CS 130B—Data Structures and Algorithms II

Discussion Section Week 7

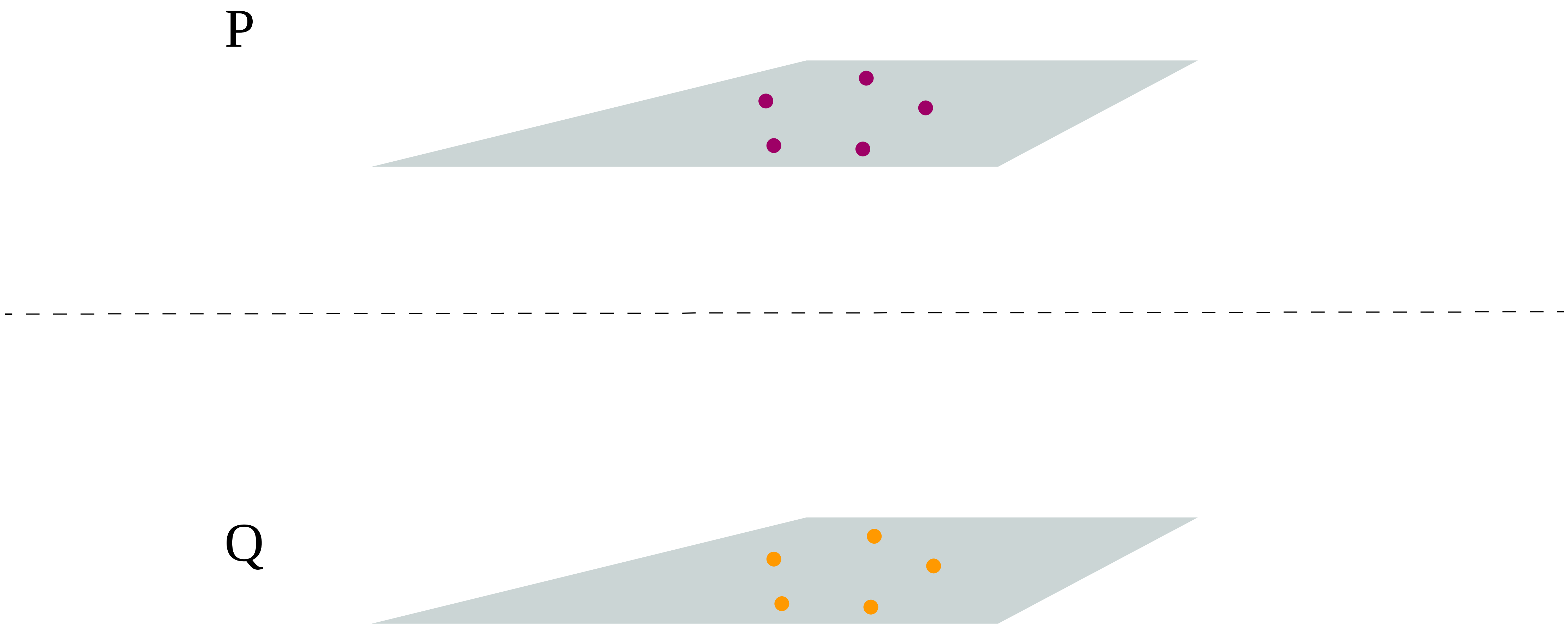
Written Assignment 3

- Due Wednesday May 24^{th} at 4pm

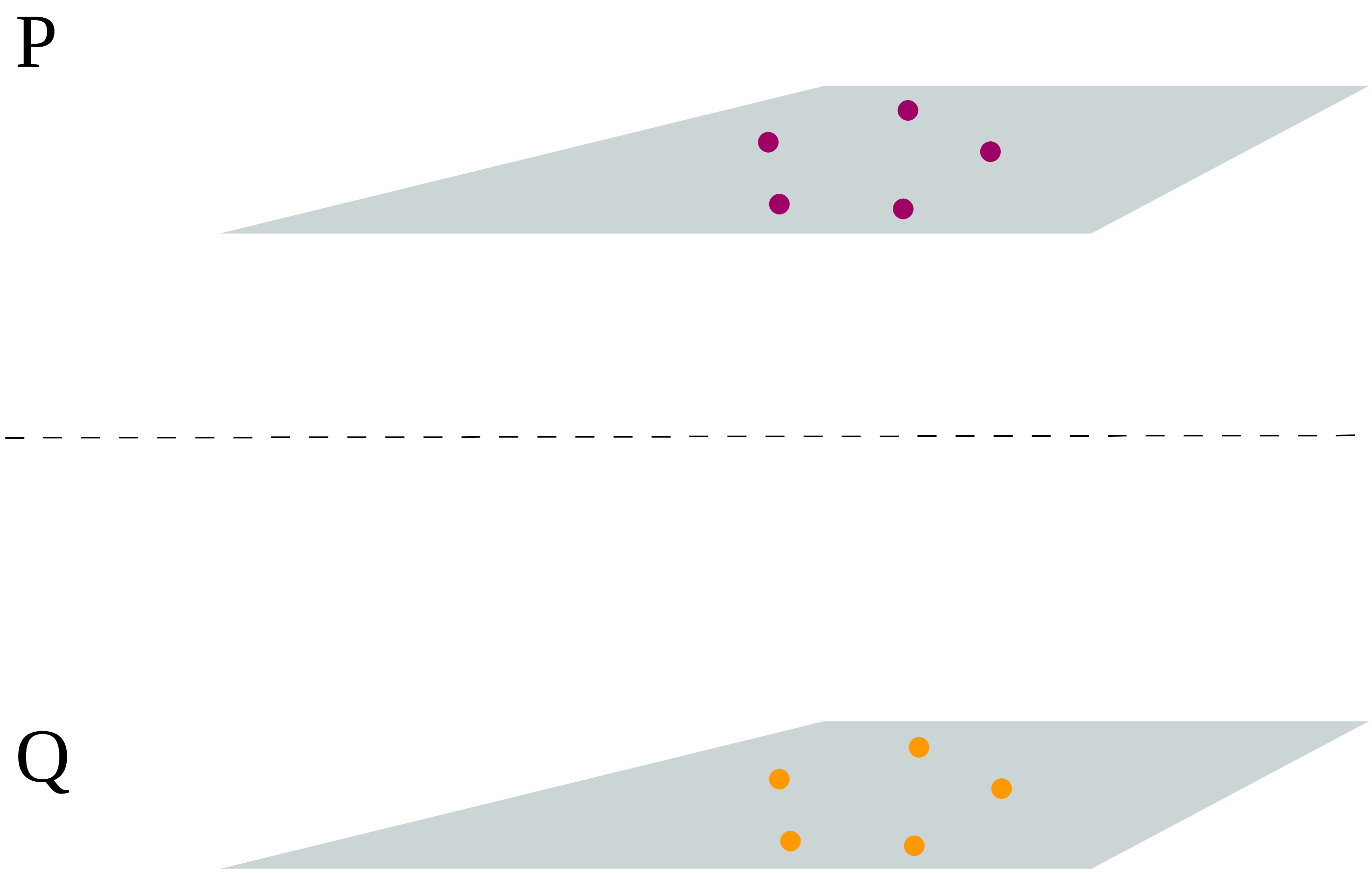
Programming Assignment 3

- Due Wednesday May 31st at 11:59pm

Programming Assignment 3



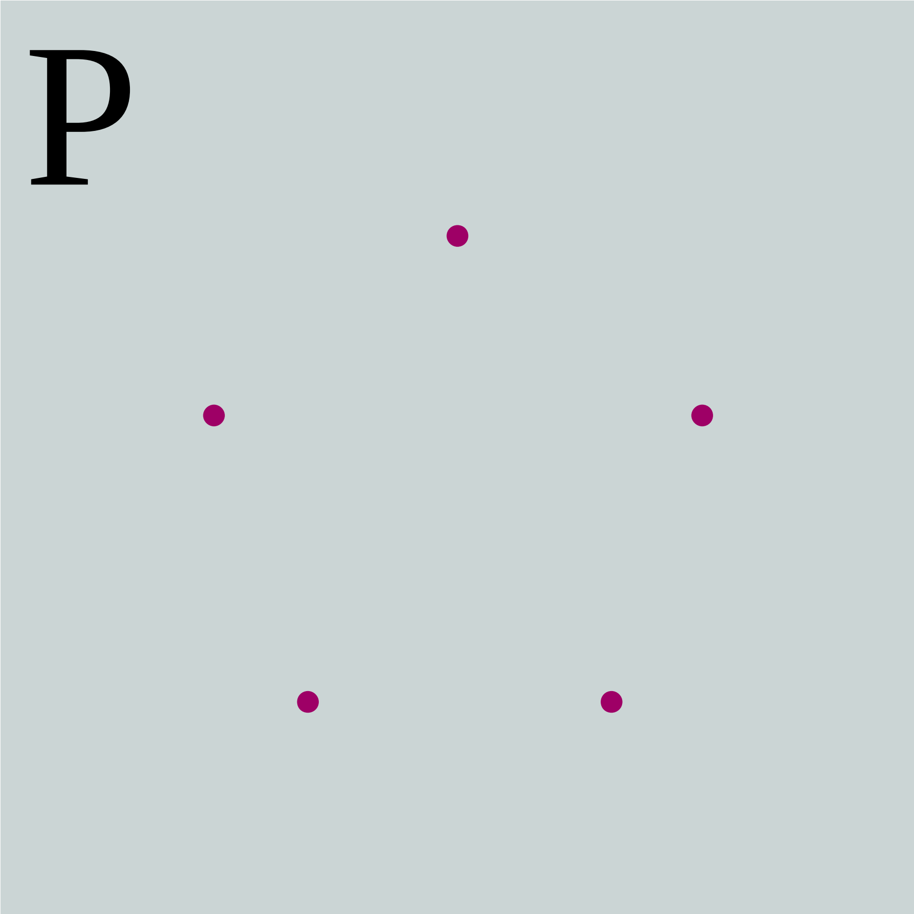
File Format



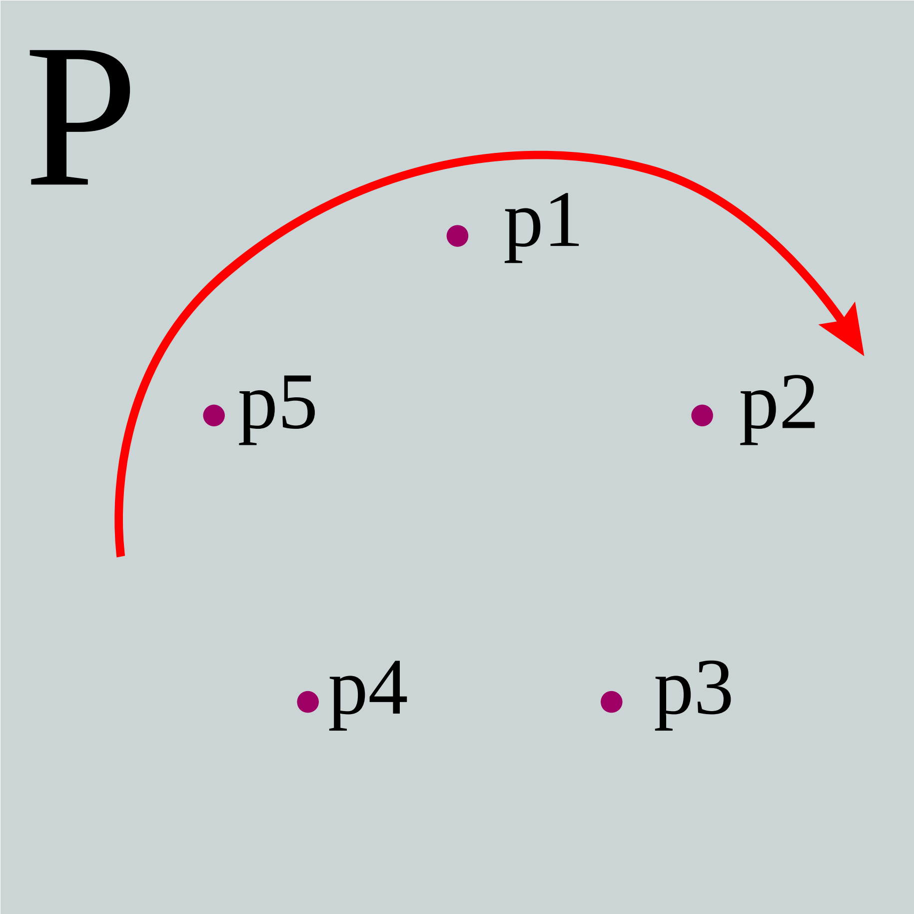
input.txt

```
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
```

