Theoretical Part.

1. Subproblems :-

The main aim is to find MIN(n). We can solve this task by diving it into 'i' fragments.

→ The last line segment approximates the points $P_i \ldots P_n$.
→ The rest of line segments constitute the solution for $P_1 \ldots P_{i-1}$

Therefore, the optimal solution for n can be built by breaking it into two fragments of 'i' length.

Assume the line segment L0 is the last segment of the optimal solution for $P_1 \ldots P_j$ and it gives the approx. values $P_i \ldots P_j$ and the rest of the segments of this soln are different from the set that approximates $P_1 \ldots P_{i-1}$. This set is non-empty.

We combine this soln for $P_1 \ldots P_{i-1}$ with L and get a better soln!

This contradicts our assumption for optimal solution.

Hence, set constituting the optimal solution for $P_1 \ldots P_j$ is optimal substructure.

2. Best case: (Base case).

Base case => $MIN(0) = 0$
Approximating 0 points costs 0.

We build other sub-problems based on this base case.

3. Recurrence Relation.

$$MIN(j) = \min_{1 \le i \le j} \{ e(i,j) + 1 + MIN(i-1) \}$$

=> Trying all the possibilities for the least square line segment.

For each i we find the least square approximation of $P_i \dots P_j$

This costs $e(i,j) + 1 + Min(i-1)$

$e(i,j)$ => sum of squared errors.

1 => cost of creating a new line.

$MIN(i-1)$ => cost for approximating the set of remaining points.

4.

4. Pseudocode.

```
MIN[0] = 0;
for j : 1 to n
    MIN [j] = +infinity;
    for i : 1 to j = 1
        MIN [j] = min [a, b)
return MIN[n];

a -> MIN[j]
b -> e(i, j) + 1 + MIN [i-1] );
```

5. Time complexity.

$n \rightarrow$ Total number of points.

each problem is calculated in $O(n)$

we have n- sub problems.

therefore, $O(n^2)$.