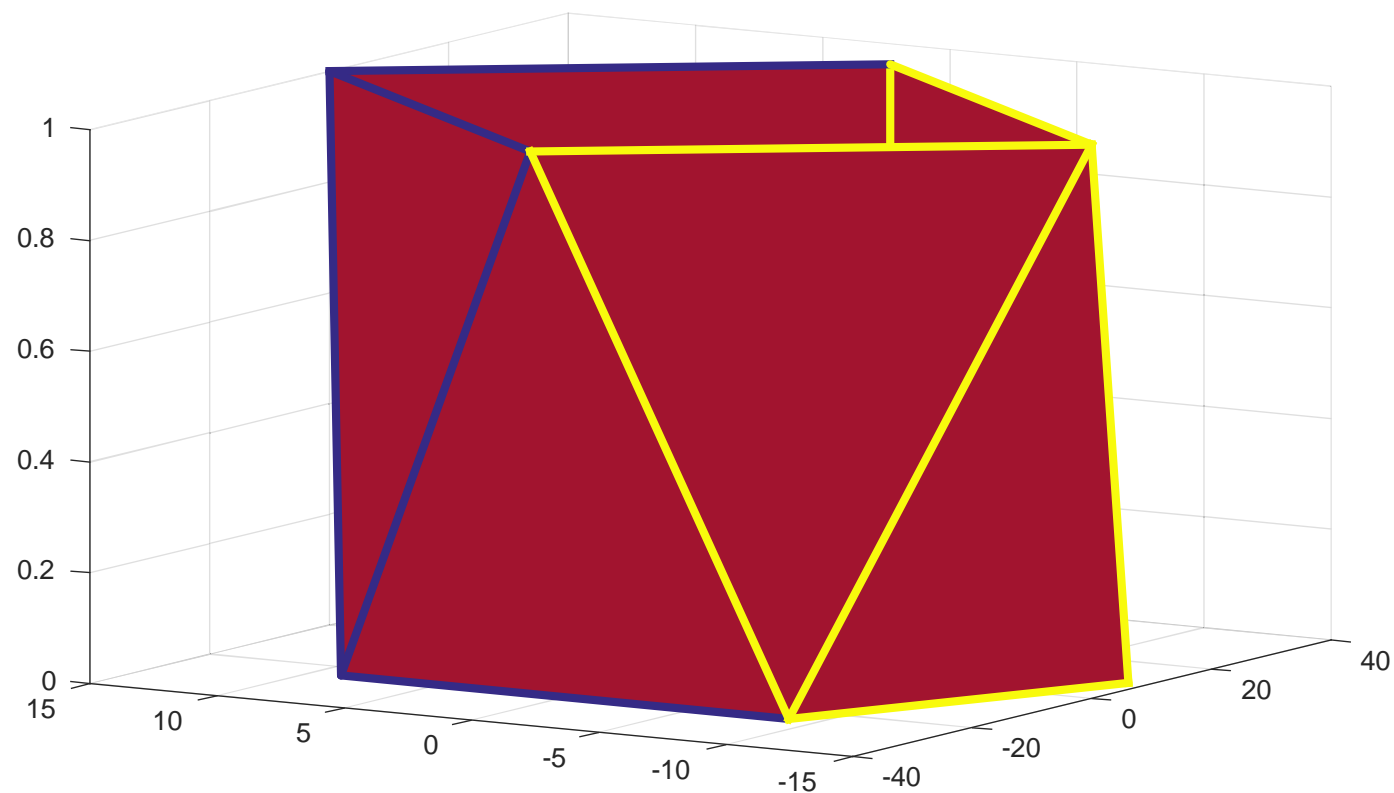
Output Examples

Plots from the Test Data

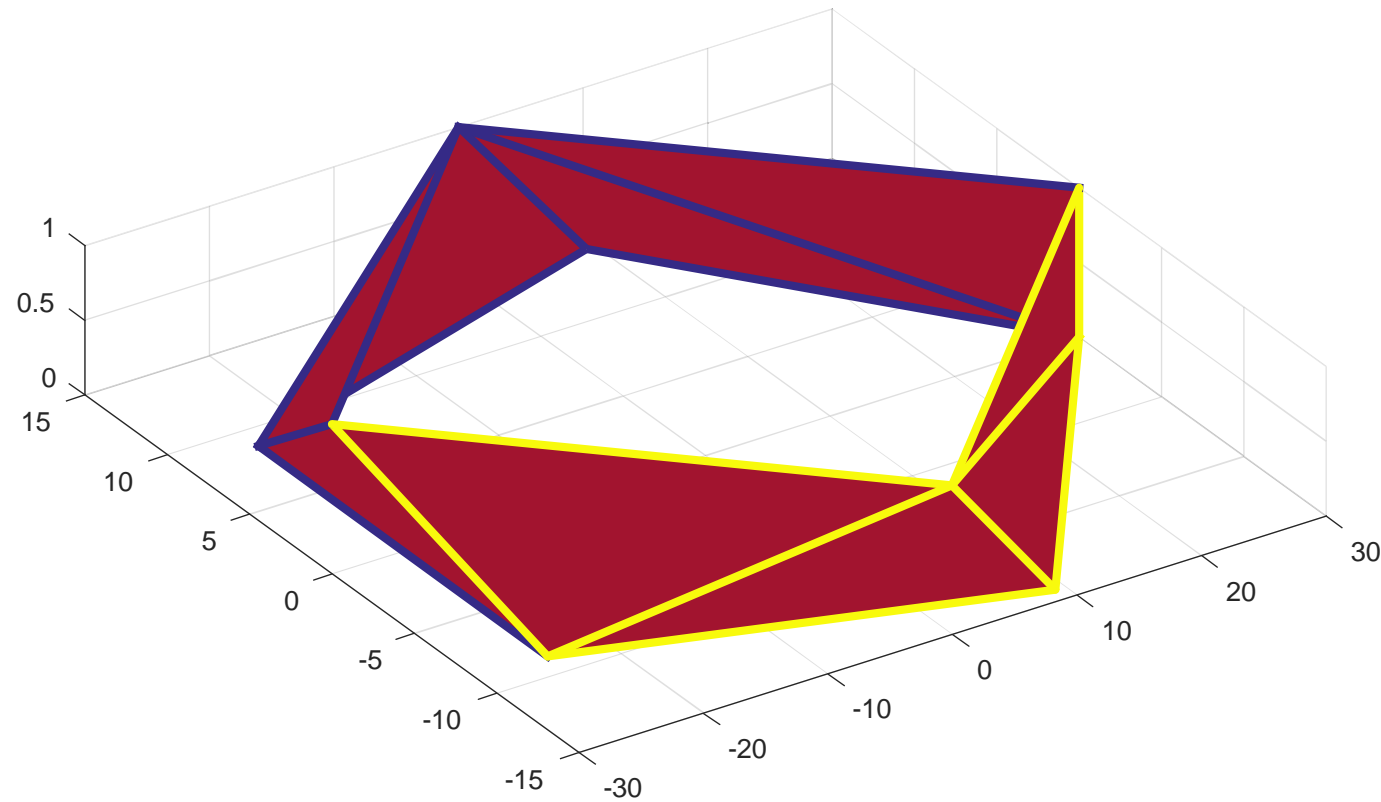
Two Squares



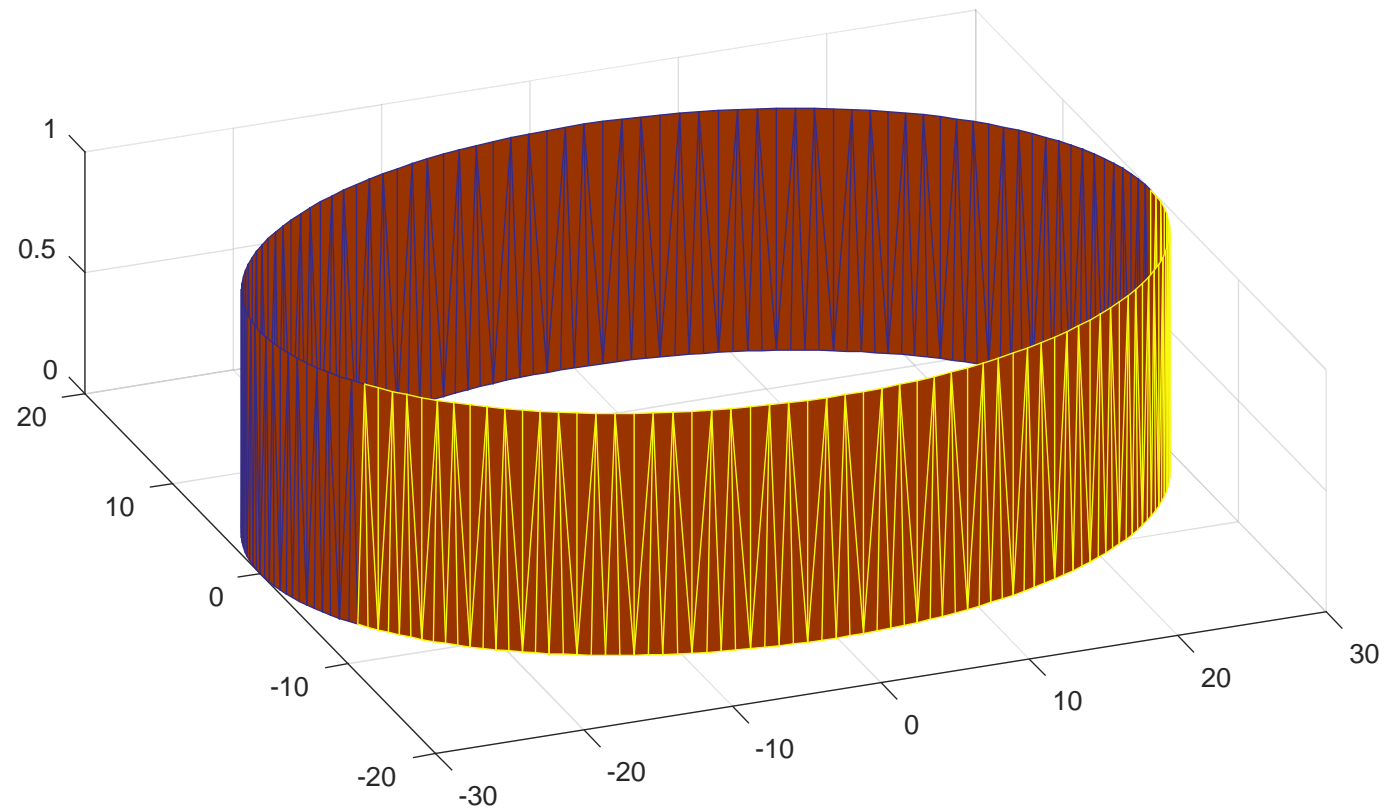
Two Different Samples of an Ellipse



Two Different Samples of an Ellipse—View2



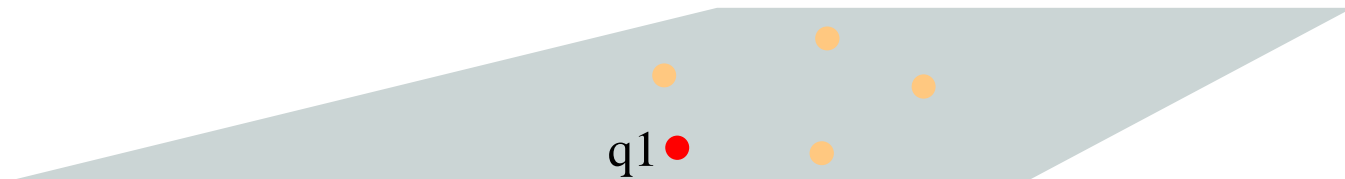
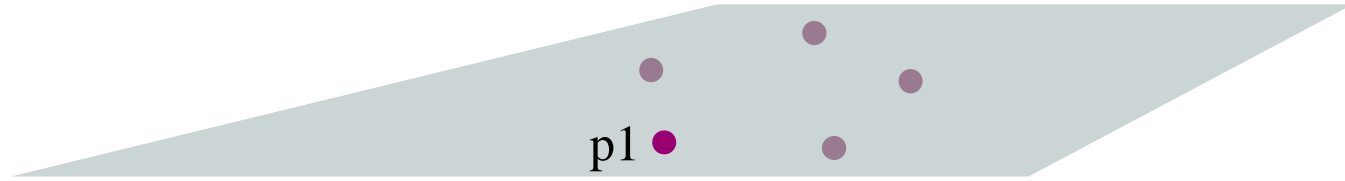
Two Ellipses