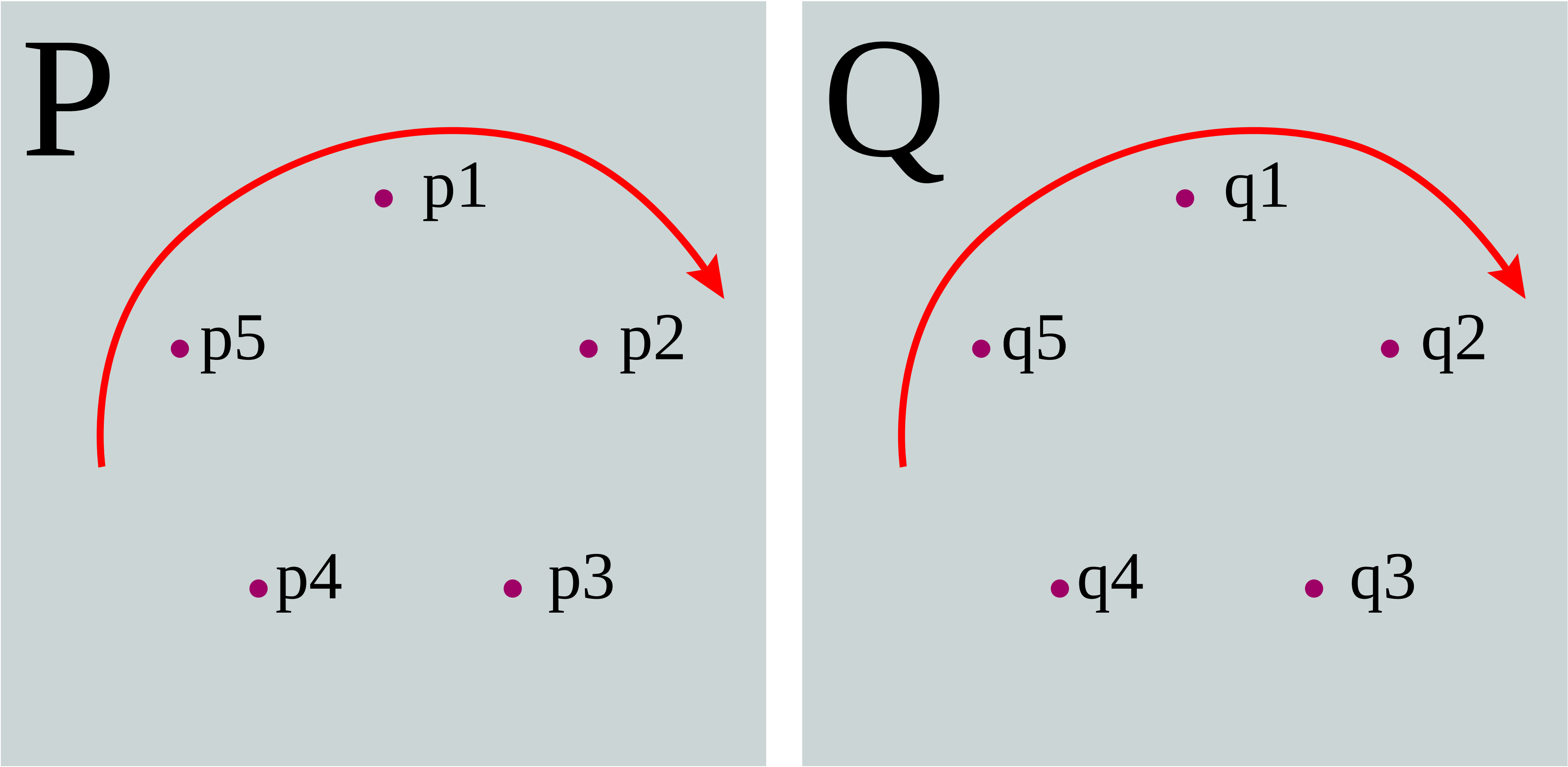
Top View



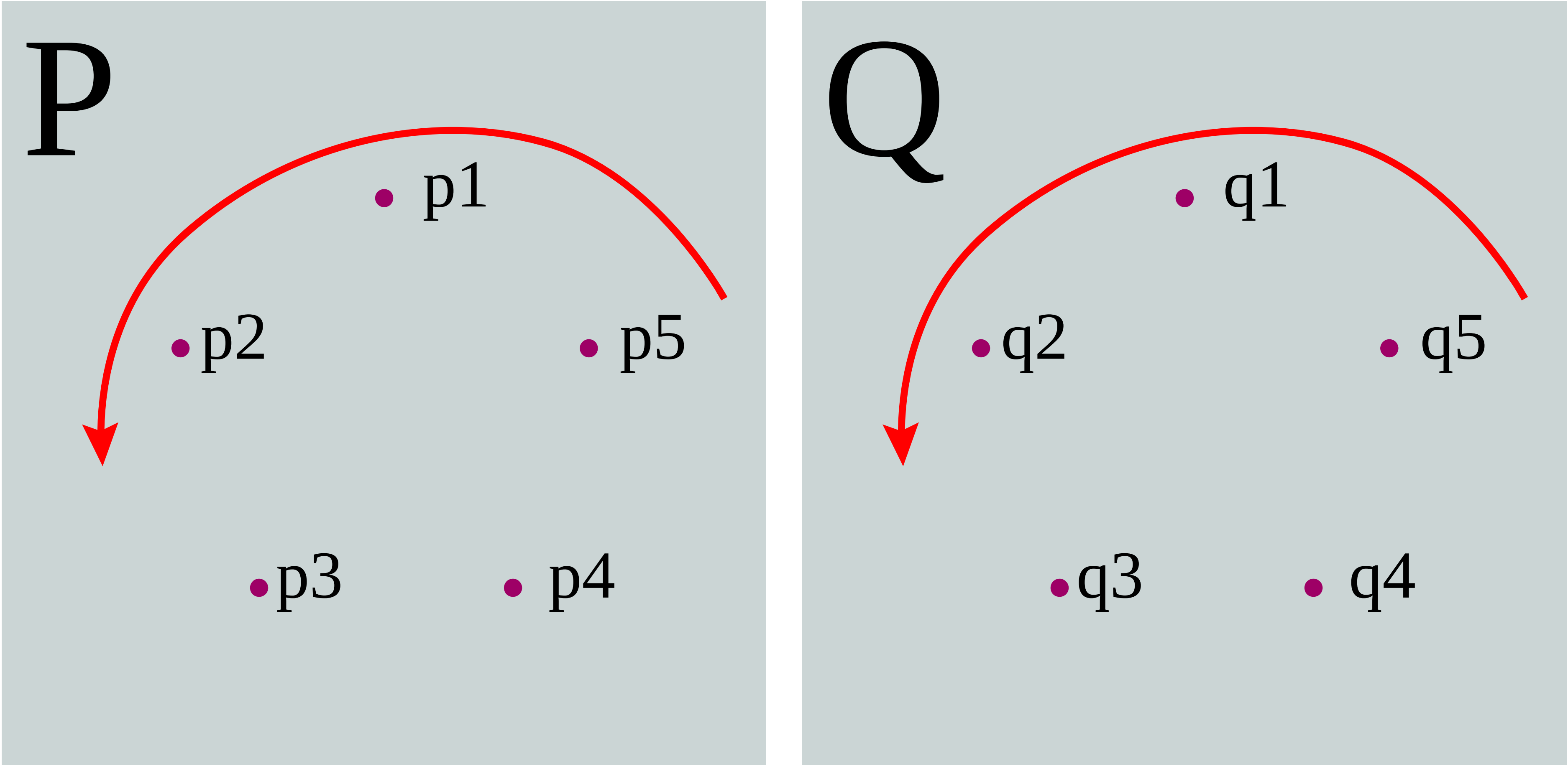
Points Are In Order



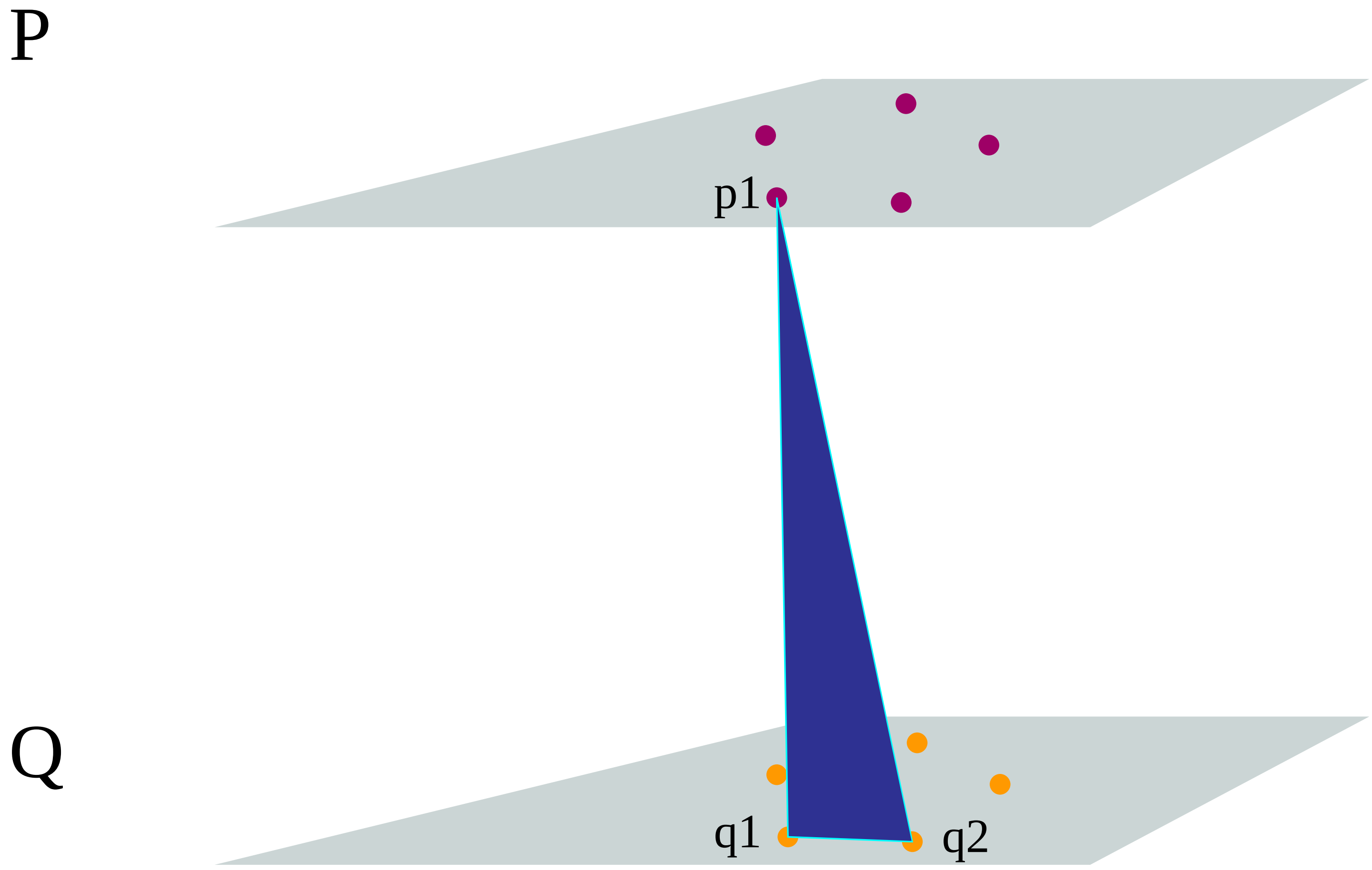
Same Order Applies Top and Bottom



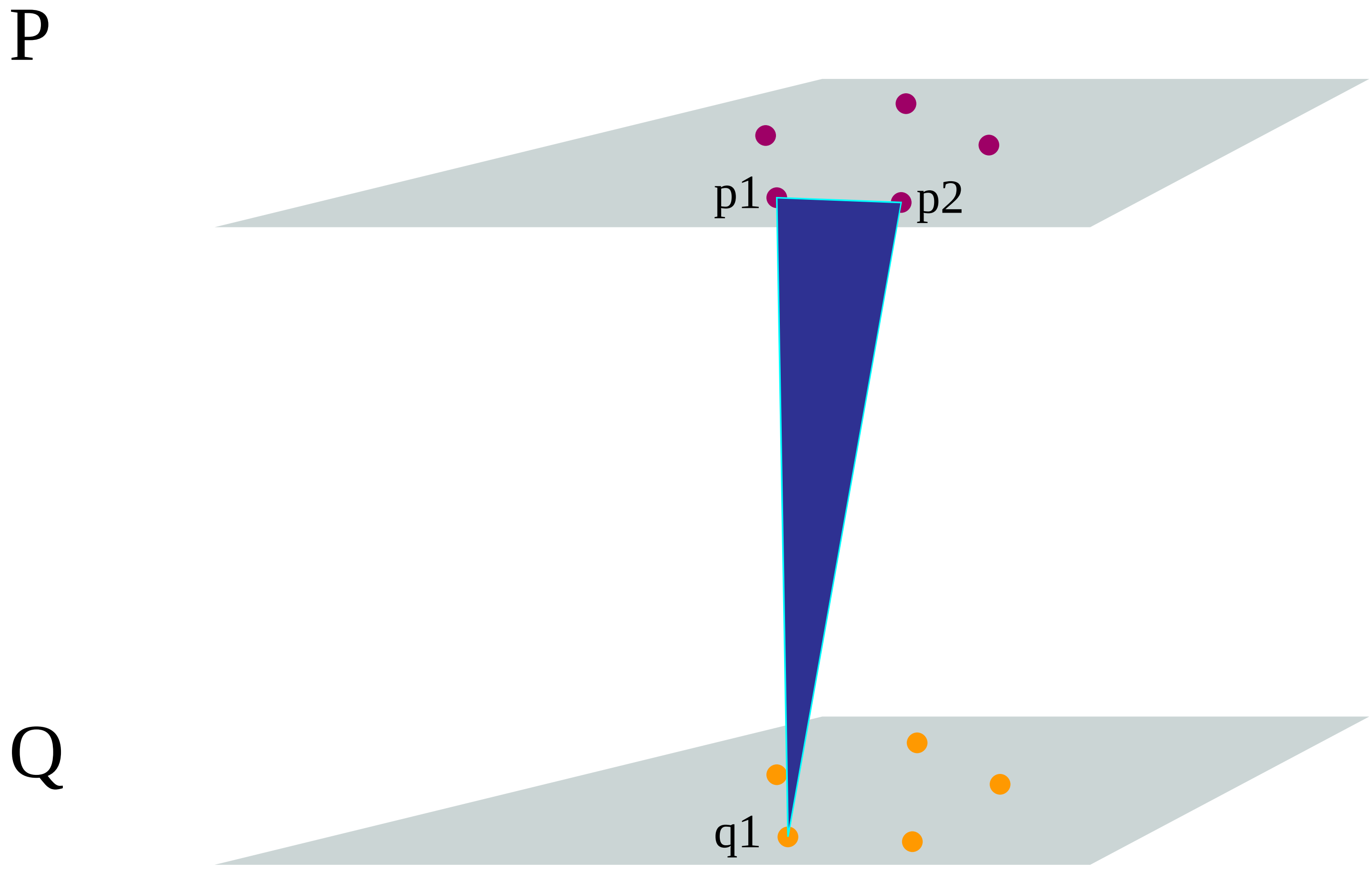
Or..



