Solution for Home Assignment 2 (Theoretical part)
Danis Alukaev BS19-02

3. Connected lines least squares.

Given n points on a 2D plane $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$ with $x_1 < x_2 < ... < x_n$ that lie roughly on a sequence of several line segments. The idea is to approximate this data set not with a single line, but with several connected lines. So, our goal is to find the minimum cost (defined below) for given n points in the plane using dynamic programming.

Let MIN(j)= minimum cost for points $(x_1, y_1), (x_2, y_2), ..., (x_j, y_j)$,
$e(i, j)$ = minimum sum of squared errors for points $(x_i, y_i), (x_{i+1}, y_{i+1}), ..., (x_j, y_j)$,
Sum of squared errors $SSE = \sum_{i=1}^{n}(y_i - (ax_i + b))^2$, that is minimized when

$a = \frac{n\sum_i(x_iy_i) - (\sum_i x_i)(\sum_i y_i)}{n\sum_i x_i^2 - (\sum_i x_i)^2}$ and $b = \frac{\sum_i y_i - a\sum_i x_i}{n}$

Now, we can define trade-off (cost) function for a particular solution of connected lines least squares problem: Trade-off = (the sum of SSE in each segment) + (the number of lines in our approximation).

1.Define what are the subproblems.
The minimal cost of the particular point (x_j, y_j) can be either $e(i, j)$ for the solution that keep the line started in (x_i, y_i) to approximate point (x_j, y_j) where $1 \leq i < j \leq n$, or $e(i, j-1)+1$ for solution that requires introduction of a new line. We notice that for some problem MIN(j) the segment (with the right endpoint (x_j, y_j) and the optimal partition) starts at the some earlier point (x_i, y_i). Therefore, we can solve the problem MIN(j) if we know the solution for the subproblem MIN($i-1$) for $1 \leq i \leq j \leq n$.

2.Recognize and solve the base cases.
Since the calculation of MIN(j) requires the value of MIN($i-1$) and $i \geq 1$ (number of points), our base case will be MIN(0). Let's consider such a case: if we have no points, then we do not introduce any lines, and minimal cost of the solution will be equal to 0. Therefore, MIN(0) = 0.

3.Write down the recurrence that relates subproblems.
There are two possible scenarios for a particular point (x_j, y_j), where $1 \leq j \leq n$:

First scenario is to keep the line started in some point (x_i, y_i) where $1 \leq i < j$, and use this line to approximate current point (x_j, y_j). The cost of such a solution will be the $e(i, j)$.
Second scenario is to introduce new line started in (x_j, y_j) and the cost of such a solution will be the $e(i, j - 1) + 1$.
So, for a particular MIN(j) we try to find the balance of MIN($i - 1$) and the $e(i, j)$, where $1 \leq i \leq j \leq n$. It means, that we consider the partition defined in the previous iteration and decide is it optimal to introduce new line, or just keep the line continuous from some point (x_i, y_i) to approximate the current point (x_j, y_j).
Hence, the recurrence relation for the connected lines least squares problem can be defined as:
MIN(j) = min($e(i, j)$+ MIN($i - 1$) +1), for $1 \leq i \leq j$ and $j \neq 0$
MIN(j) = 0, for $j = 0$.

4.Write the pseudocode (always using DP) that returns MIN(n).
The pseudocode shown in the table Algorithm 1 (page 3).
Firstly, we declare two arrays to store computations of the minimum cost and minimum SSE for corresponding points. The bottom-up approach allows us to avoid the overlapping of subproblems, i.e. it optimizes the process of $e(i, j)$ and MIN(i) computation. Also, we define base case for connected lines least squares problem that is MIN(0) = 0. Then, we calculate and store minimum SSE for all combinations of points $(x_i, y_i), (x_{i+1}, y_{i+1}), ..., (x_j, y_j)$ with $1 \leq i \leq j \leq n$. Next step is the computing and storing the minimum cost for all sequences of points $(x_1, y_1), ..., (x_t, y_t)$ with $t \in [1..n]$. To do that we use the derived in previous task recurrent formula MIN(j) = min($e(i, j)$+ MIN($i - 1$) +1), for $1 \leq i \leq j \leq n$ and $j \neq 0$. Finally, the result will be stored in MIN[n], and we can return this value.

5.State and justify the complexity of your pseudocode.
To treat all pairs (i, j) with $1 \leq i \leq j \leq n$ we need the quadratic time. Every computation of $e(i, j)$ takes linear time, because we process all points $(x_i, y_i), ..., (x_j, y_j)$ to compute their squared errors. Therefore, the computation of all $e(i, j)$ for $1 \leq i \leq j \leq n$ takes $O(n^3)$ time.
Then, we process given n points, and on each iteration j we try to find such an i that $1 \leq i \leq j$ holds and [$e(i, j)$+MIN($i - 1$)+1] minimized. It can be performed in $O(n^2)$ time. Hence, the time complexity of proposed algorithm is $O(n^3) + O(n^2) = O(n^3 + n^2) = O(n^3)$.

Algorithm 1 Connected lines least squares.

```
// declare arrays
new array Min[n + 1] // to store computations of the minimum cost for corresponding points
new array E[n + 1][n + 1] // to store computations of the minimum SSE for corresponding points

// set the base case
Min[0] ← 0

for j = 1 to n do
    for i = 1 to j do
        // compute and store the minimum SSE for points
        //(x_i, y_i), (x_{i+1}, y_{i+1}), ..., (x_j, y_j)
        E[i][j] ← e(i, j)

for j = 1 to n do
    // introduce new variable that will store the MIN(j)
    min ← MAX_VALUE
    for i = 1 to j do
        // compute the cost for corresponding points
        temp ← Min[i − 1] + E[i][j] + 1
        // check whether the obtained value is the minimum
        // among all previously calculated values
        if temp < min then
            // set the preliminary minimum cost
            min ← temp
    // store the result of MIN(j)
    Min[j] ← min
return Min[n]
```