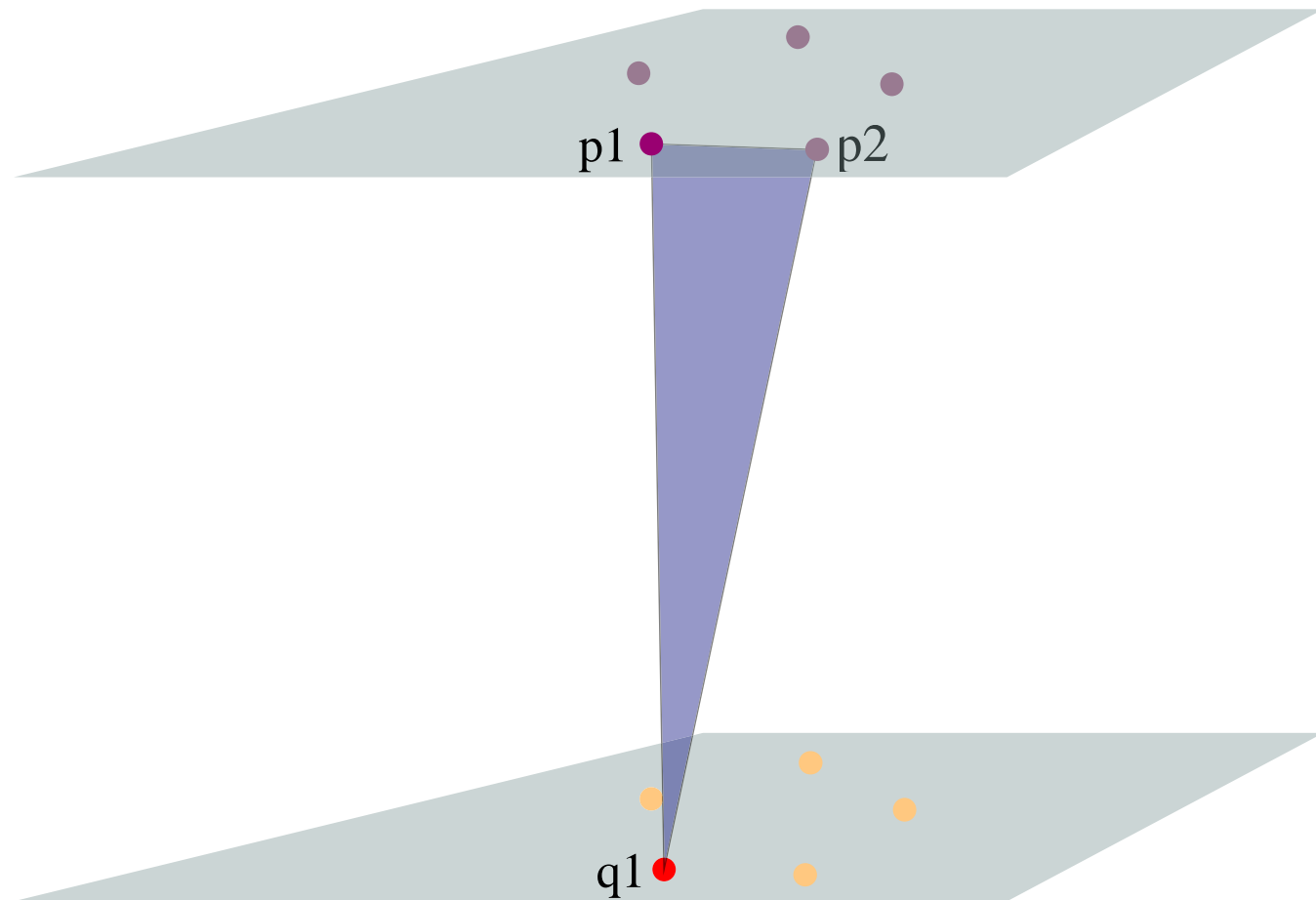
General Approach

A Recursive Solution

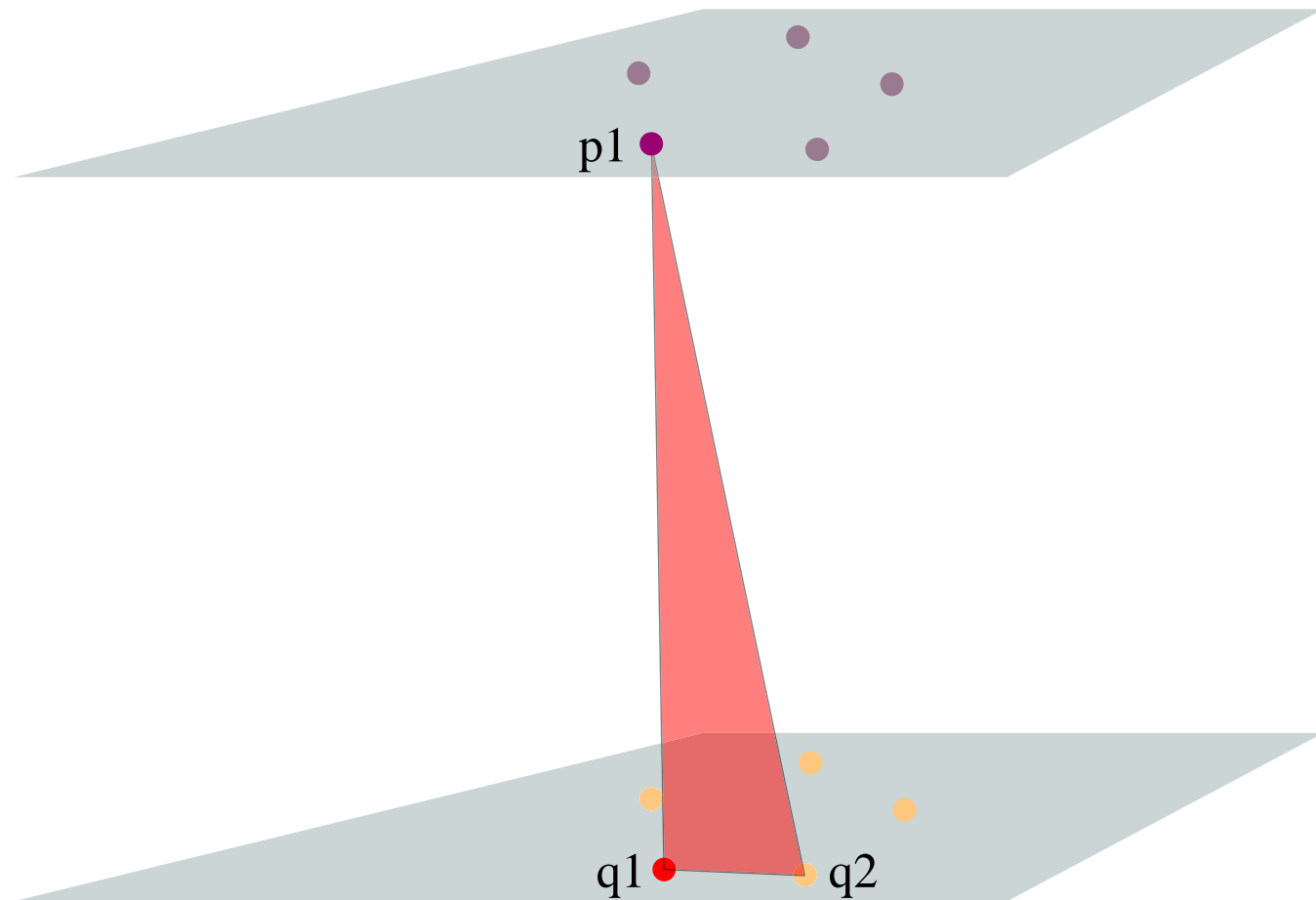
Starting at an Arbitrary Point



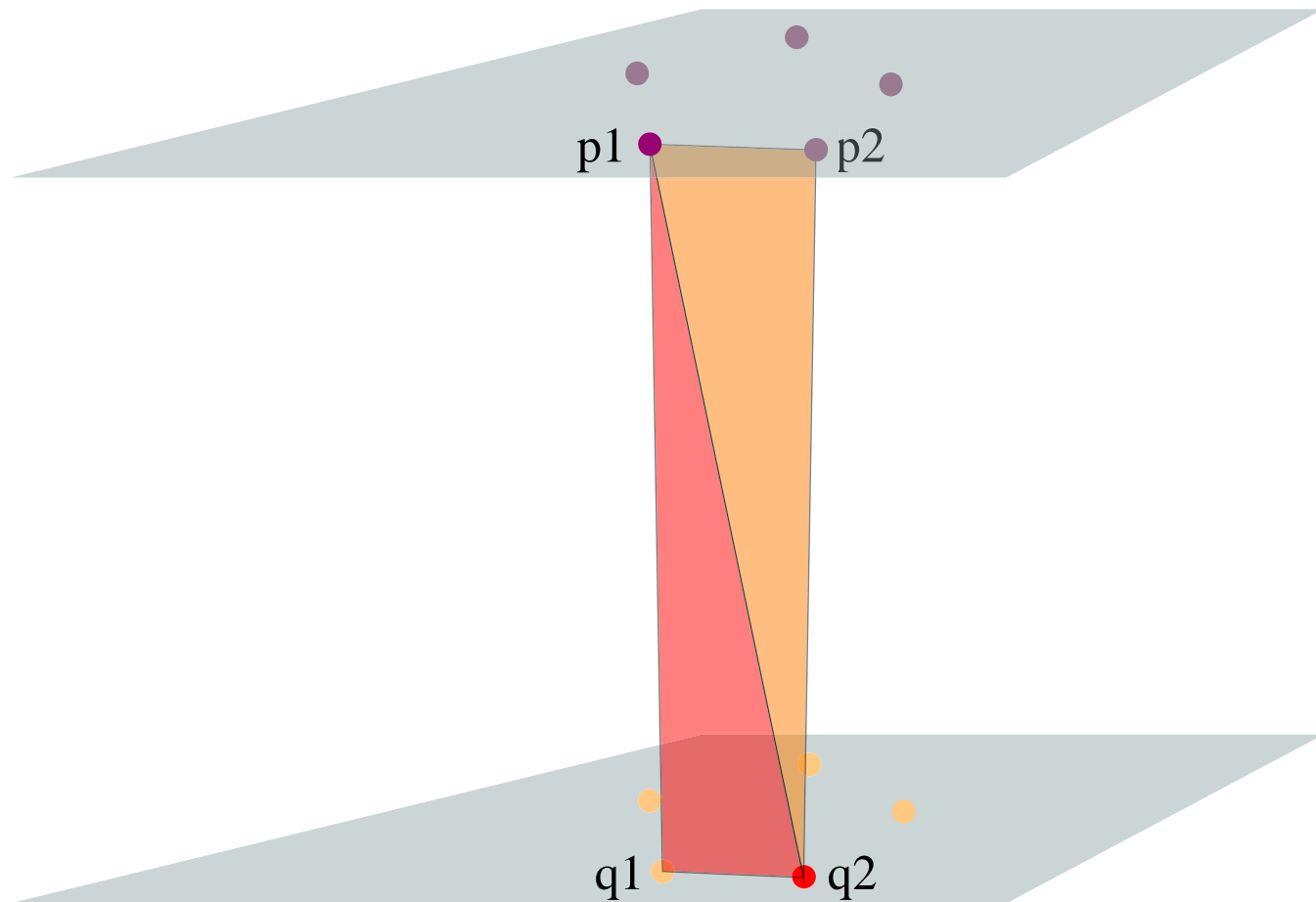
Triangle Option One



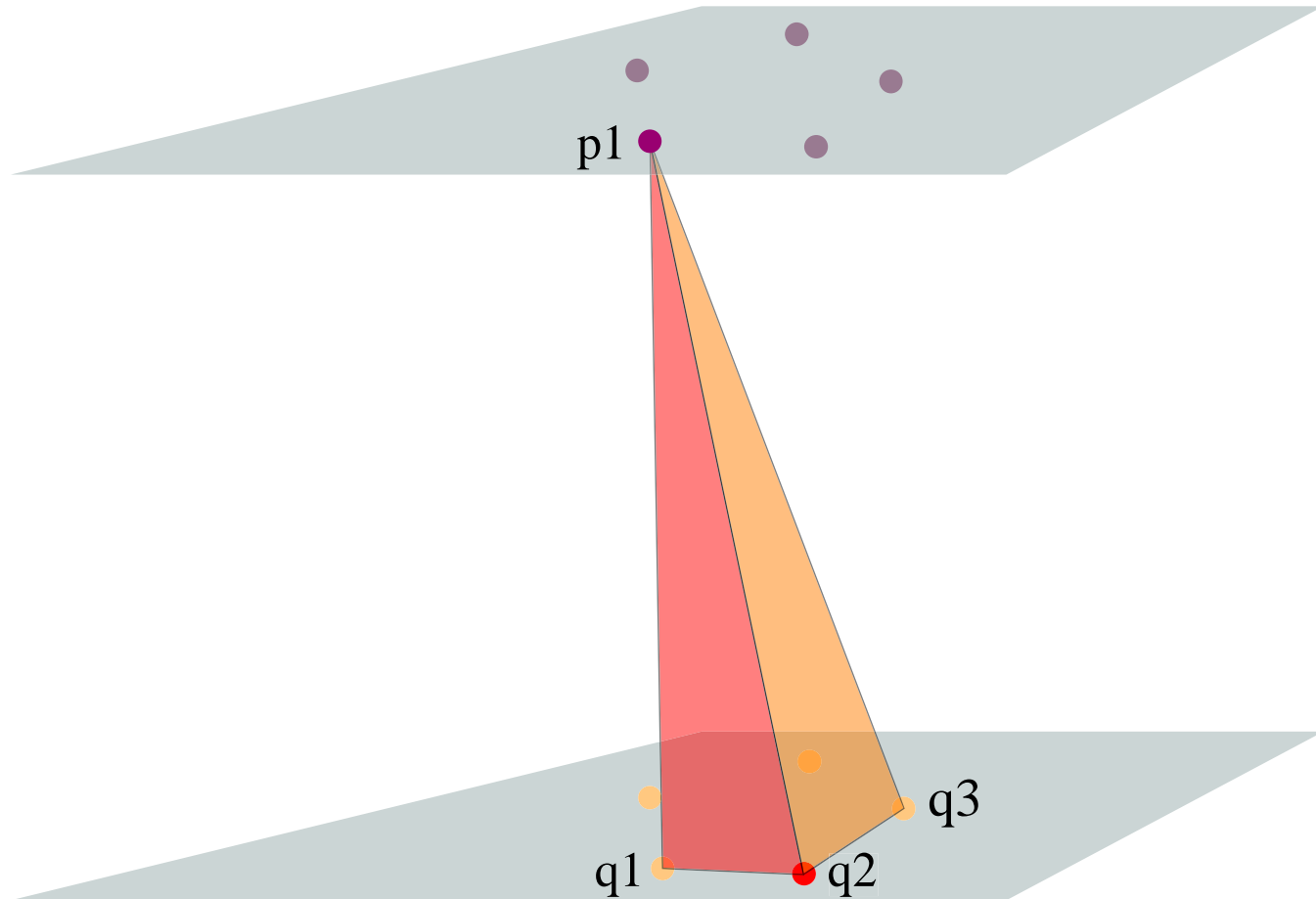
Triangle Option Two



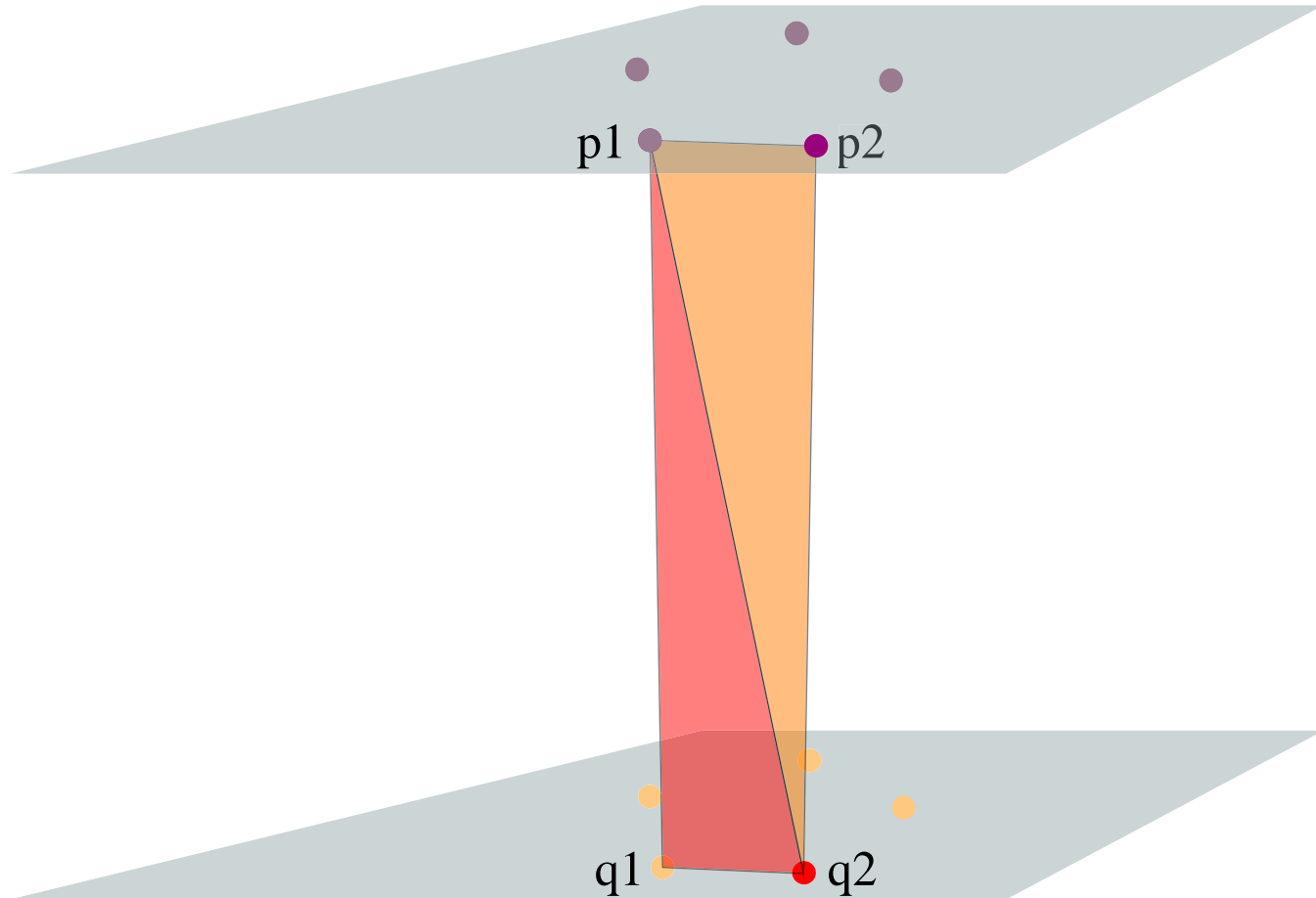
Option 2 - 1



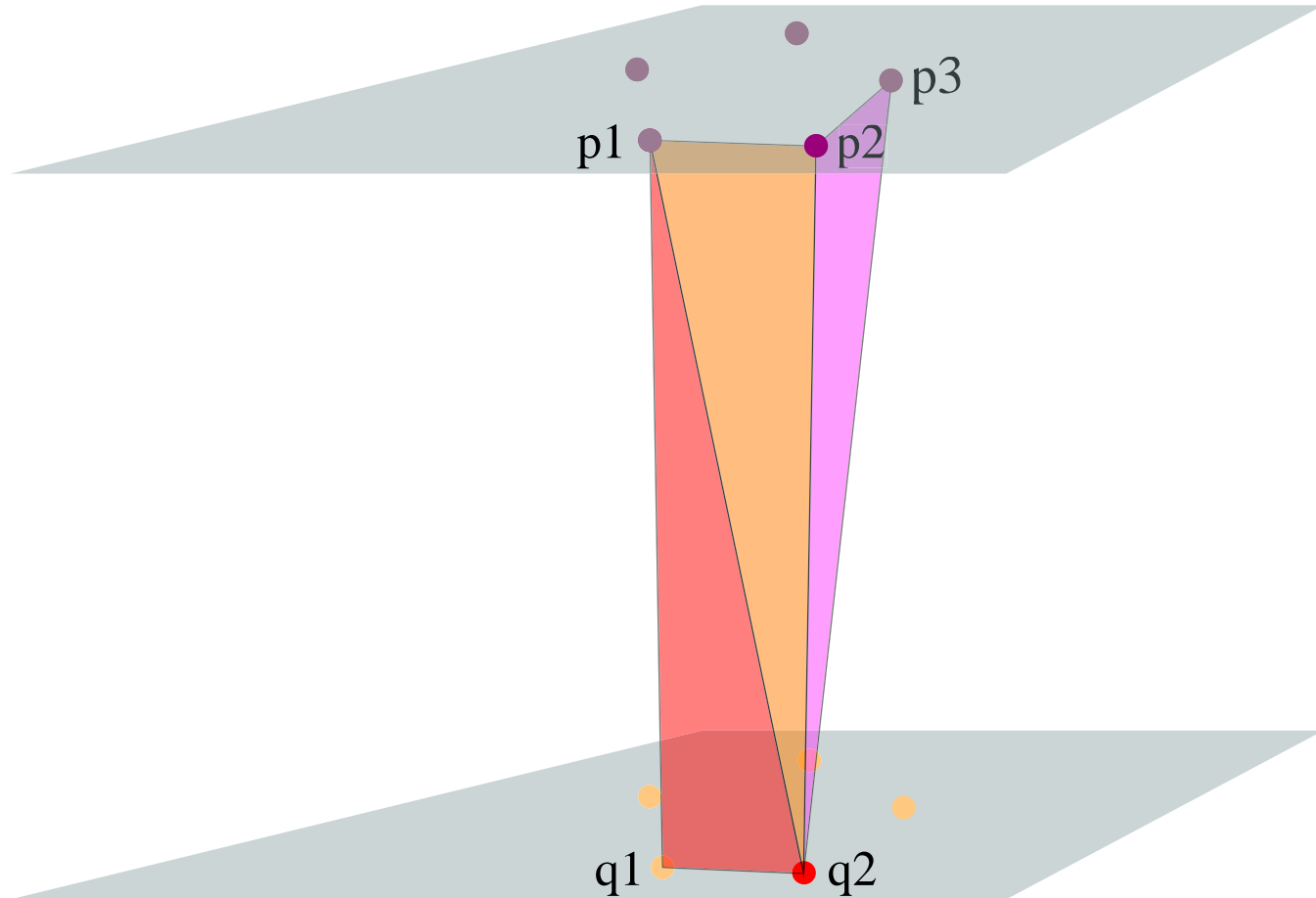
Option 2 - 2



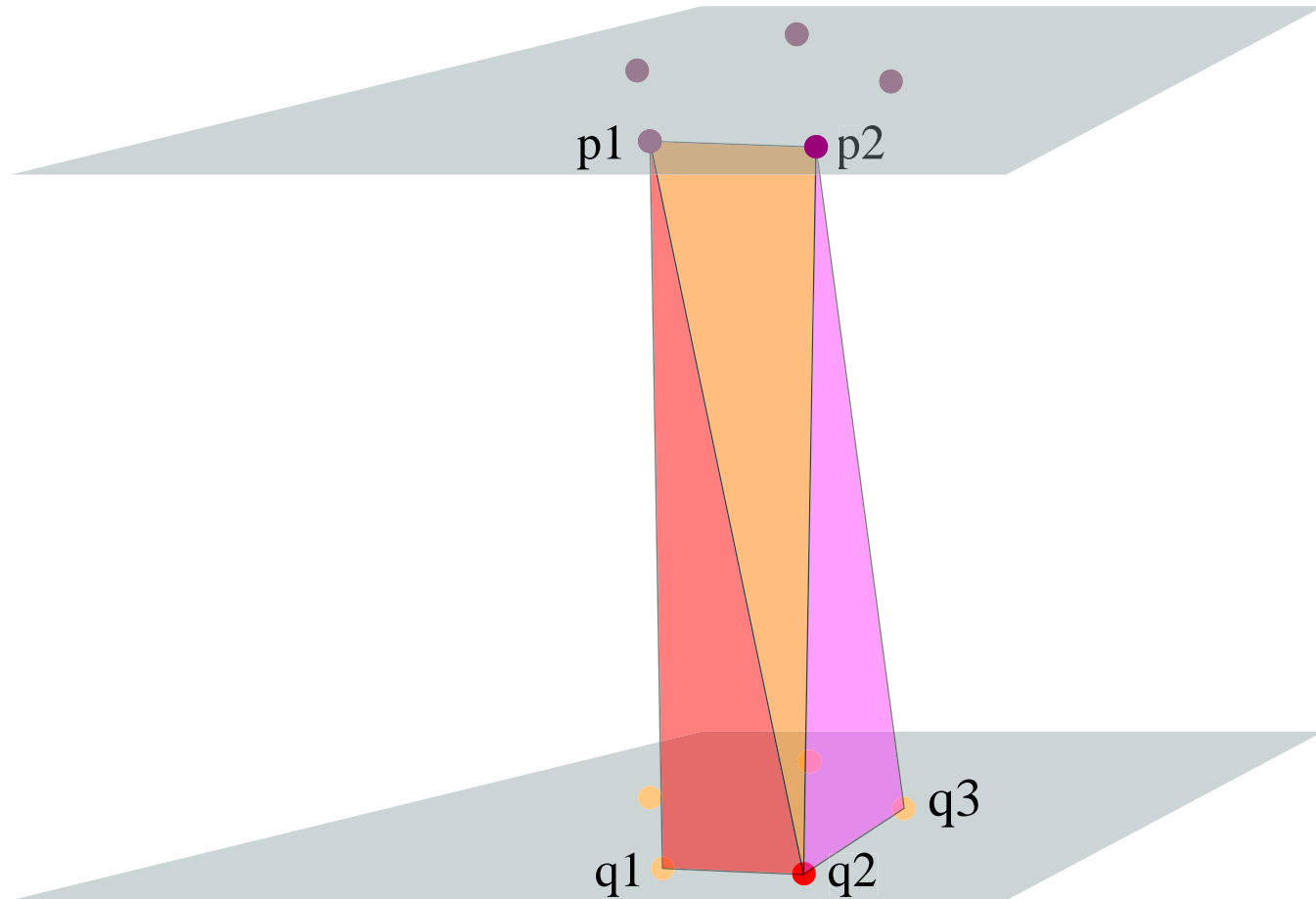