Finding Triangles



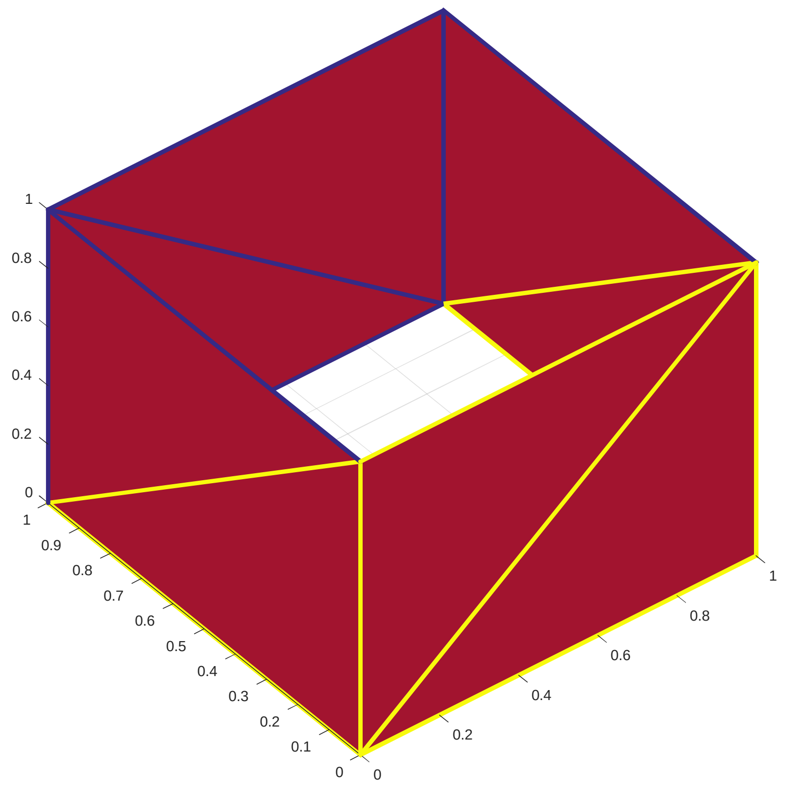
Need to Minimize Area of Surface



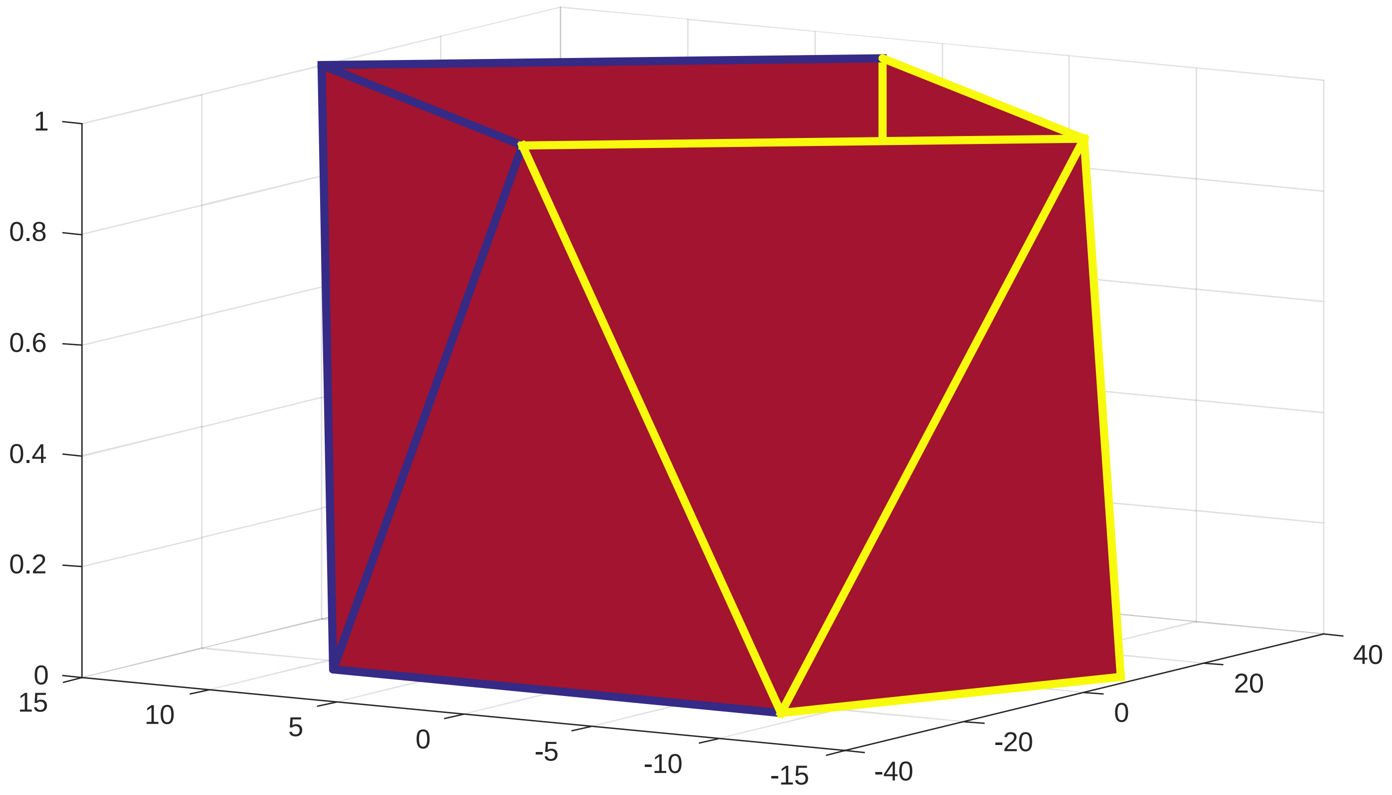
Output Examples

Plots from the Test Data

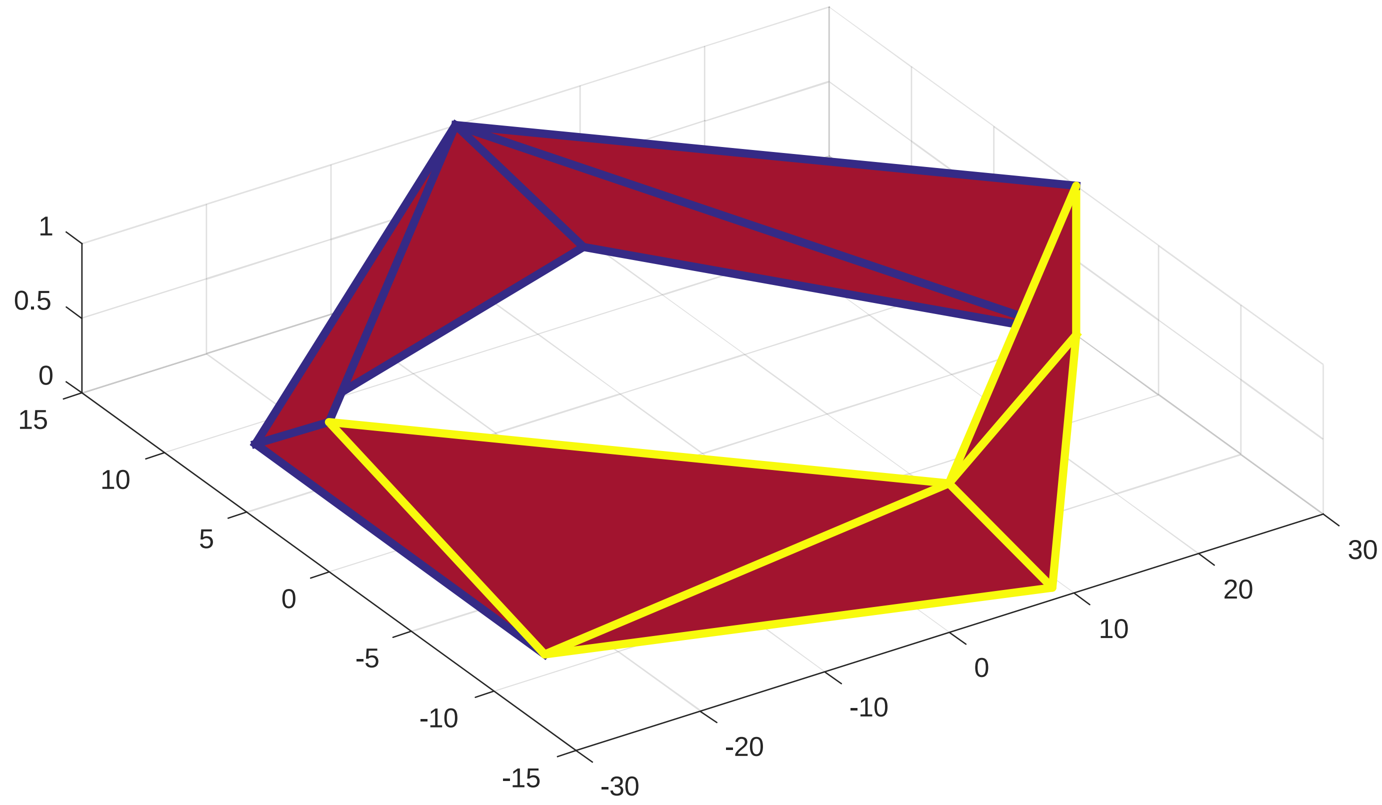
Two Squares



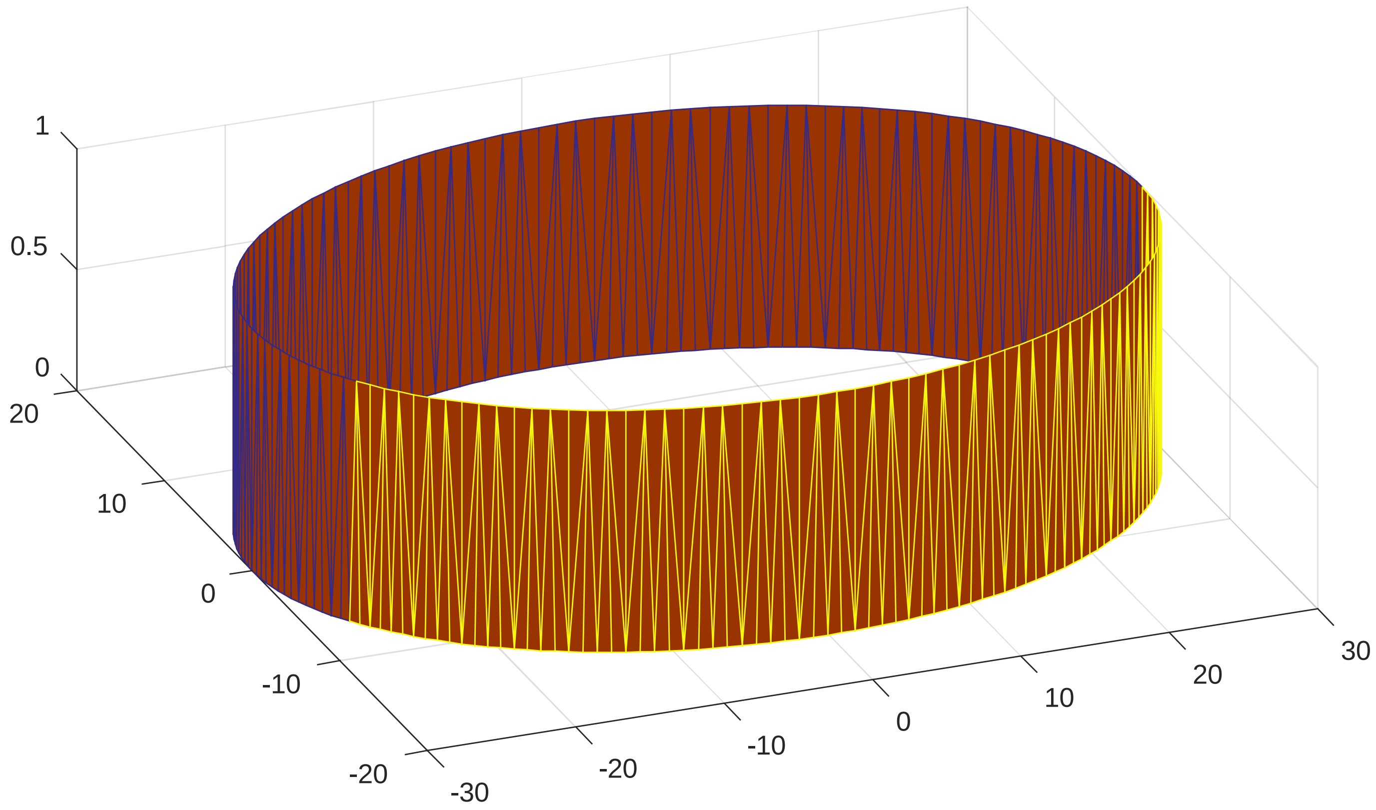
Two Different Samples of an Ellipse



Two Different Samples of an Ellipse—View2



