PRESENTATION ON

Optimal Polygon Triangulation By

Md. Saiful Islam

ID: 2022-3-6-045

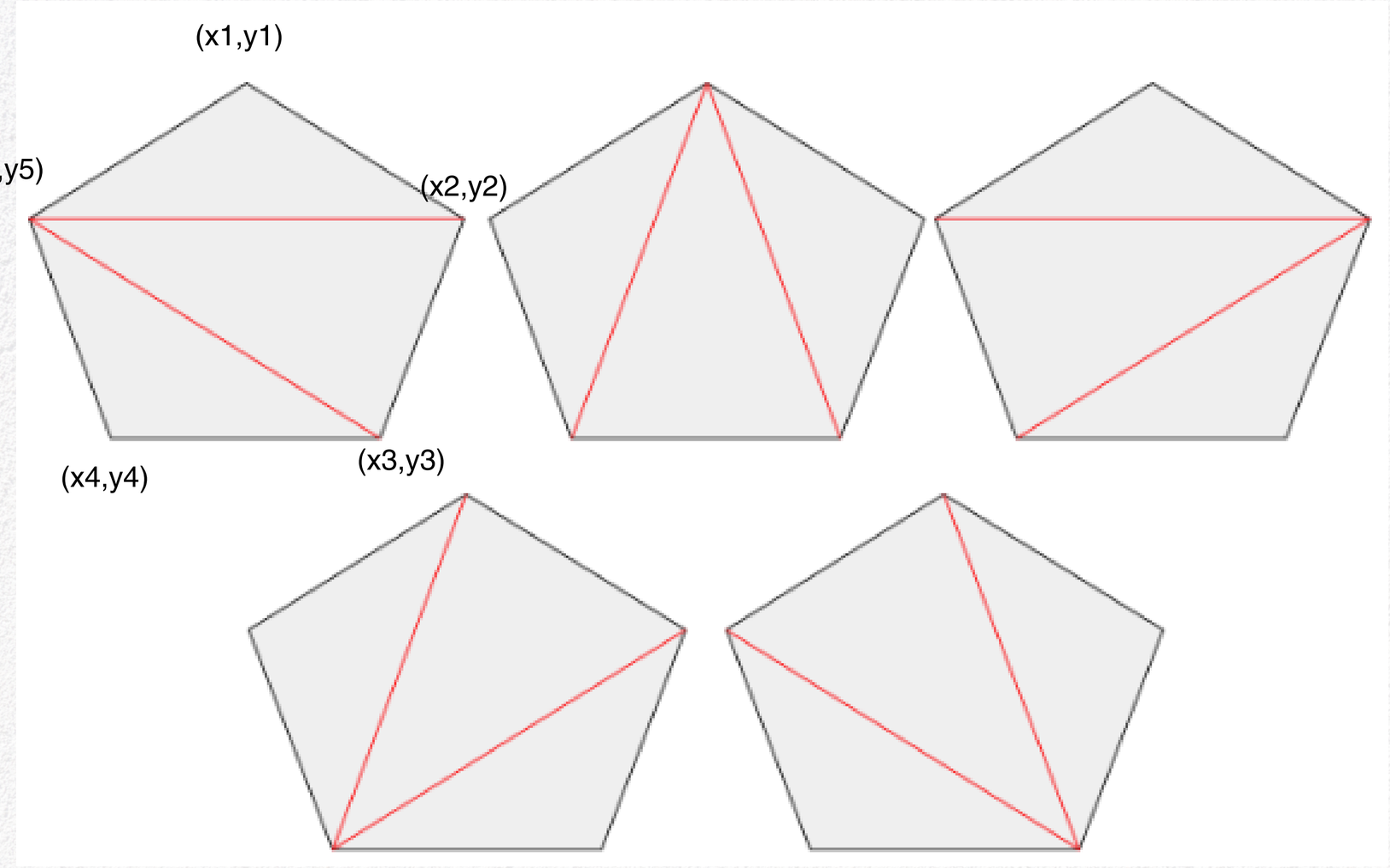
Umme Mukaddisa

ID: 2022-3-60-317

What is Optimal Polygon Triangulation

Optimal Polygon Triangulation refers to the process that divides a simple polygon into non-overlapping triangles such that a certain cost function is minimized (or optimized).