Go with Option 2 – 1



Option 2 – 1 - 1



Option 2 -1 -2



Greedy Solution

Works for regular polygons, but not in general

Brute Force Recursion (Only Returns Cost)

```
function findTriangles(Triangles, allPoints, currentPoints):  
    // Base case  
    if currentPoints == allPoints[end]:  
        return (Area of all Triangles)  
  
    if (Enough points left on top):  
        option1 = Triangles + [triangle with 2 points on top]  
        cost1 = findTriangles(option1, points, move current top point over)  
  
    if (Enough points left on the bottom):  
        option2 = Triangles + [triangle with 2 points on bottom]  
        cost2 = findTriangles(option2, points, move current bottom point over)  
  
    return min(cost1, cost2)
```

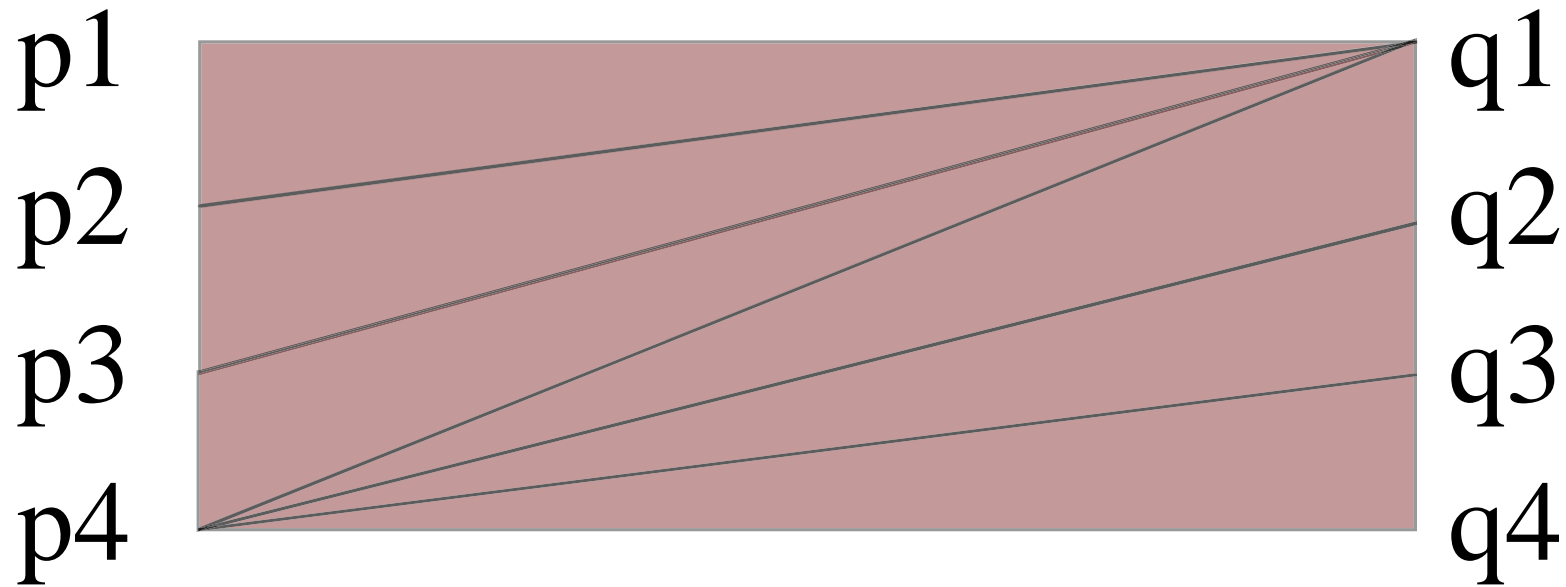
Note:

allPoints in an ordered 2D array
where the starting points are
repeated at the end of the
array