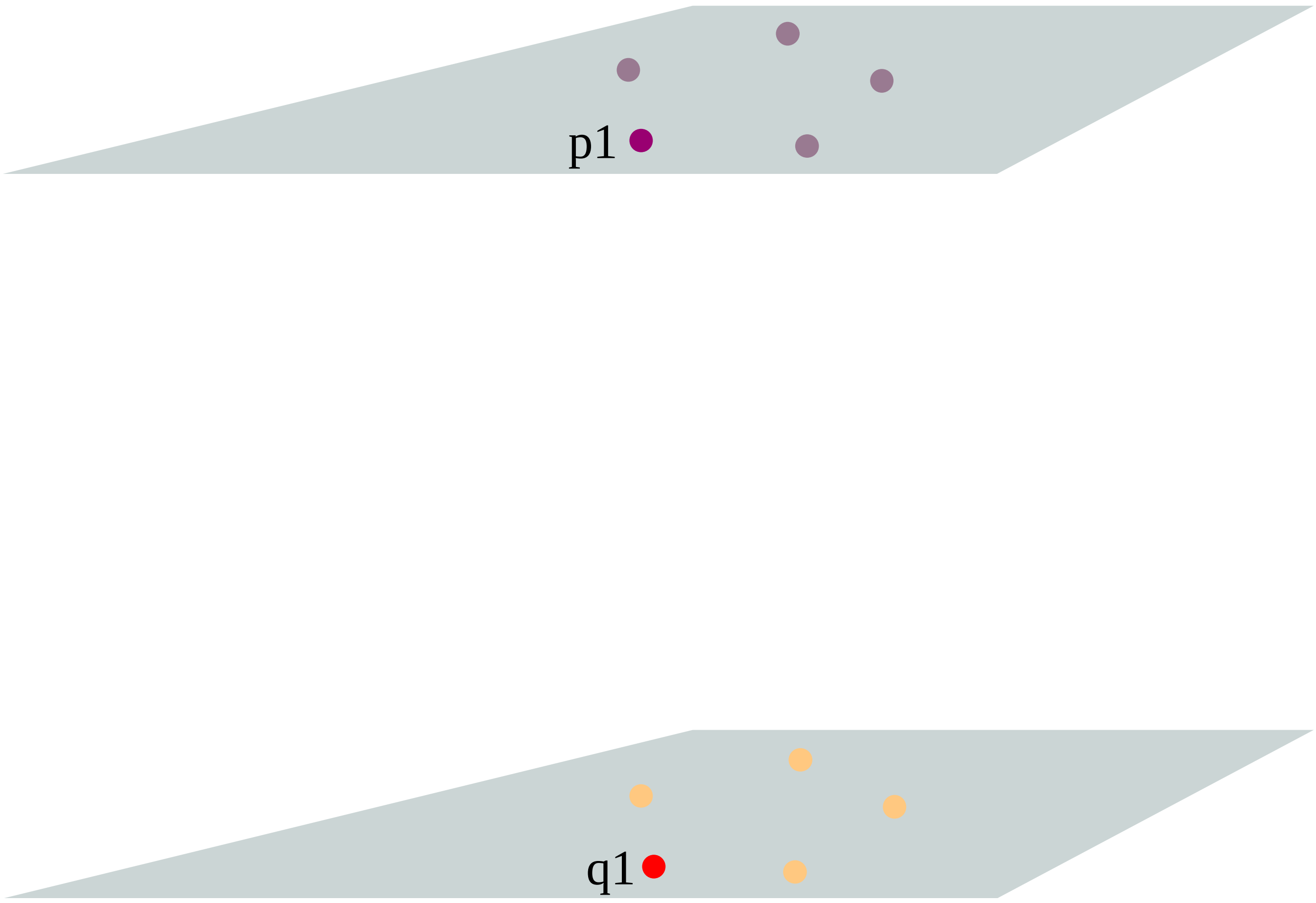
Two Ellipses



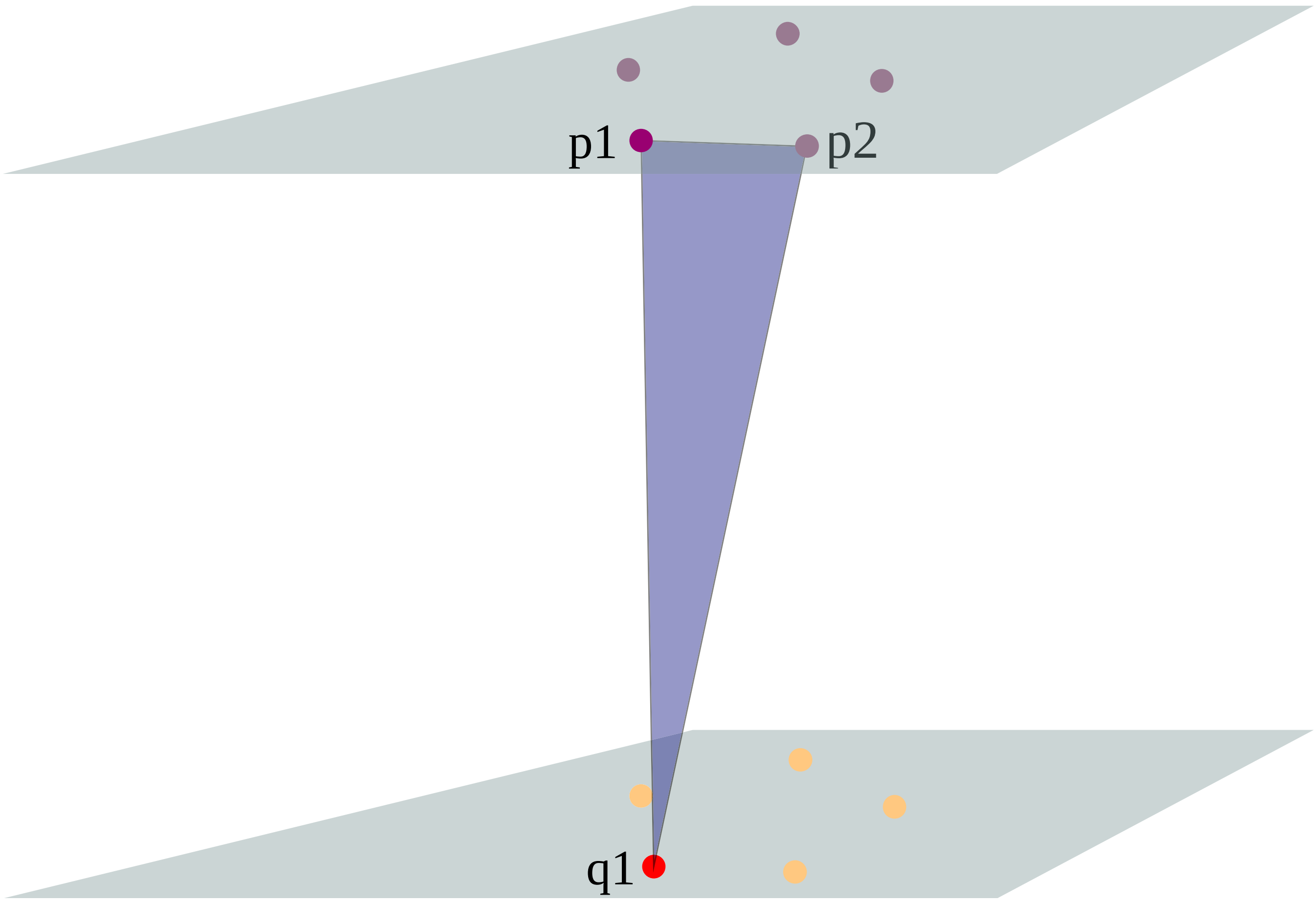
General Approach

A Recursive Solution

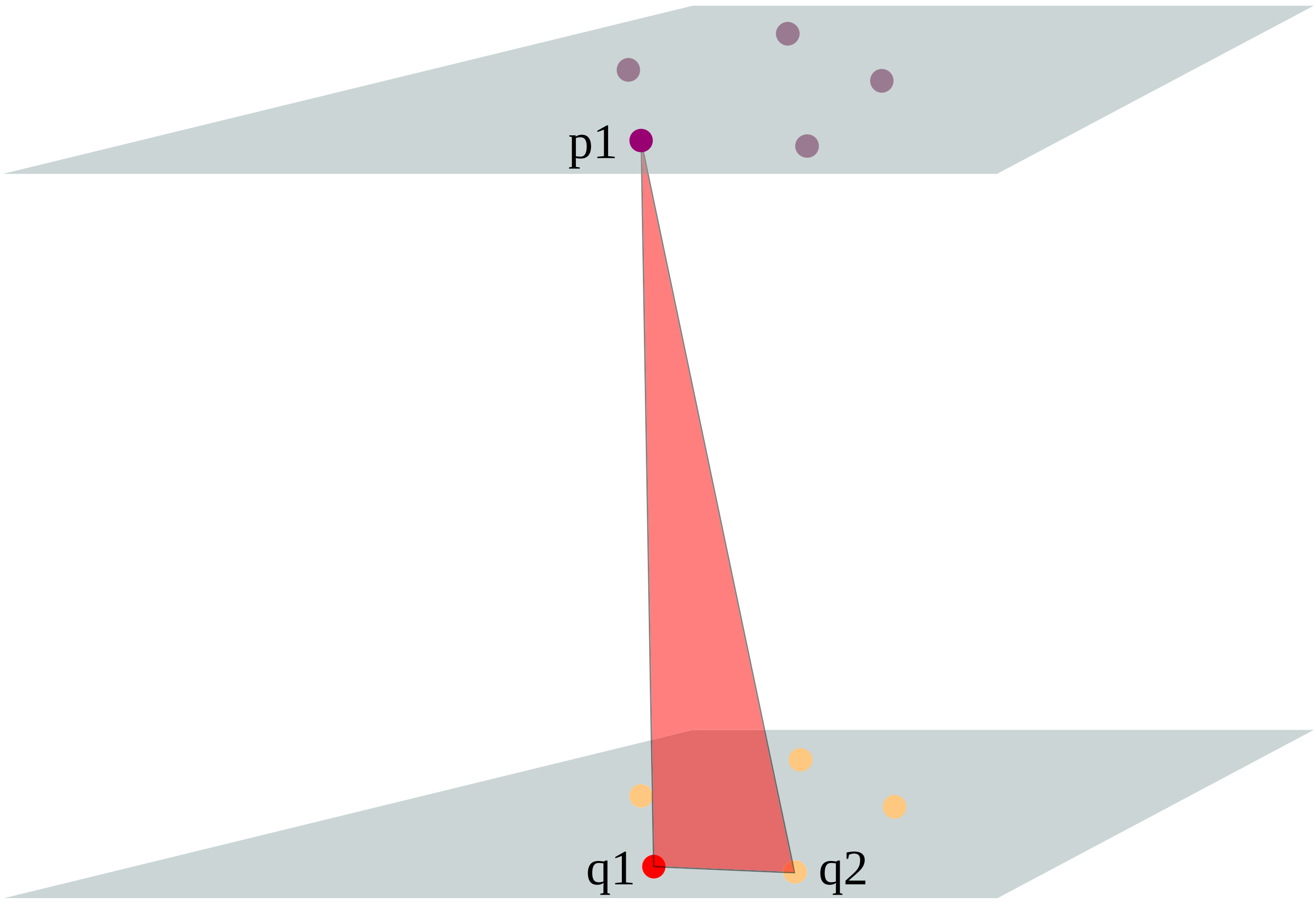
Starting at an Arbitrary Point



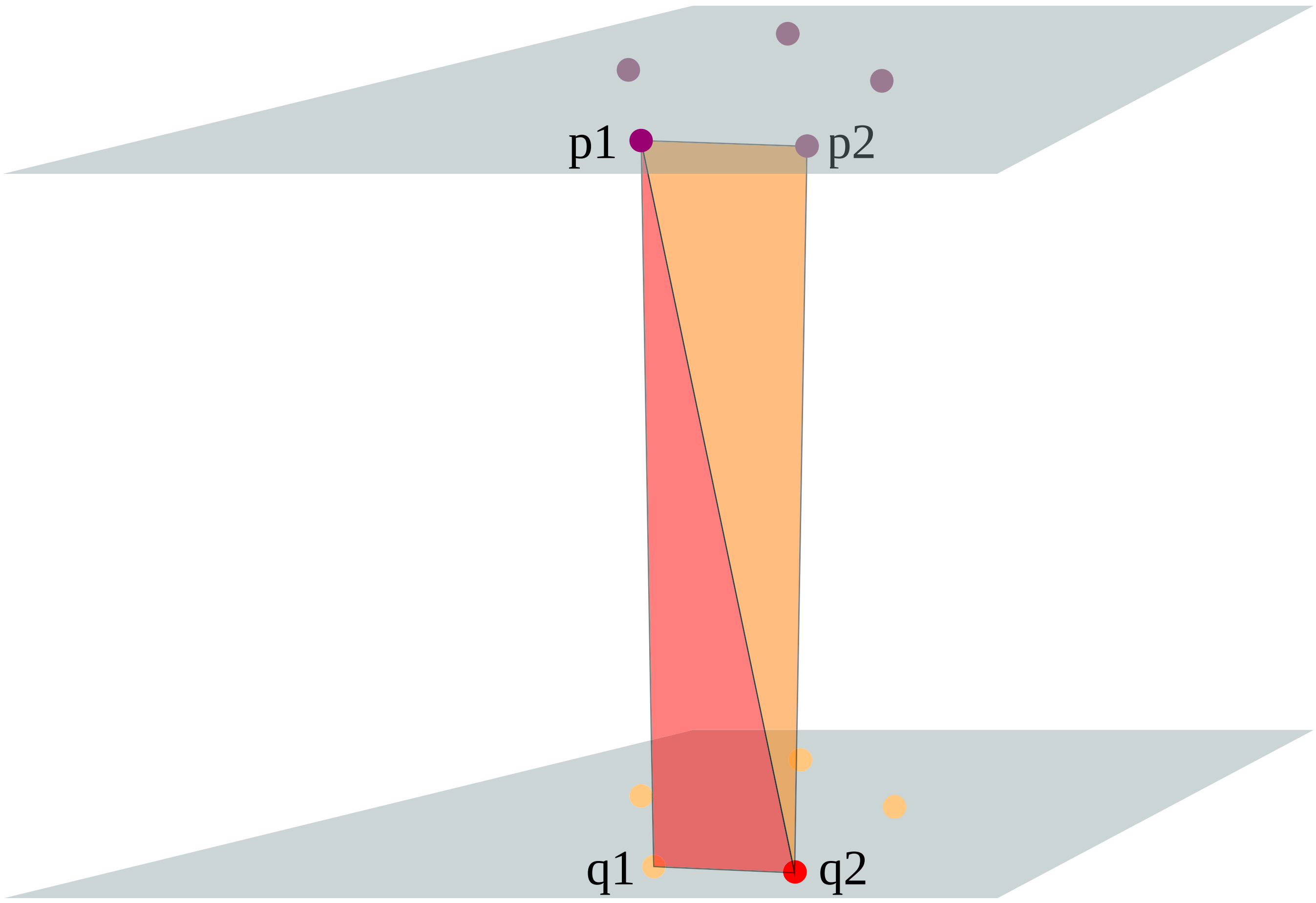
Triangle Option One



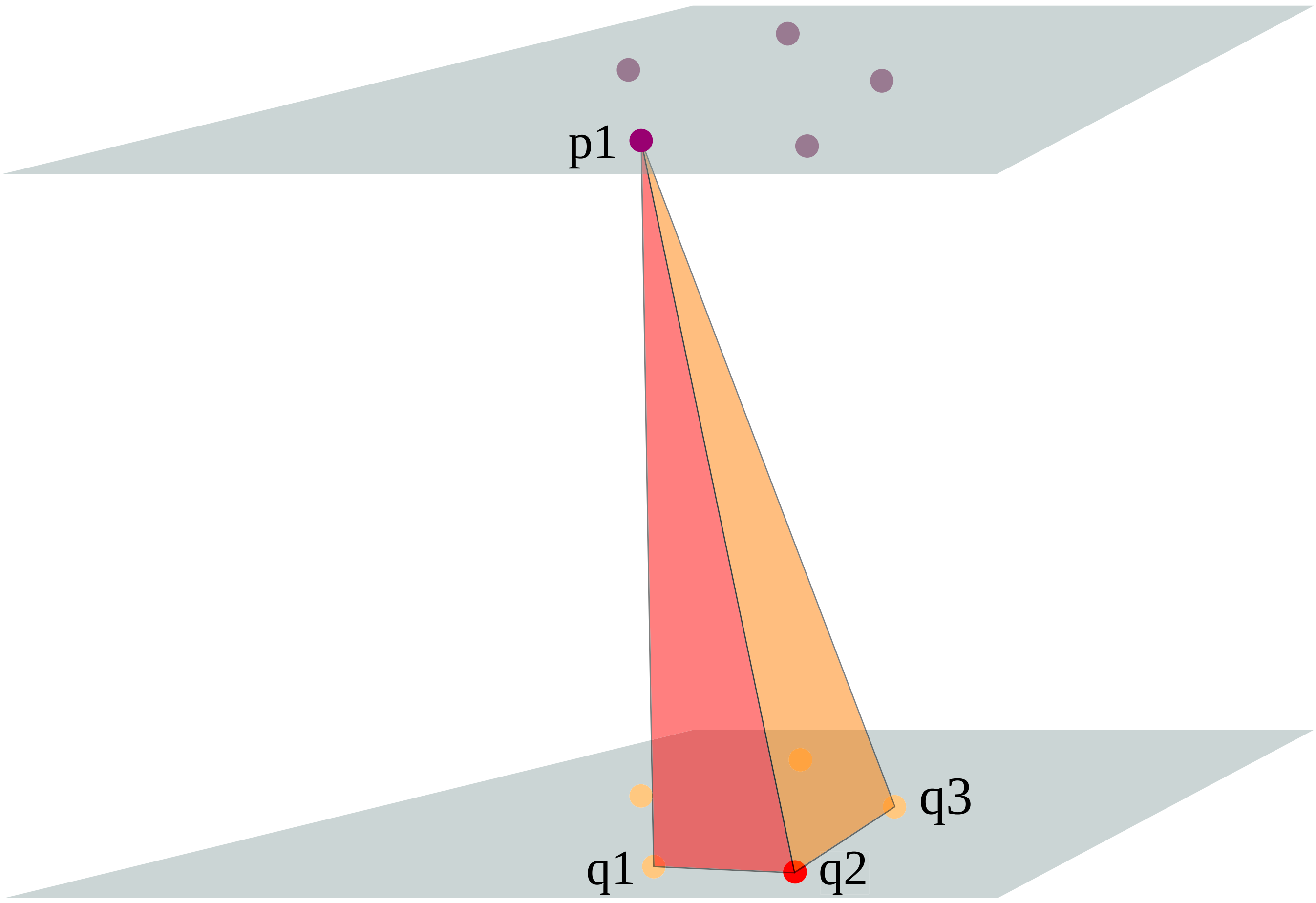
Triangle Option Two



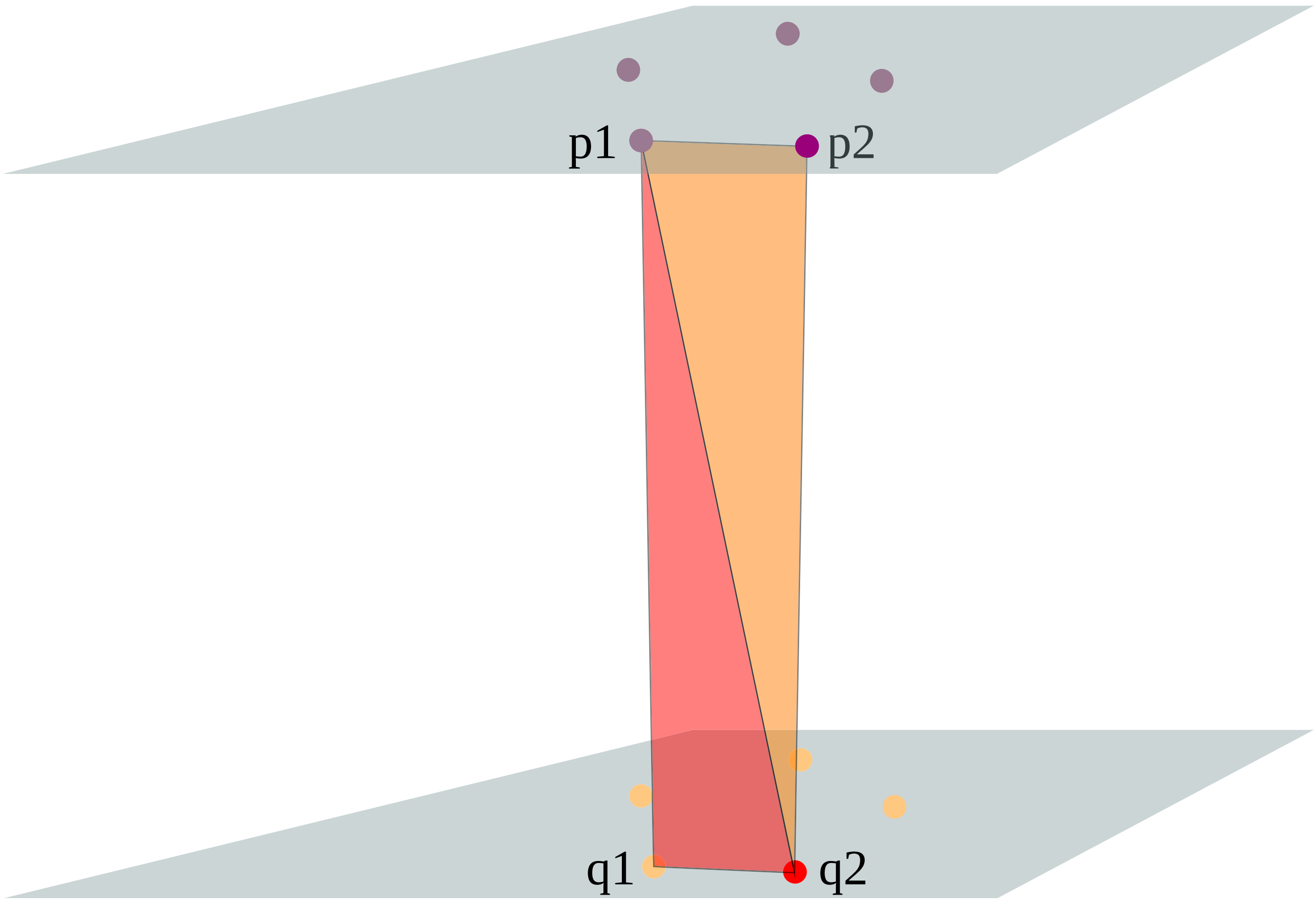
Option 2 - 1