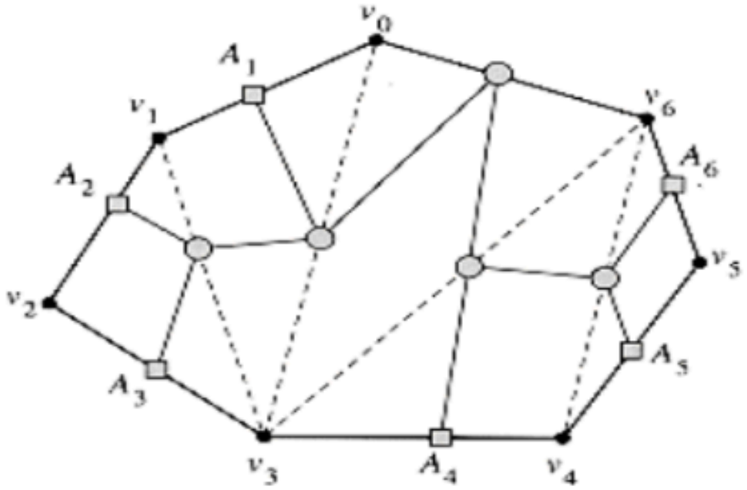
- **Polygon Triangulation**
- **Optimality Criteria**

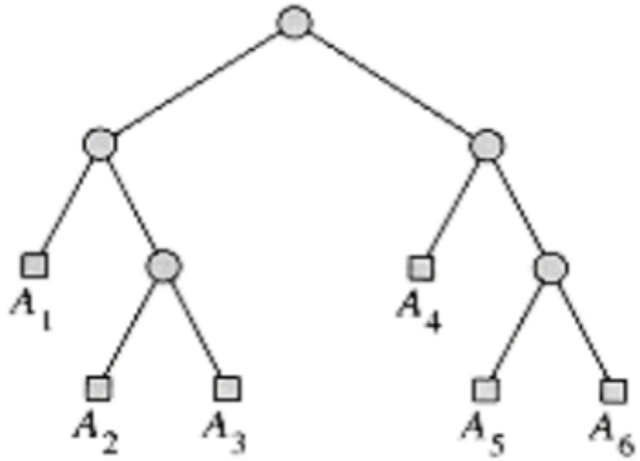


Triangulation Cost:
Sum of the perimeter of all the non-overlapping trinagles.

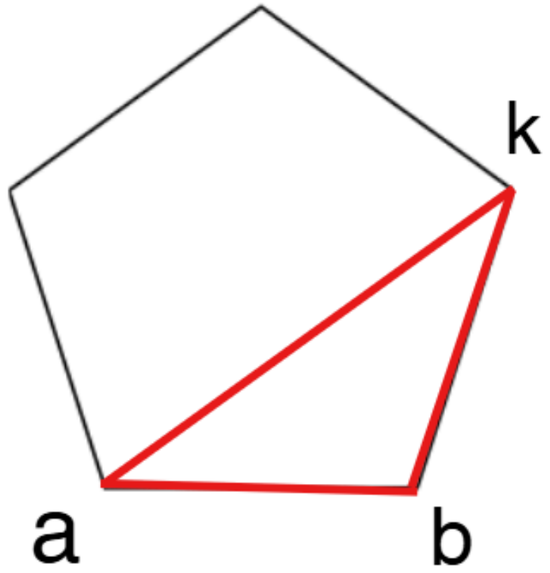
Algorithm to Determine the Optimal Solution

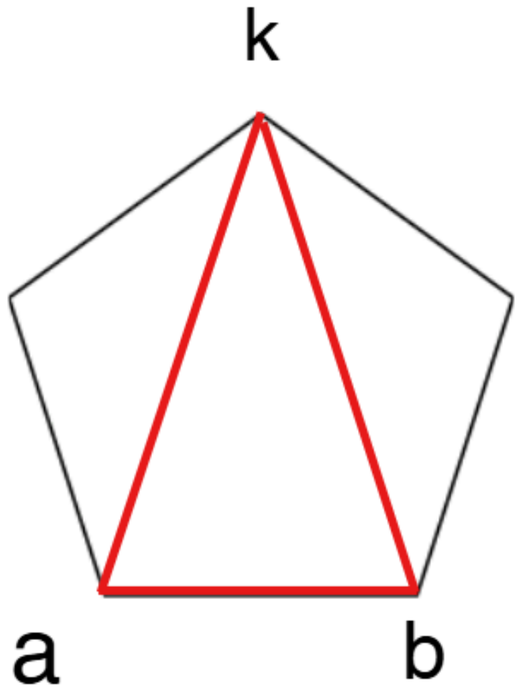
Optimal Polygon Triangulation Using Dynamic Programming:

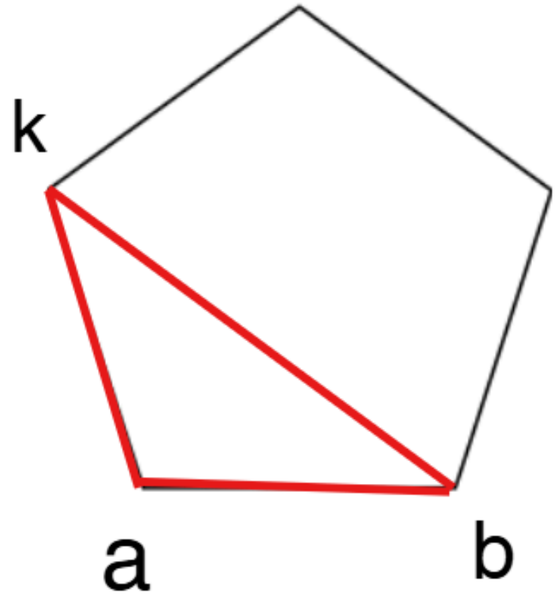




How it Works







Data Type & Data Selection

Input:

A list of n vertices, where each vertex is defined by its x and y coordinates in the format (x, y).