Recursion Branch Leaf Example

TOP

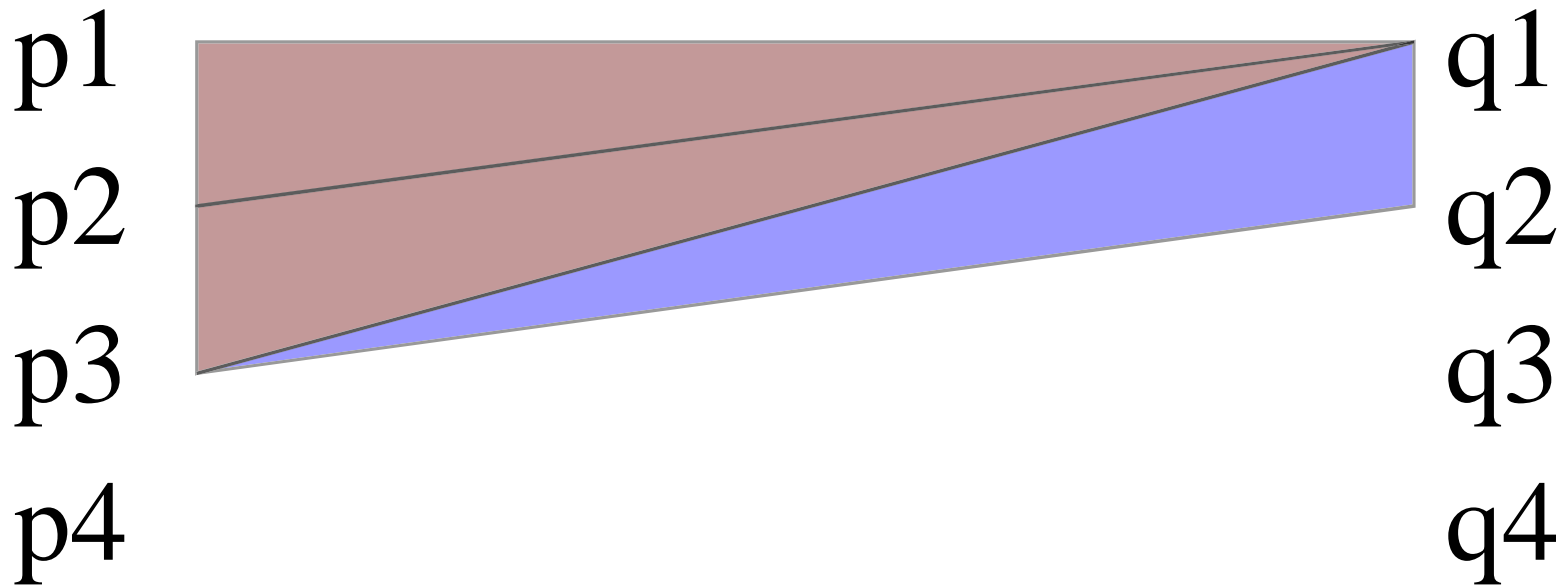
BOTTOM



A Different Branch

TOP

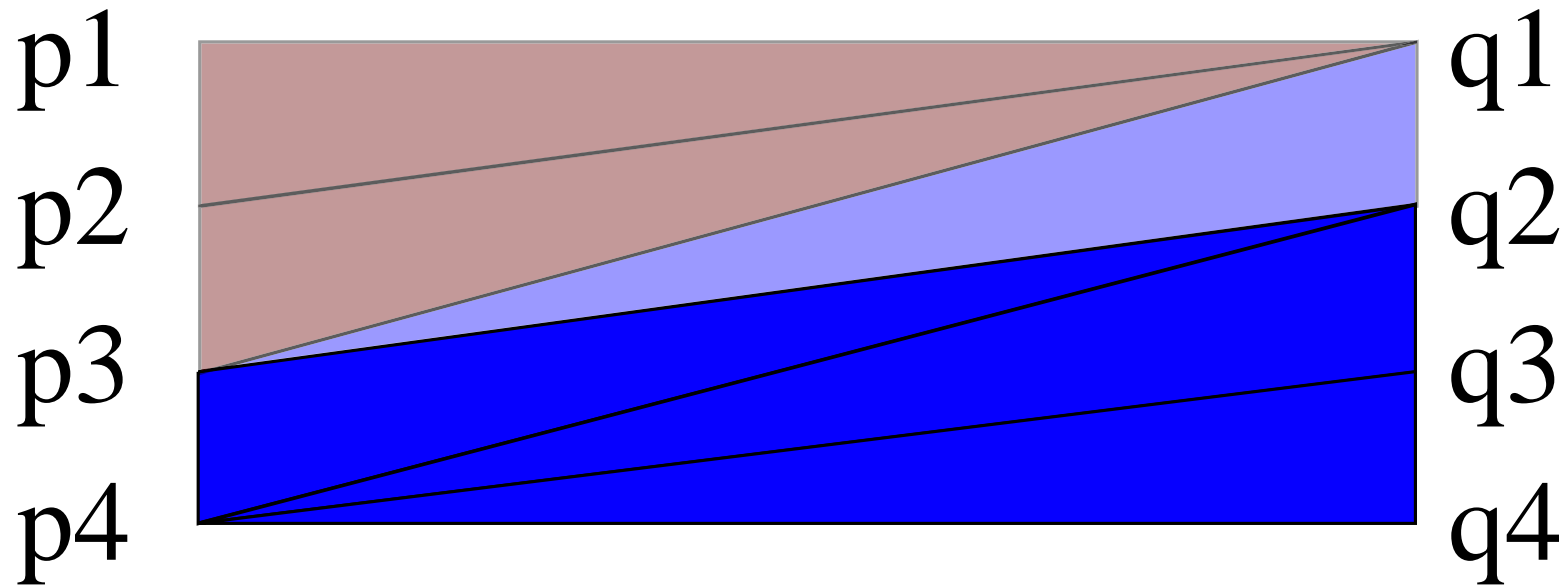
BOTTOM



The Other Branch's Corresponding Leaf

TOP

BOTTOM

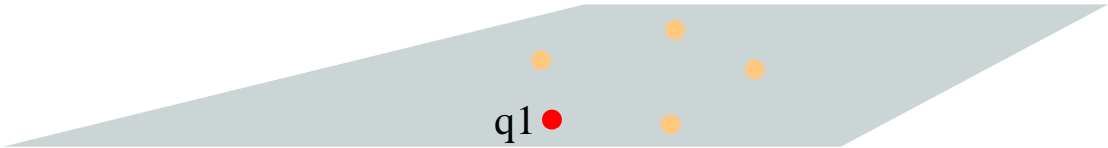
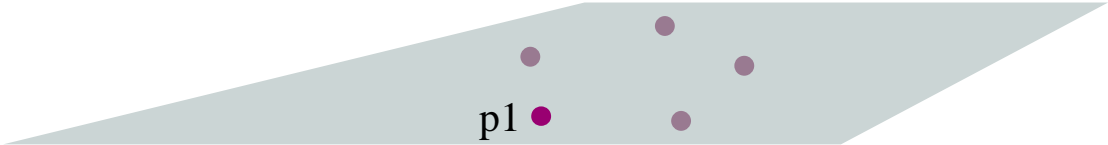


Recursion Issues

- Recursion is $O(2^n)$ assuming top and bottom points are aligned
- If you do not know which point to start with on top, it's $O(n2^n)$

Dynamic Programming

Toroidal Graph Method



	p1	p2	p3	p4	p5	p1
q1	●	●	●	●	●	●
q2	●	●	●	●	●	●
q3	●	●	●	●	●	●