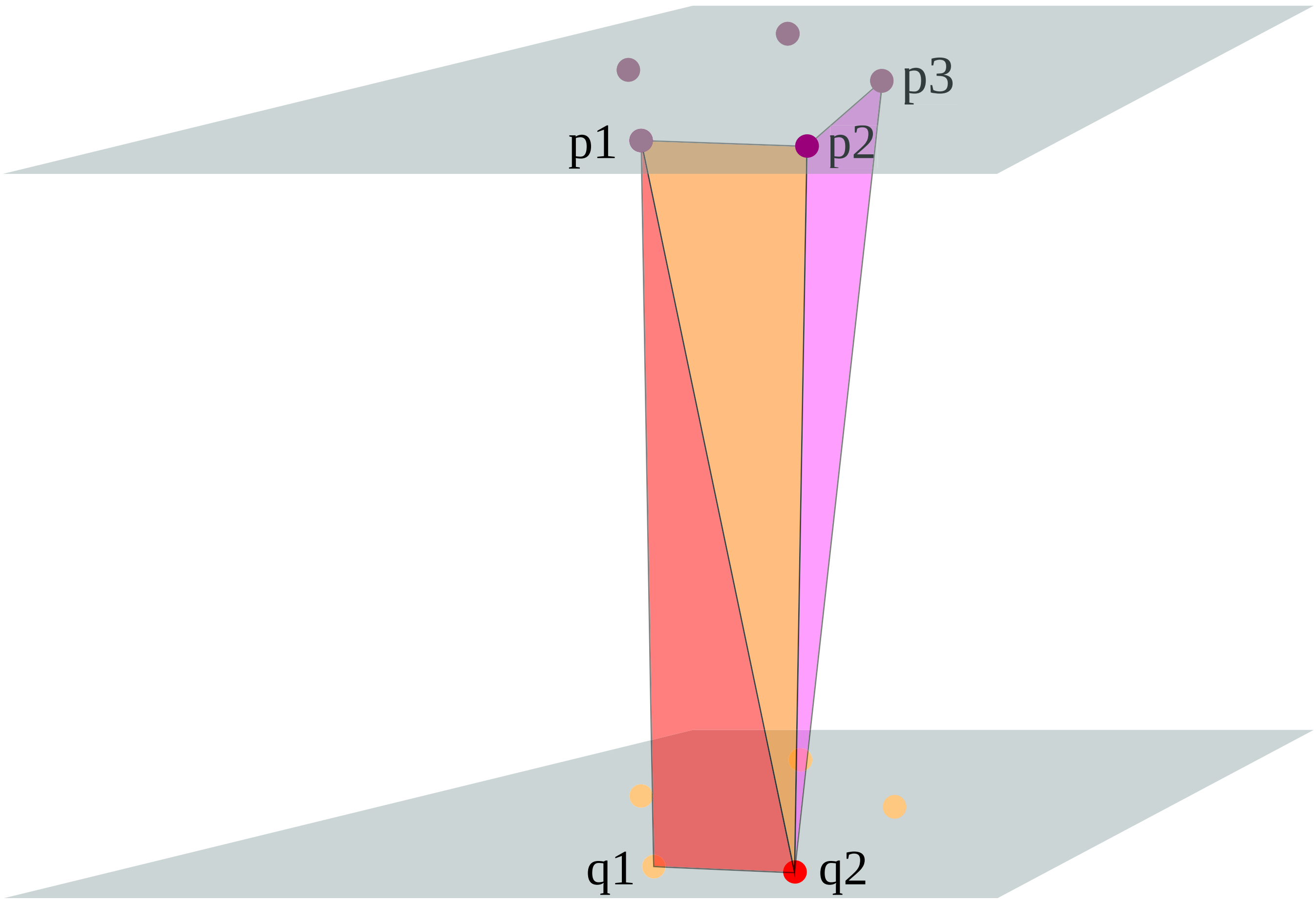
Option 2 - 2



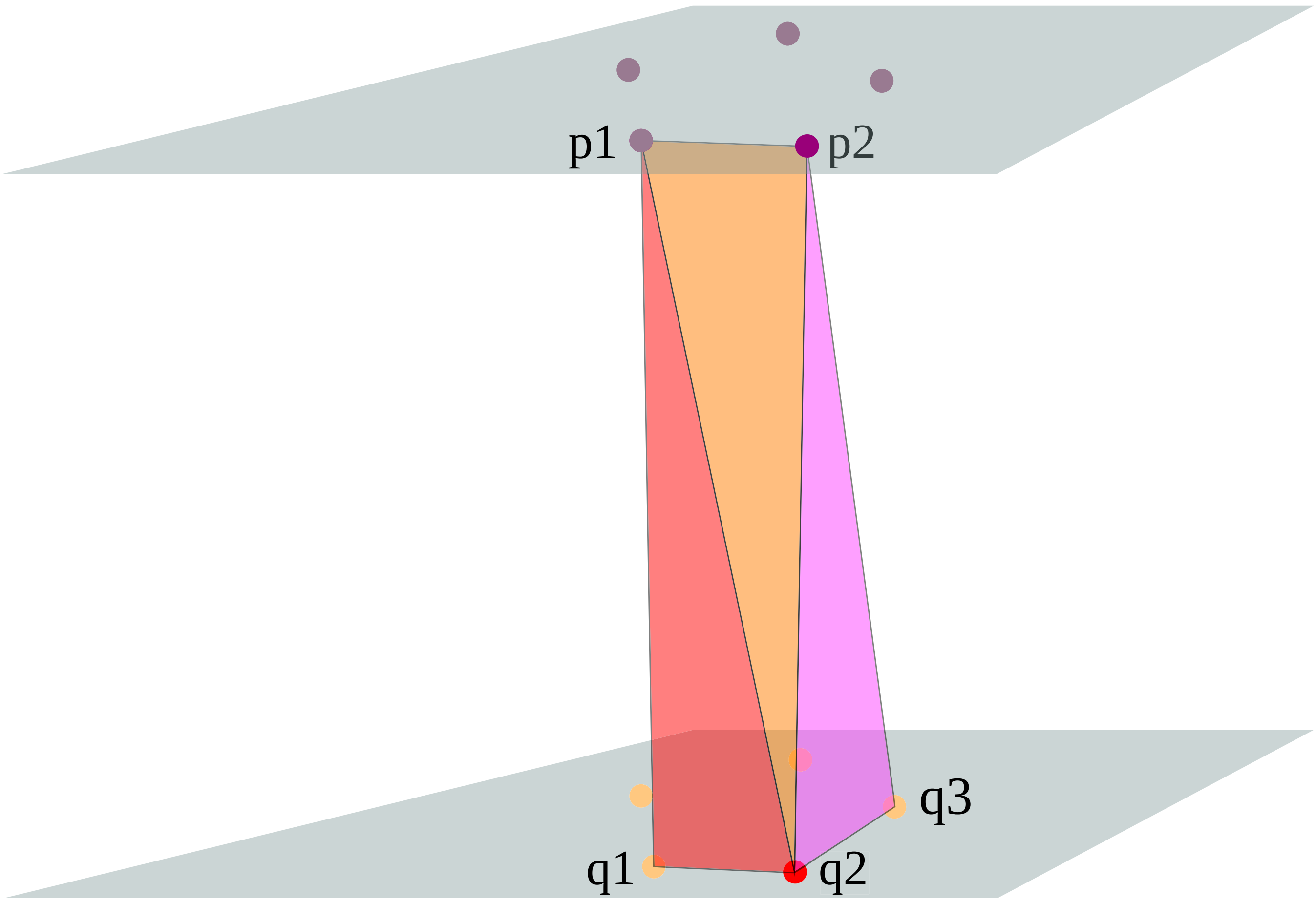
Go with Option 2 – 1



Option 2 – 1 - 1



Option 2 -1 -2



Greedy Solution

Works for regular polygons, but not in general

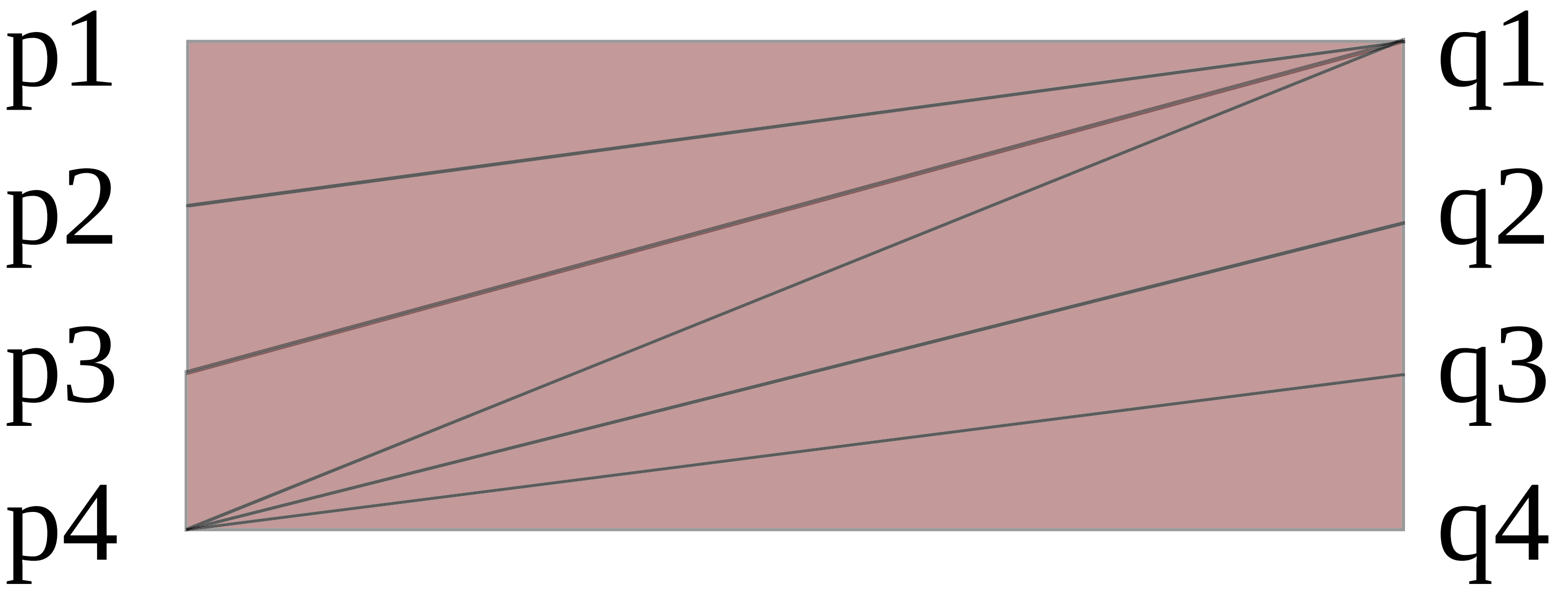
Brute Force Recursion (Only Returns Cost)

```
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)
```