Example: V1(0,0), V2(2,0), V3(3,1), V4(1,3), V5(–1,2)

Output:

The optimal cost of triangulation .

Pseudo Code

```
double computeOptimalTriangulation( )
                        {
    for (int len = 2; len < n; ++len) {
        for (int i = 0; i + len < n; ++i) {
            int j = i + len;
            dp[ i ][ j ] = numeric_limits<double>::infinity( );

      for (int k = i + 1; k < j; ++k) {
          double cost = dp[ i ][ k] + dp[ k ][ j ] +  trianglePerimeter( points[i], points[k], points[j]
                        );
           if (cost < dp [ i ][ j ] ) {
               dp [ i ][ j ] = cost;
               split [ i ][ j ] = k;
                        }
                    }
                        }
                          }
           return dp[ 0 ][n - 1];
                        }
```

Step By Step Calculation

dp Table after len = 2

dp table:

	V1	V2	V3	V4	V5
V1	0	0	0	0	0
V2	0	0	0	0	0
V3	0	0	0	0	0
V4	0	0	0	0	0
V5	0	0	0	0	0

Split table:

	V1	V2	V3	V4	V5
V1	-1	-1	-1	-1	-1
V2	-1	-1	-1	-1	-1
V3	-1	-1	-1	-1	-1
V4	-1	-1	-1	-1	-1
V5	-1	-1	-1	-1	-1

dp Table after len =3

$$dp[i][j] = \min_{k \in [i+1, j-1]} \left(dp[i][k] + dp[k][j] + \text{Cost of Triangle}(V_i, V_k, V_j) \right)$$

dp table:

	V1	V2	V3	V4	V5
V1	0	0	6.58	0	0
V2	0	0	0	7.4	0
V3	0	0	0	0	9.19
V4	0	0	0	0	0
V5	0	0	0	0	0

Split table:

	V1	V2	V3	V4	V5
V1	-1	-1	1	-1	-1
V2	-1	-1	-1	2	-1
V3	-1	-1	-1	-1	3
V4	-1	-1	-1	-1	-1
V5	-1	-1	-1	-1	-1

dp Table after len =4

dp table:

	V1	V2	V3	V4	V5
V1	0	0	6.58	15.73	0
V2	0	0	0	7.4	16.41
V3	0	0	0	0	9.19
V4	0	0	0	0	0
V5	0	0	0	0	0

Split table:

	V1	V2	V3	V4	V5
V1	-1	-1	1	2	-1
V2	-1	-1	-1	2	3
V3	-1	-1	-1	-1	3
V4	-1	-1	-1	-1	-1
V5	-1	-1	-1	-1	-1

dp Table after len =5

dp table:

	V1	V2	V3	V4	V5
V1	0	0	6.58	15.73	23.3639
V2	0	0	0	7.4	16.41
V3	0	0	0	0	9.19
V4	0	0	0	0	0
V5	0	0	0	0	0

Split table:

	V1	V2	V3	V4	V5
V1	-1	-1	1	2	3
V2	-1	-1	-1	2	3
V3	-1	-1	-1	-1	3
V4	-1	-1	-1	-1	-1
V5	-1	-1	-1	-1	-1

Time Complexity

- Outer Loop (len loop)

  Time Complexity O(n)

- Middle Loop (i loop)

  Time Complexity O(n)

- Inner Loop (k loop)

  Time Complexity O(n)

As these three are nested loop so the total time complexity will be: O(n3)

Why We Have Chosen Dynamic Programming

Choosing the best method:

- Brute Force Enumeration
- Greedy Algorithms
- Dynamic Programming

We have chosen Dynamic Programming to solve this problem as it ensures the optimal solution which we can’t get from other methods.

Real Life Example

- Computer Graphics and 3D Rendering
- Robotics and Pathfinding
- Game Development
- Architecture and Urban Planning
- Art and Animation
- Network Optimization

Conclusion

Polygonal triangulation using dynamic programming is an efficient & reliable method to solve the problem by manipulating its optimal substructure & overlapping subproblems. DP ensures optimal solutions, reduces computational redundancy & achieves a time complexity of $O(n^3)$, making it ideal for real-world applications like 3D modeling, robotics & structural analysis.

THANK YOU