q4	●	●	●	●	●	●
q5	●	●	●	●	●	●
q1	●	●	●	●	●	●

Directed Edges on the Graph

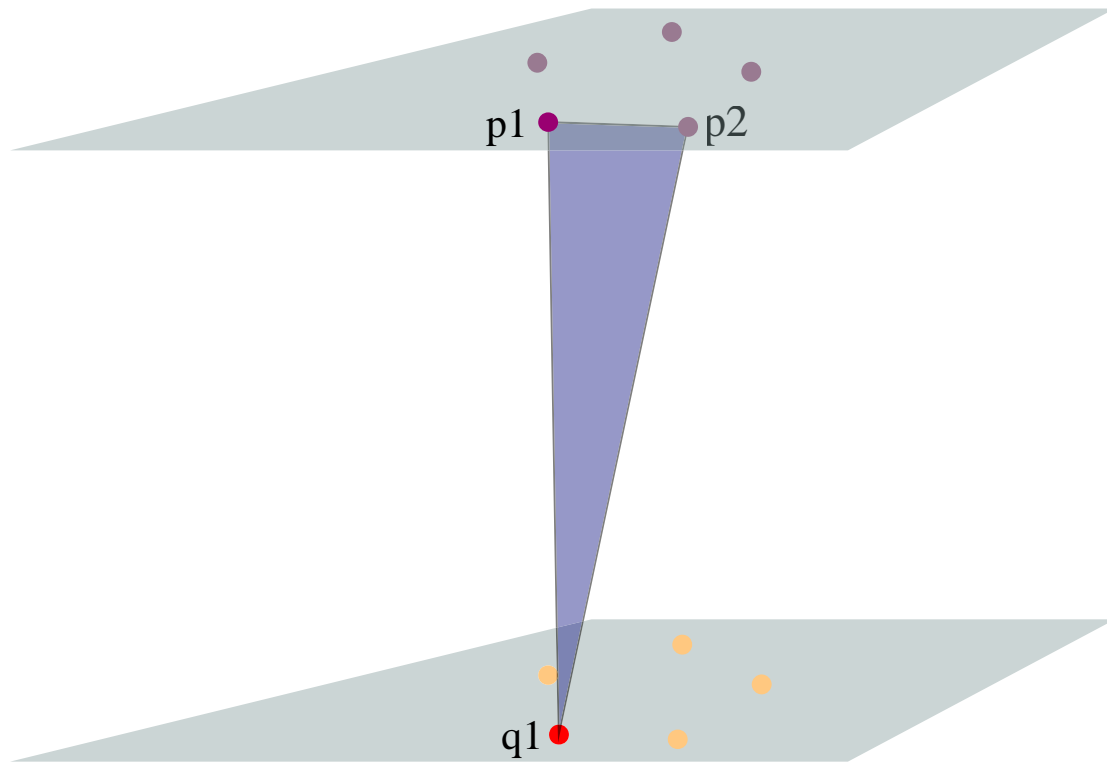
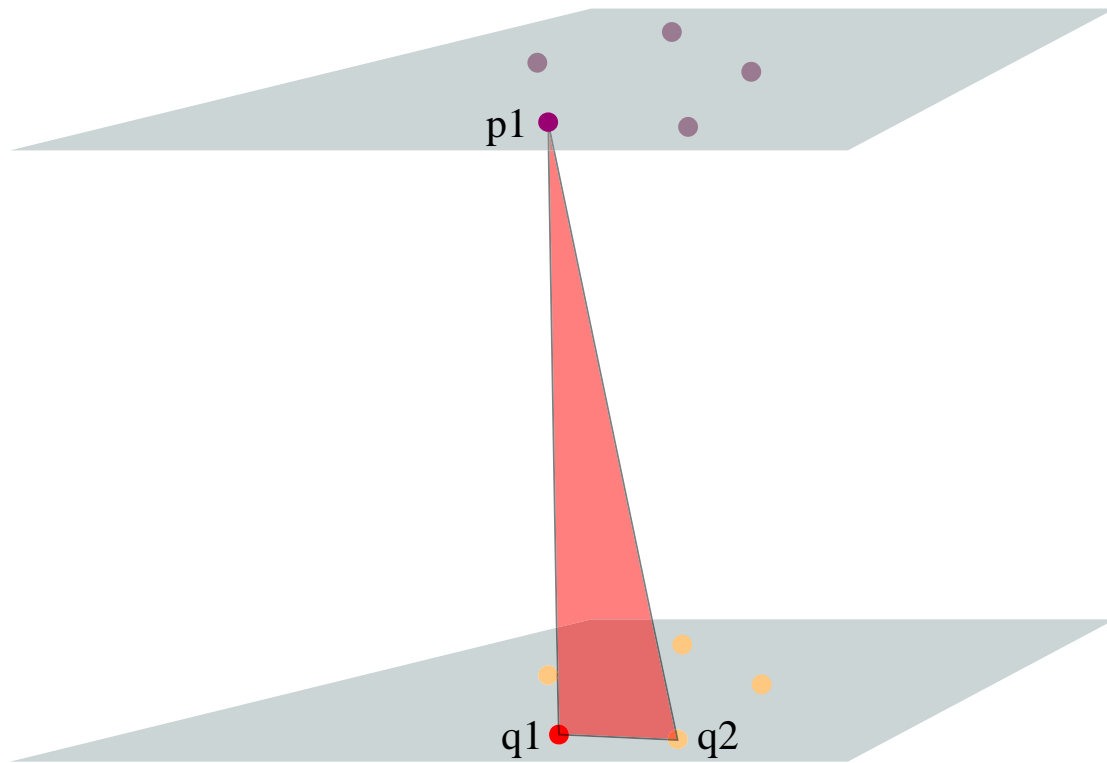






































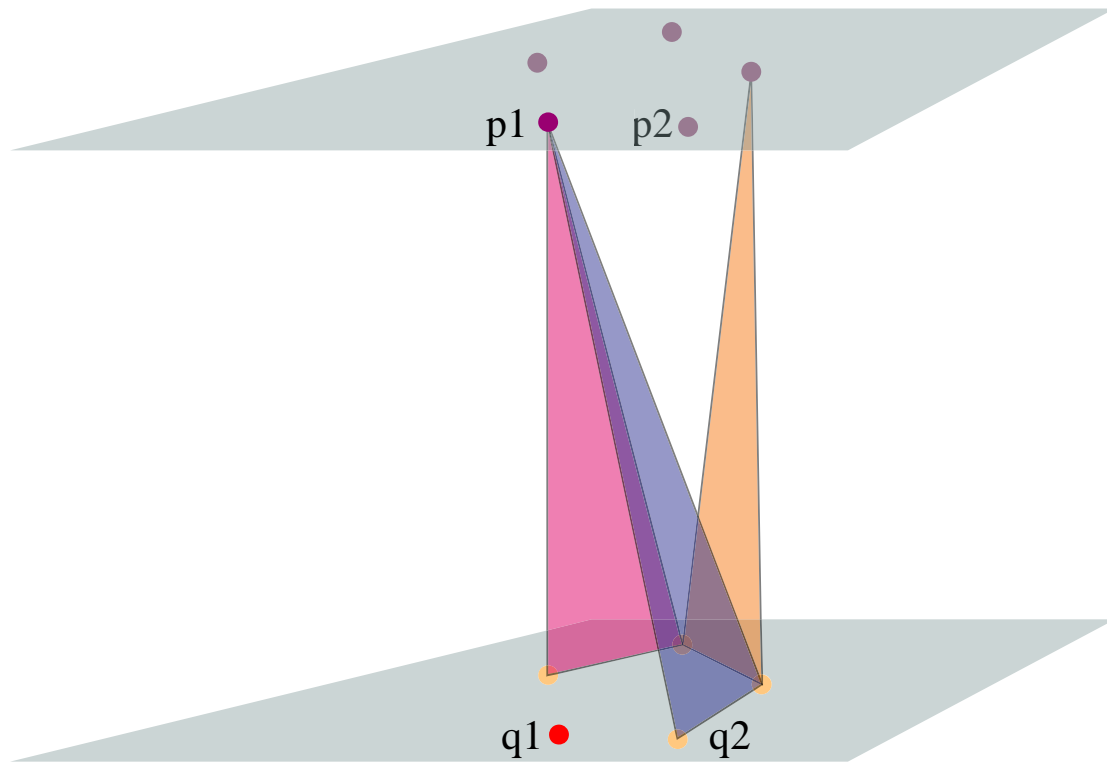
Diagram illustrating a 2D lattice structure with points labeled $q1, q2, q3, q4, q5$ (rows) and $p1, p2, p3, p4, p5$ (columns). A red circle highlights the point at $(q1, p1)$, and a blue arrow points from it to the point at $(q1, p2)$. The points in the bottom row (labeled $q1$) are gray, while the others are black.