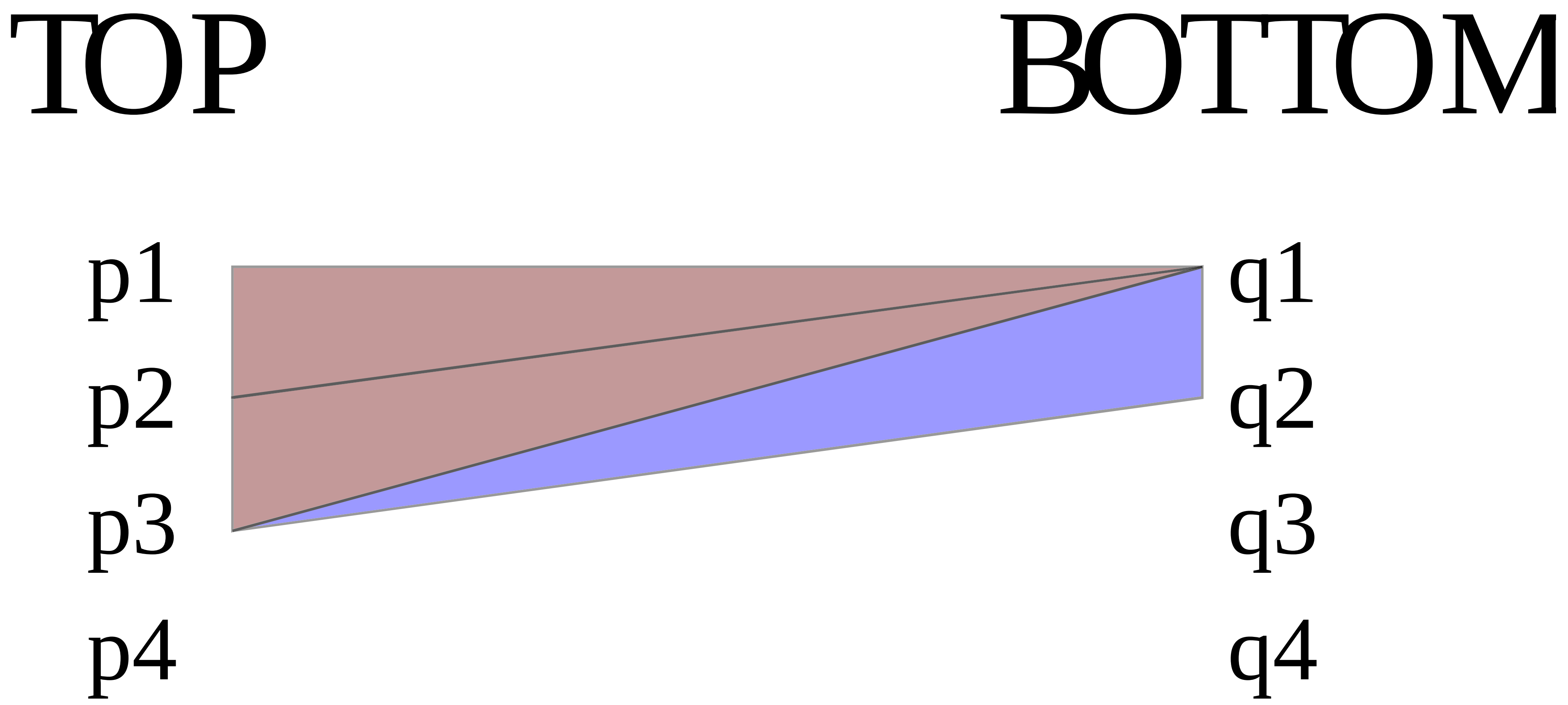
Note:
allPoints in an ordered 2D array where the starting points are repeated at the end of the array

Recursion Branch Leaf Example

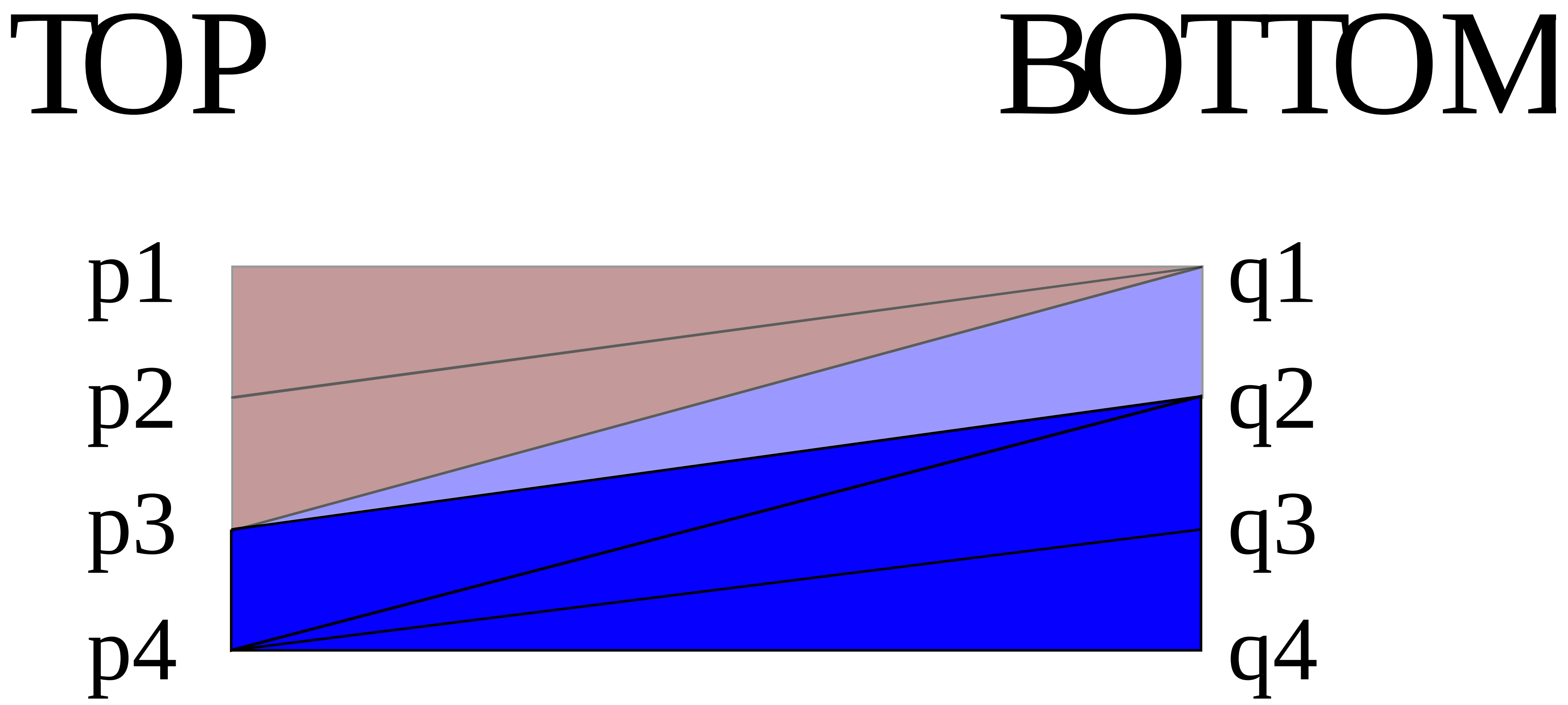
TOP BOTTOM



A Different Branch



The Other Branch's Corresponding Leaf

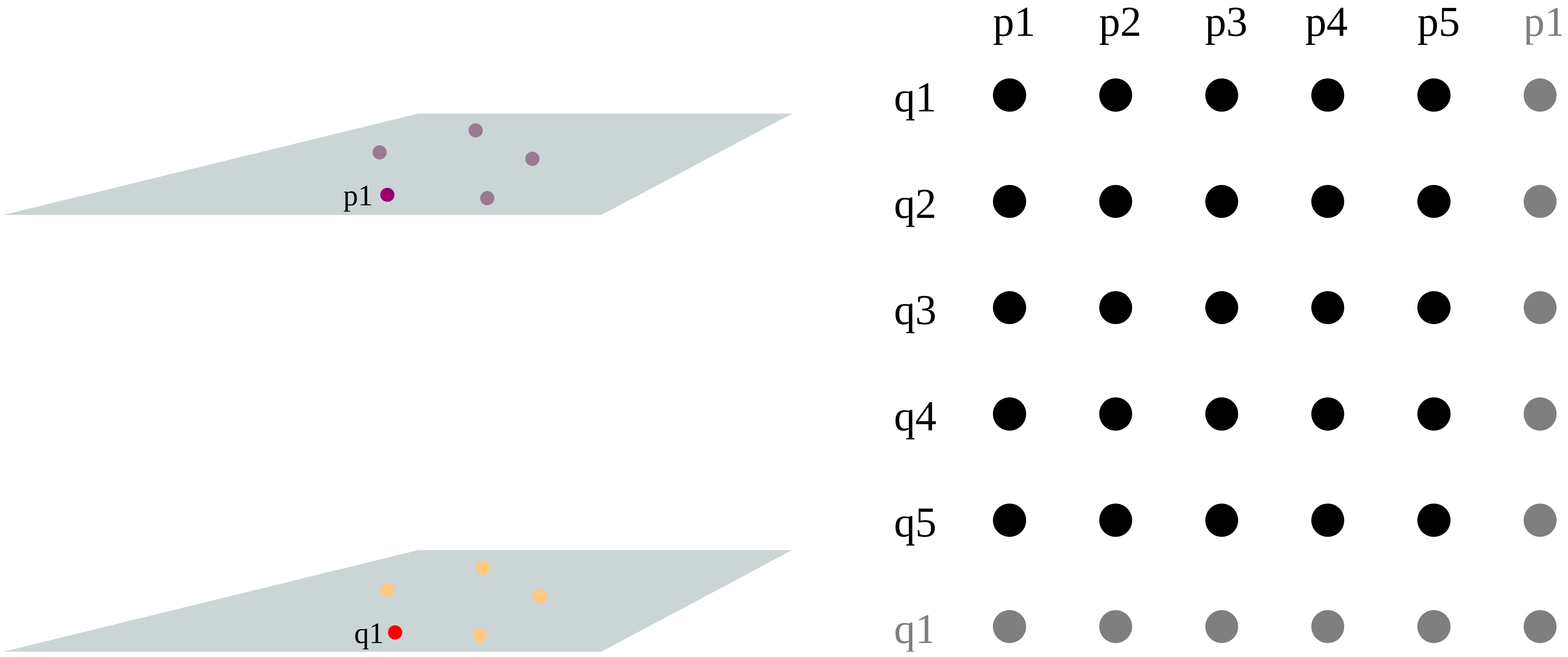


Recursion Issues

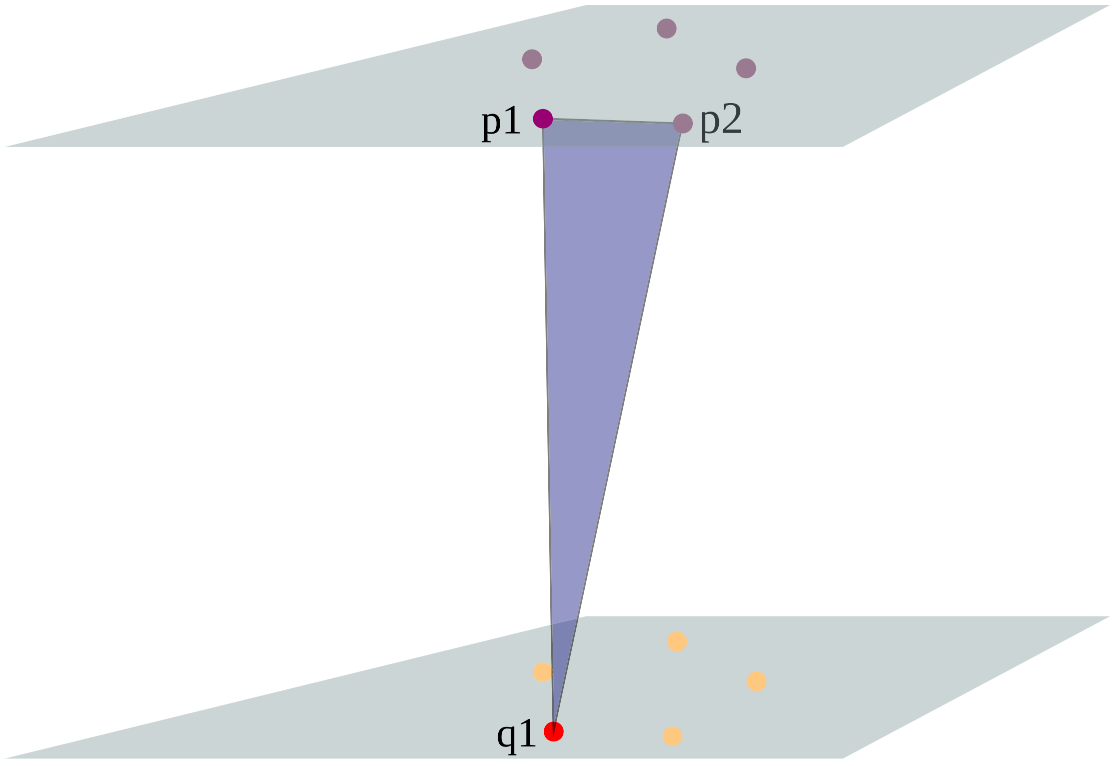
- Recursion is $O(2^n)$ assuming top and bottom points are aligned
- If you do not know which point to start with on top, it's $O(n2^n)$

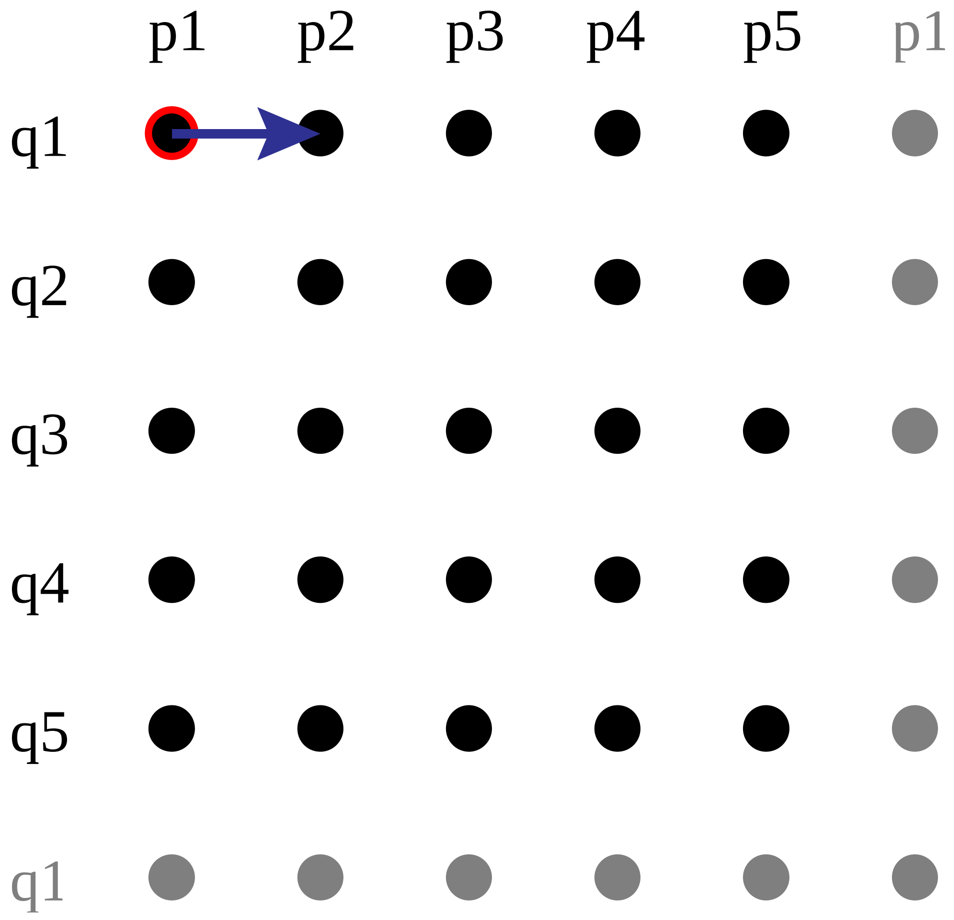
Dynamic Programming

Toroidal Graph Method

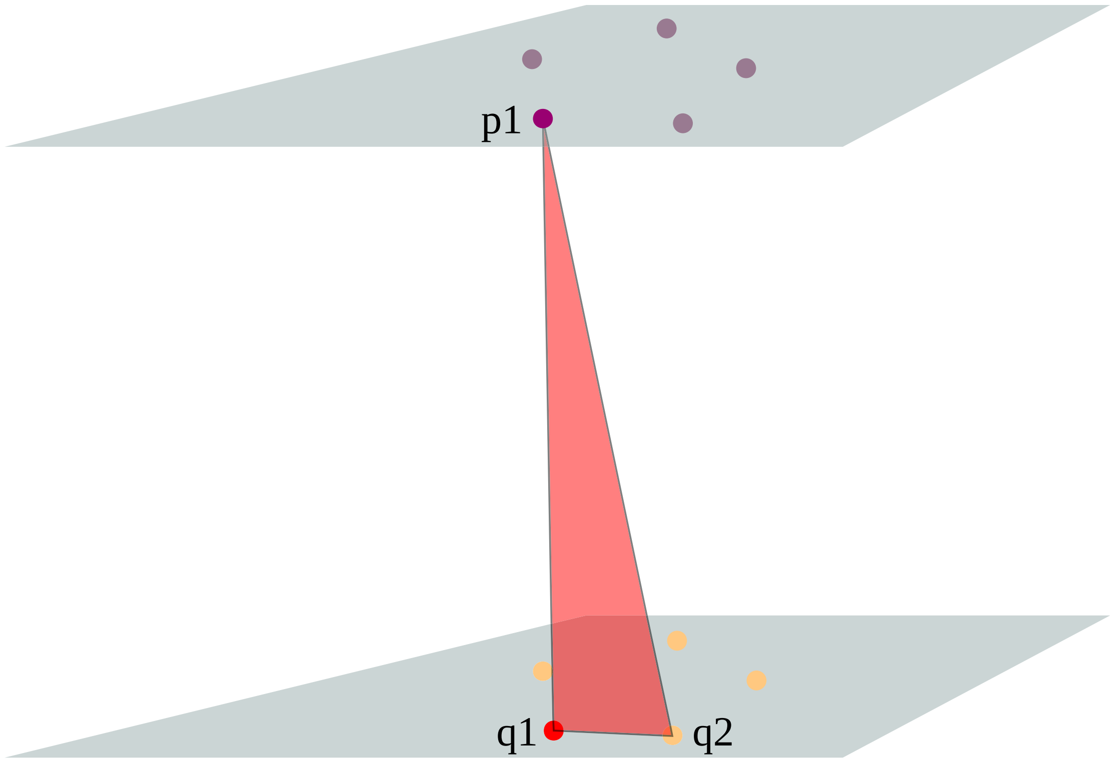


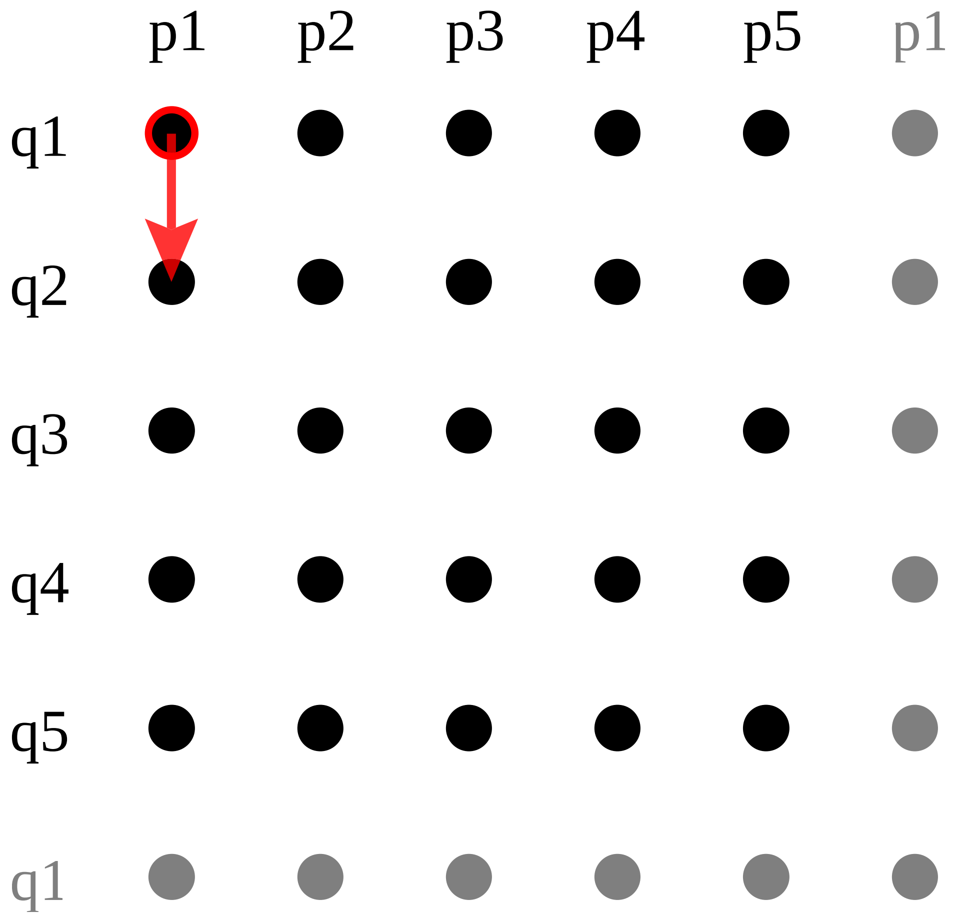
Directed Edges on the Graph



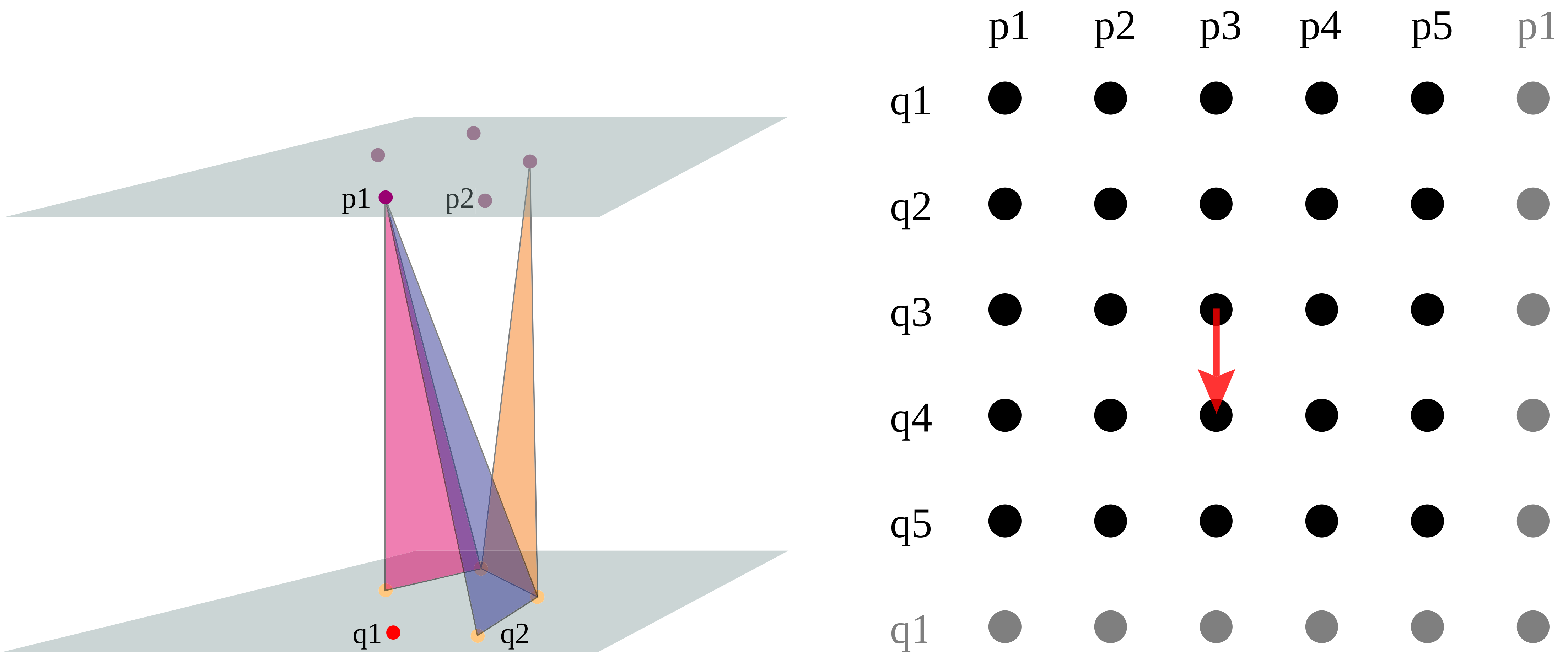


Right or Down

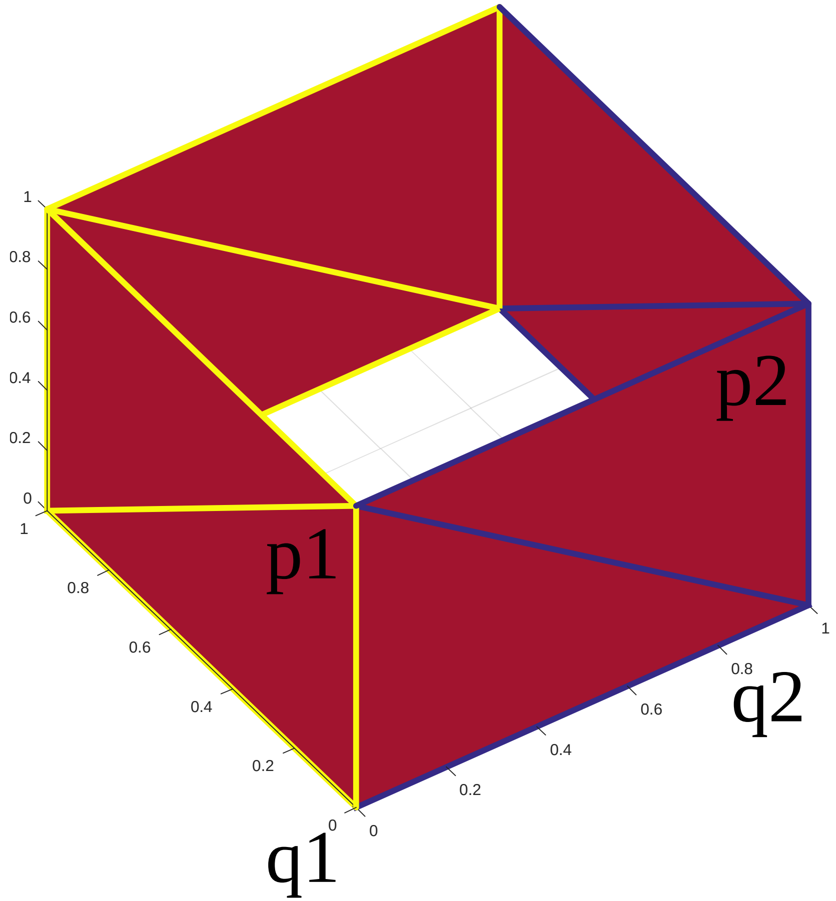


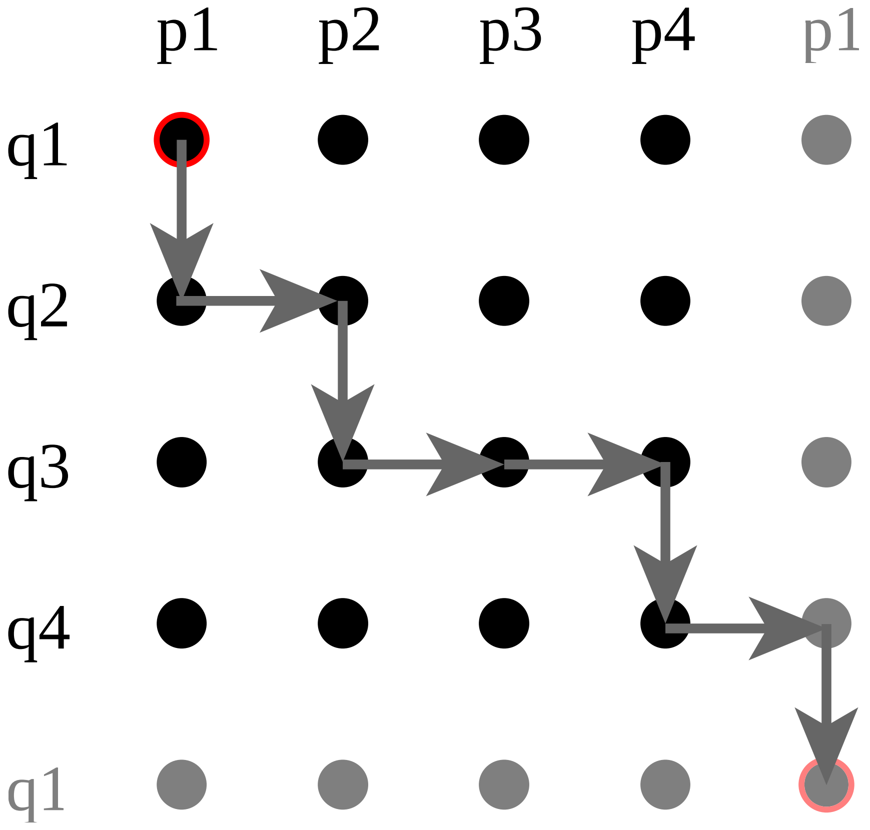


Which Triangle Does this Edge Represent?