Right or Down



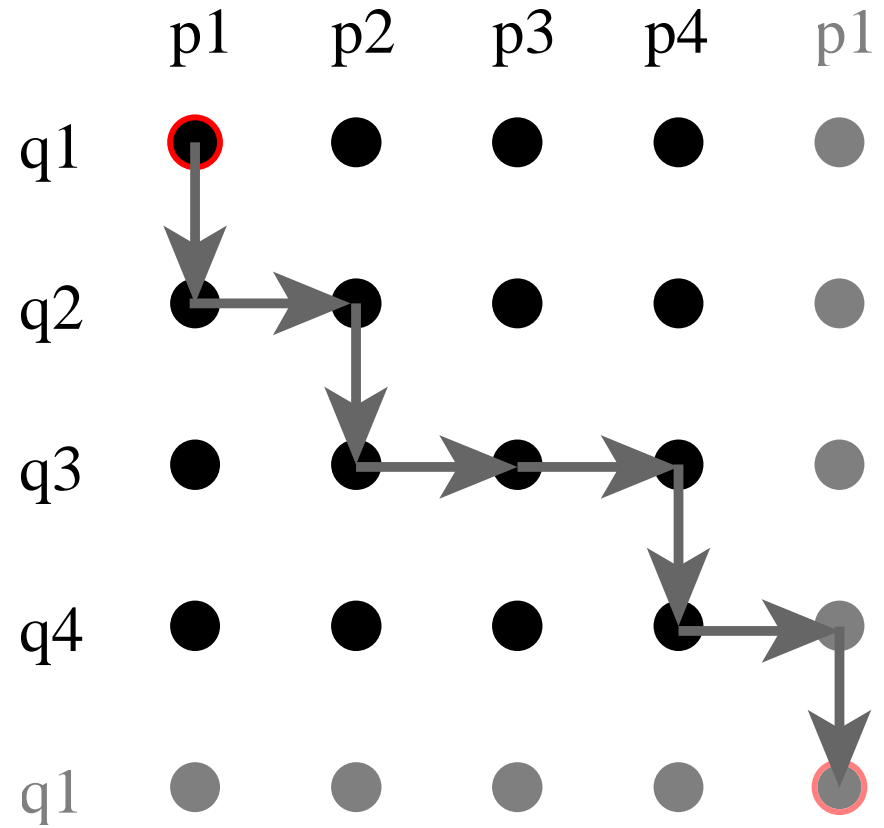
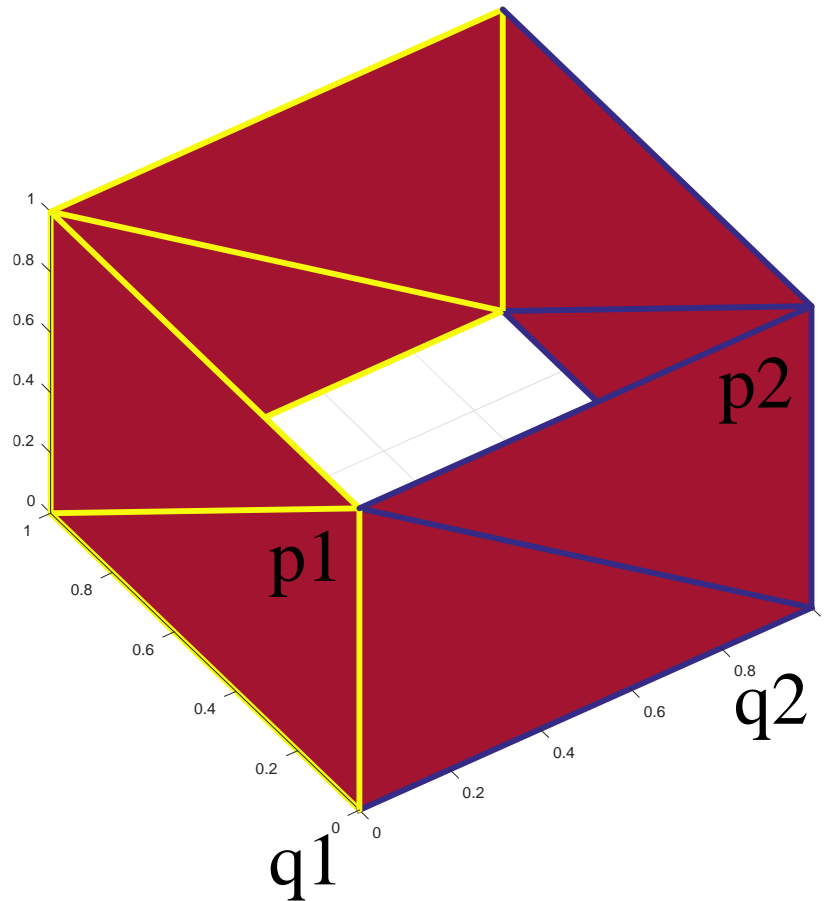
	p1	p2	p3	p4	p5	p1
q1						
q2						
q3						
q4						
q5						
q1						

Which Triangle Does this Edge Represent?

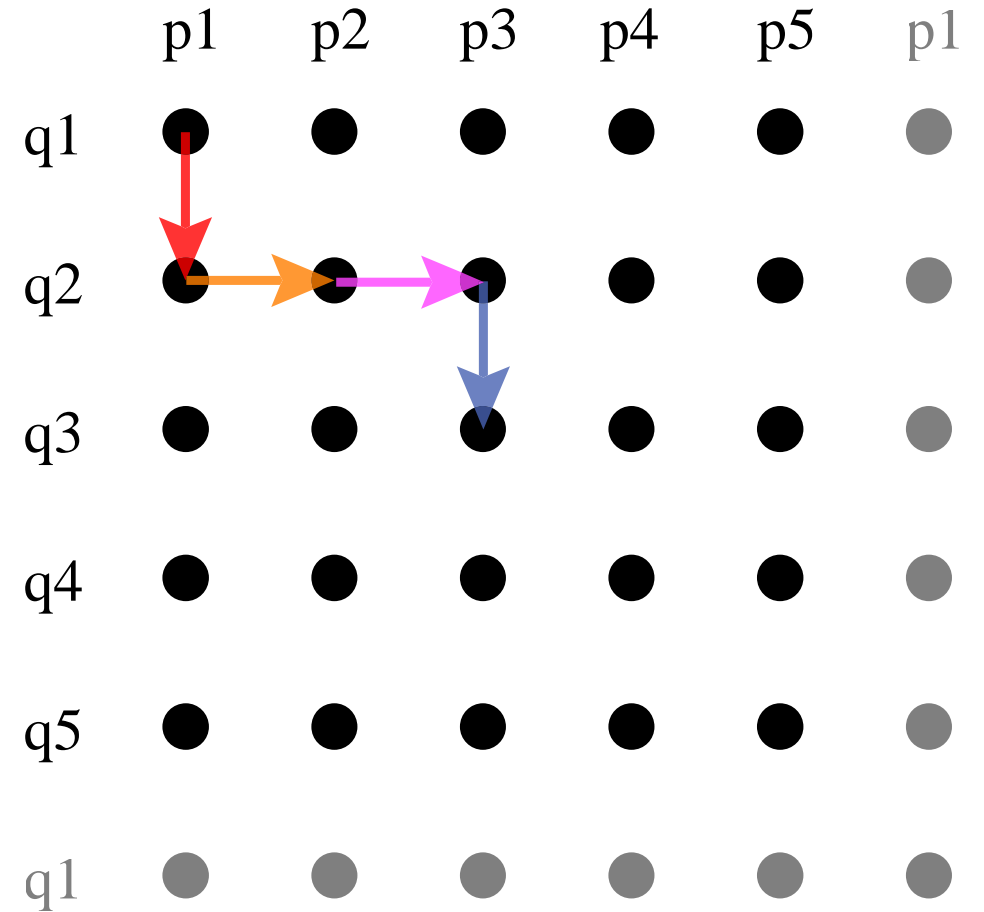
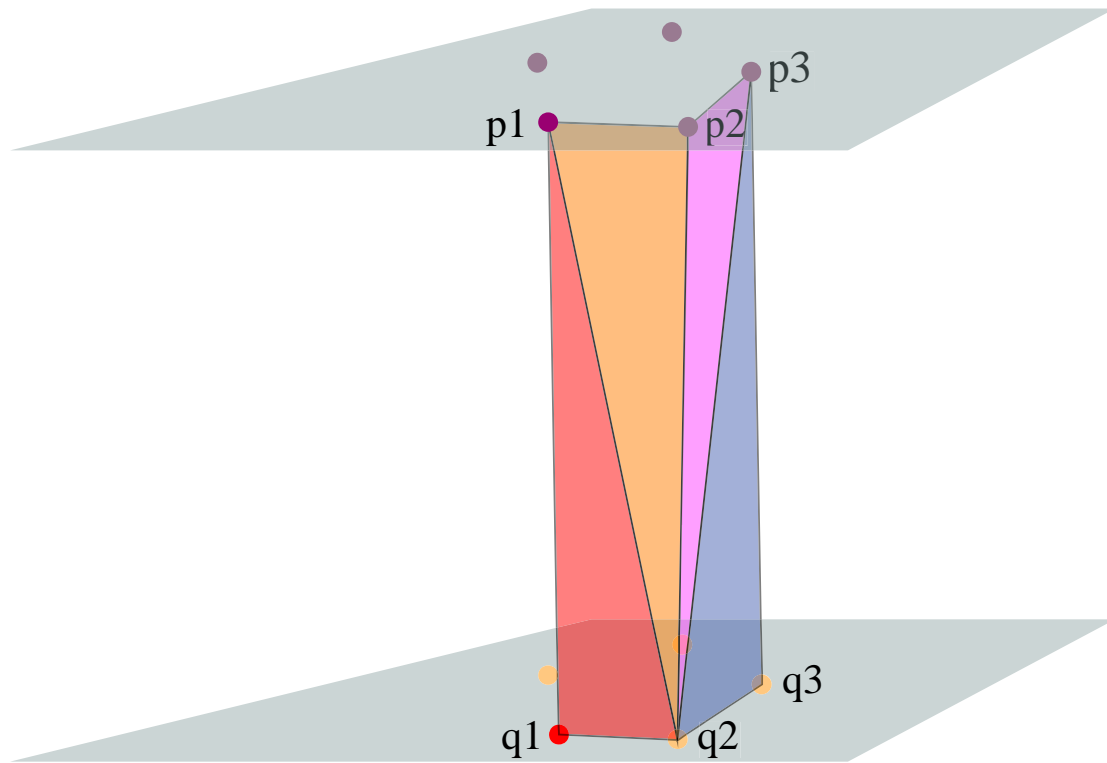