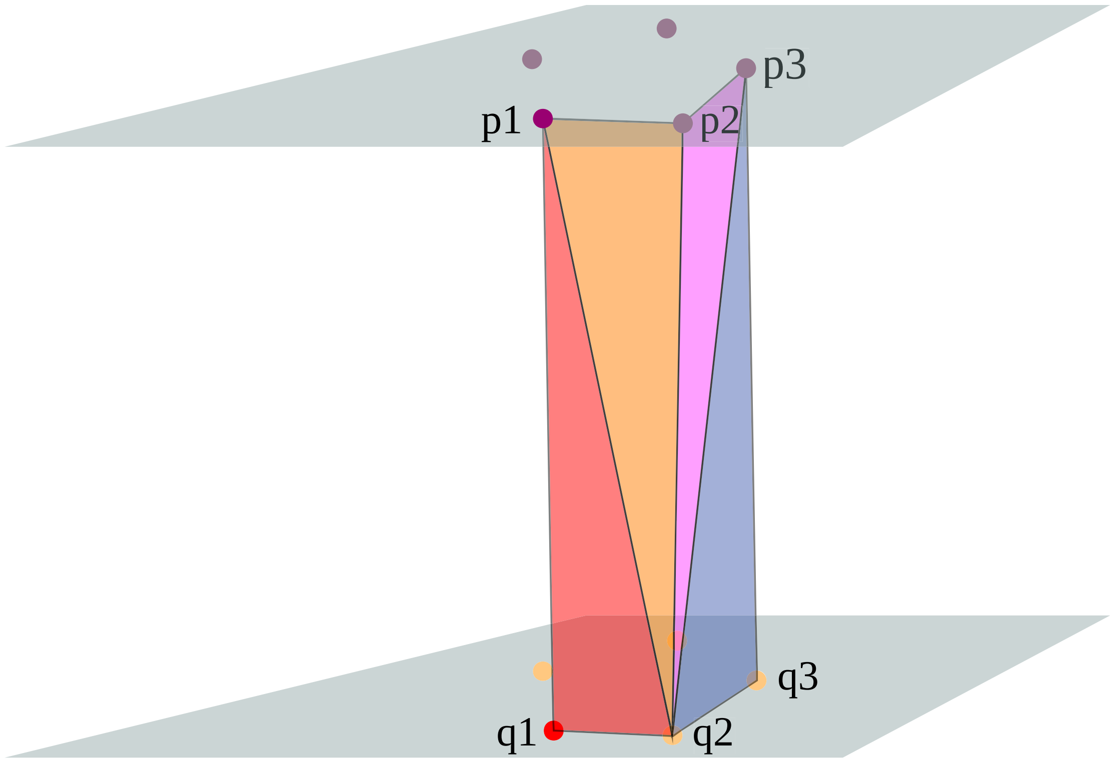


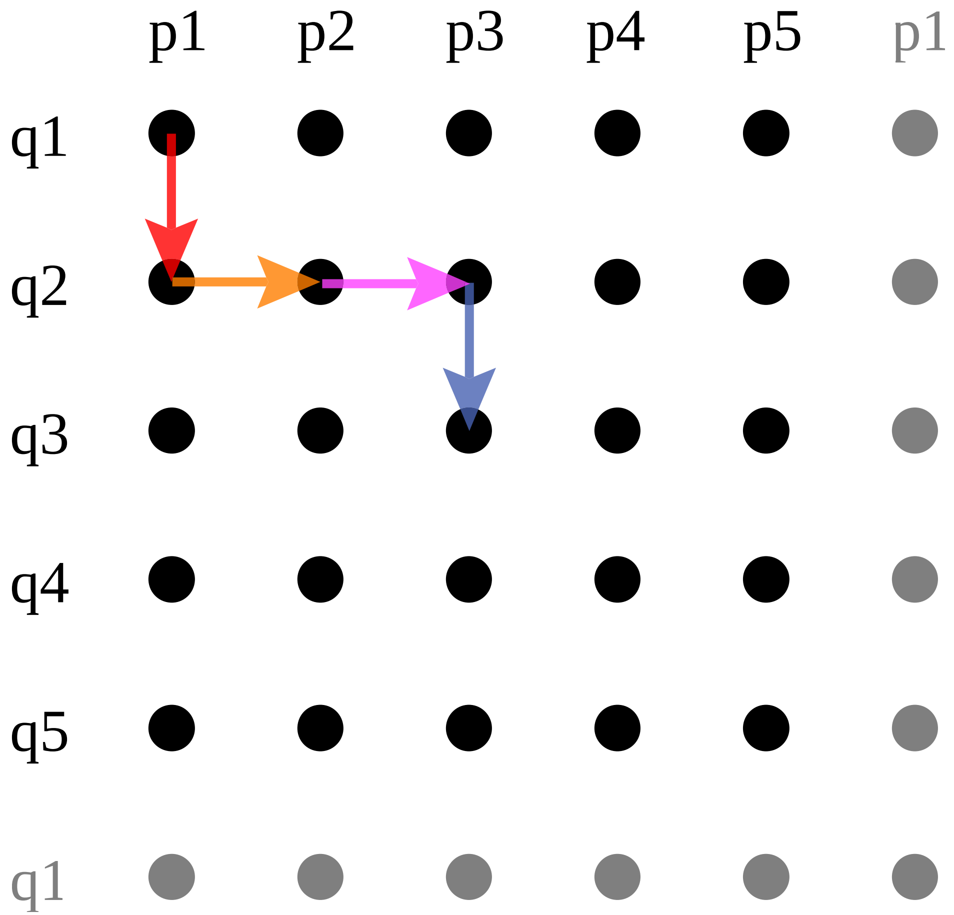
A Full Path Represents a Solution





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

Find the Minimum Cost Path

1	3	6
2	2	3
1	9	4

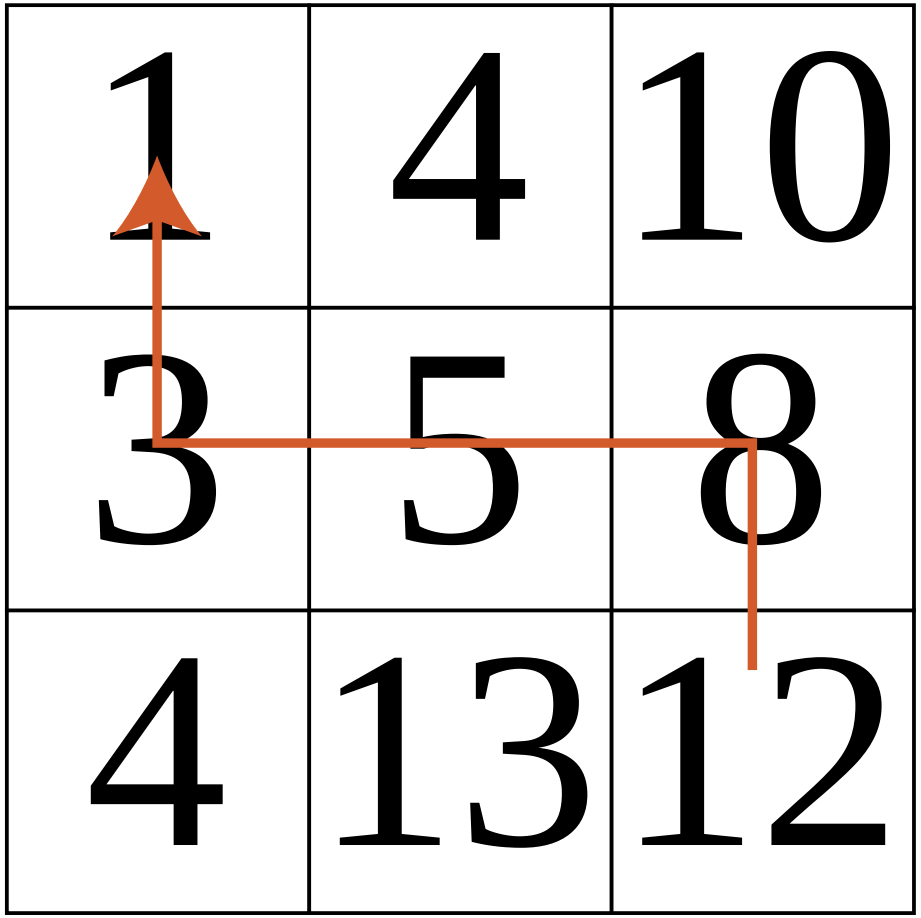
Solution: Build a Path Cost Matrix

1	3	6
2	2	3
1	9	4

1	4	10
3	5	
4		

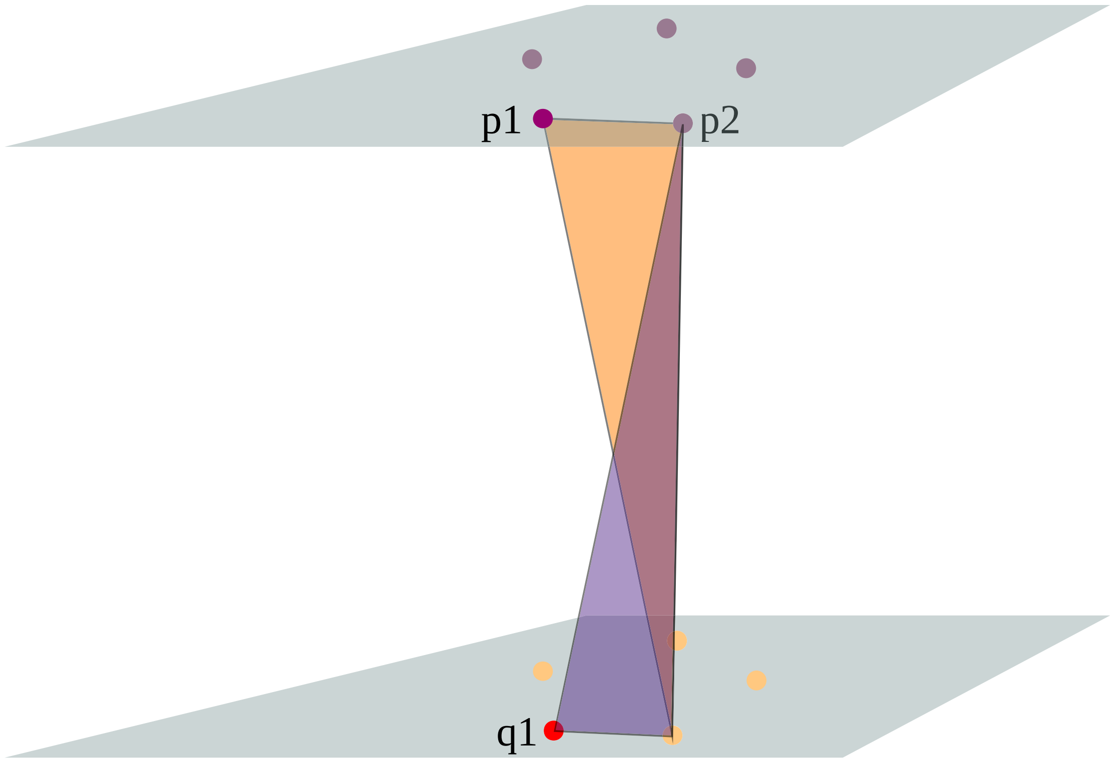
Greedy Algorithm Finds Path

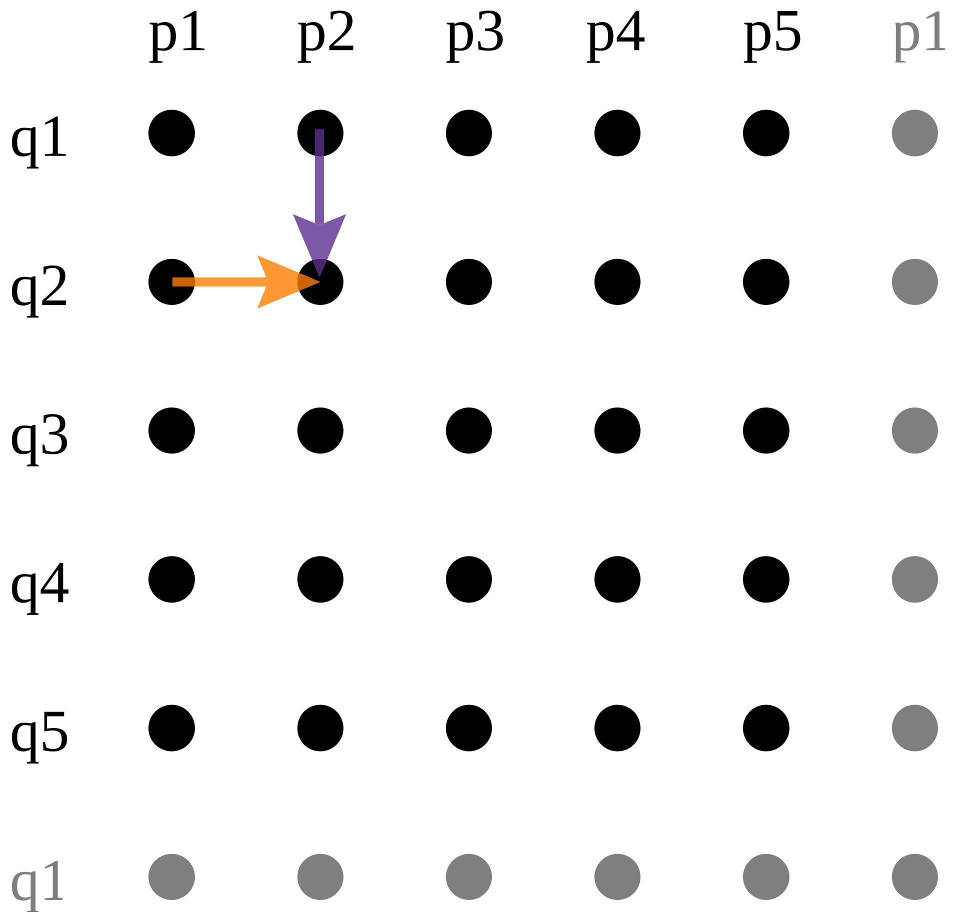
1	3	6
2	2	3
1	9	4



Minimum Cost Path for Toroidal Graph

Previous Algorithm Will not Work





Programming Assignment 3 - Conlcusion

- 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 (p1,q1)