	$p1$	$p2$	$p3$	$p4$	$p5$	$p1$
$q1$	●	●	●	●	●	●
$q2$	●	●	●	●	●	●
$q3$	●	●	●	●	●	●
$q4$	●	●	●	●	●	●
$q5$	●	●	●	●	●	●
$q1$	●	●	●	●	●	●

A Full Path Represents a Solution



Find the Path With Least Cost



Minimum Cost Path

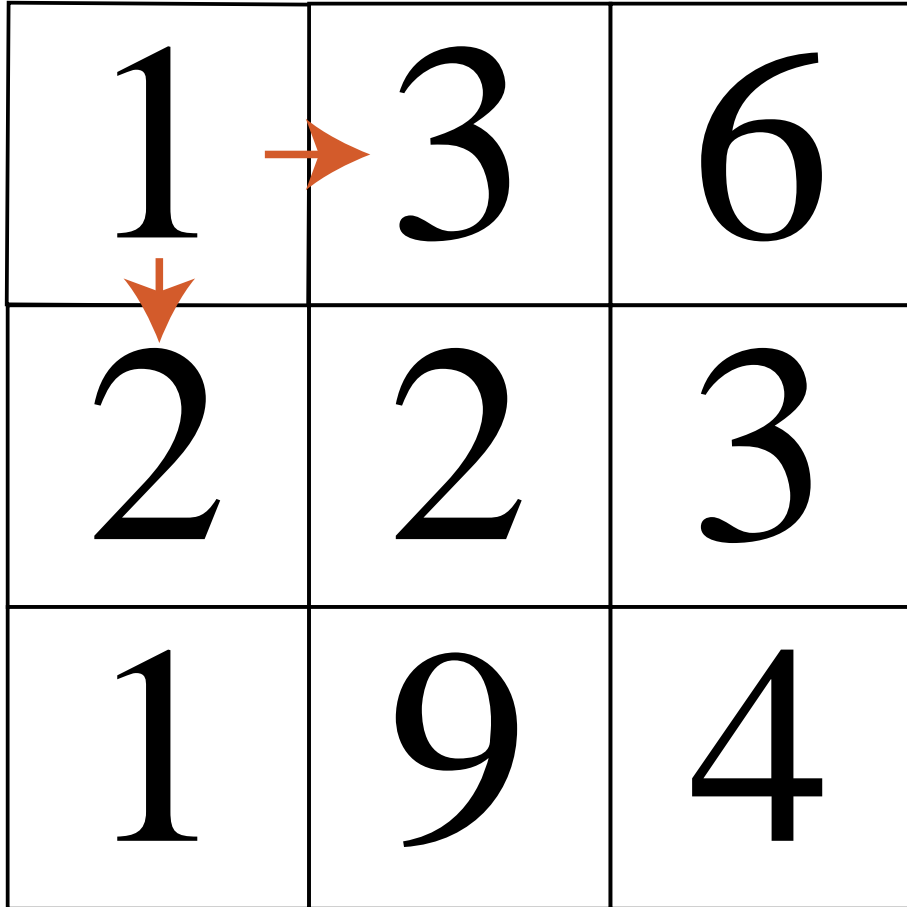
A Simple Example

Cost Matrix Example

1	3	6
2	2	3
1	9	4

Can Only Move Right or Down

1	3	6
2	2	3
1	9	4



The image shows a 3x3 grid of numbers. The top row contains 1, 3, and 6. The middle row contains 2, 2, and 3. The bottom row contains 1, 9, and 4. Two orange arrows originate from the number 1 in the top-left cell. One arrow points horizontally to the right, ending at the number 3 in the top-middle cell. The other arrow points vertically downwards, ending at the number 2 in the middle-left cell. This illustrates the movement rules: only right or down.

Find the Minimum Cost Path

1	3	6
2	2	3
1	9	4

Solution: Build a Path Cost Matrix


1	3	6
2	2	3
1	9	4

1	4	10
3	5	
4		

Greedy Algorithm Finds Path

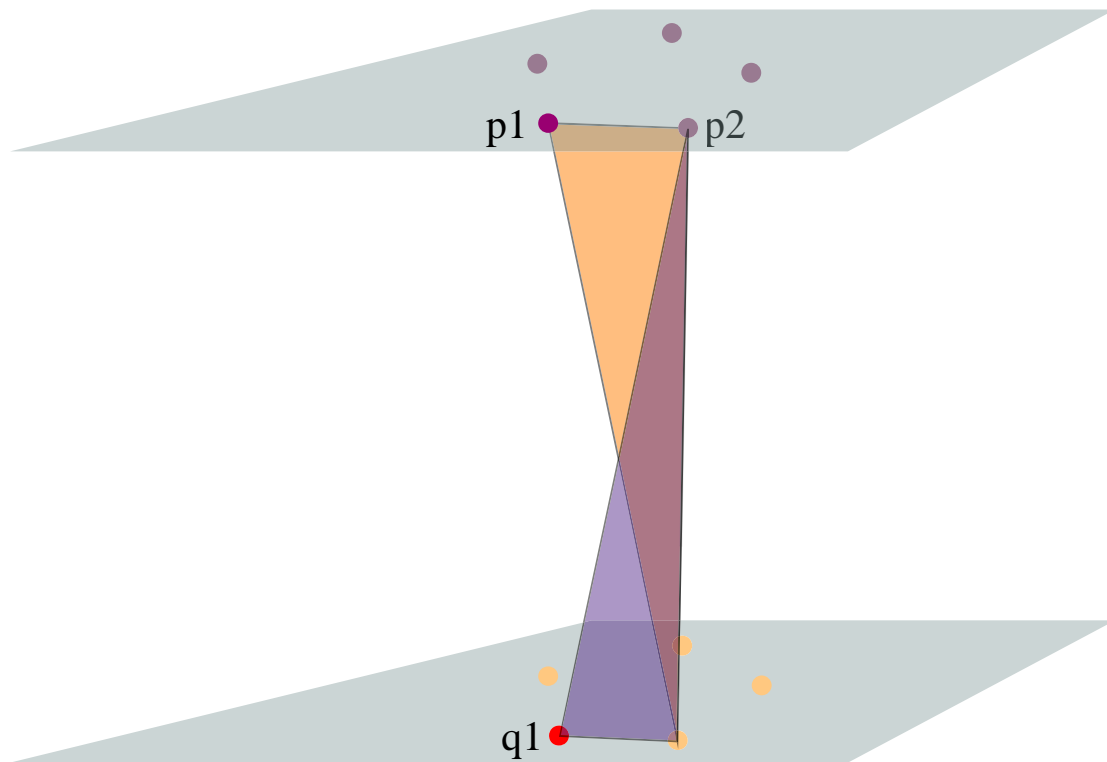
1	3	6
2	2	3
1	9	4

1	4	10
3	5	8
4	13	12



Minimum Cost Path for Toroidal Graph

Previous Algorithm Will not Work



	p1	p2	p3	p4	p5	p1
q1	●	●	●	●	●	●
q2	●	●	●	●	●	●
q3	●	●	●	●	●	●
q4	●	●	●	●	●	●
q5	●	●	●	●	●	●
q1	●	●	●	●	●	●

Programming Assignment 3 - Conclusion

- Find an algorithm to compute the minimum cost path on the toroidal graph
- Run the algorithm for any cyclic permutation of the top points
- Make sure that the path ends at (p_1, q_1)