Tutorial 2

Kousshik Raj (17CS30022)

04-08-2019

1 Problem Statement

$P[1..n]$ is an input list of n points on the xy-plane. Assume that all n points have distinct x-coordinates and distinct y-coordinates. Let p_L and p_R denote the leftmost and the rightmost points of P, respectively. The task is to find the polygon Q with P as its vertex such that the following conditions are satisfied.

i) The upper vertex chain of Q is x-monotone (increasing) from p_L to p_R.

ii) The lower vertex chain of Q is x-monotone (decreasing) from p_R to p_L.

iii) Perimeter of Q is minimum.

2 Recurrence

We are given the points $P[1..n]$ and we need to find a polygon Q with vertexes from them such that the distance is minimised when moving from the leftmost point to the rightmost point through the upper chain in a strictly increasing x-monotone and back to the leftmost point through the lower chain in a strictly decreasing x-monotone.

Now let's consider the points $R[1..n]$ which consists of all points from P sorted according to their x - coordinates. Let $dis(i,j)$ be the distance between the points $R[i]$ and $R[j]$.

Now, let the function $dp(i,j)$ represent the minimal perimeter for the polygon when the points $R[i]$ and $R[j]$ lie on separate chains. Then the required minimal perimeter is $dp(n,n)$. Since $dp(i,j) = dp(j,i)$ we need to consider the cases only for $i \leq j$. The following recurrence can be used to calculate the function dp.

$$dp(i,j) = \begin{cases} 0, & \text{if } i = j = 1 \\ dp(j,i), & \text{if } j < i \\ dp(i,j-1) + dis(j-1,j), & \text{if } i < j-1 \\ \min\limits_{1 \le k < j} dp(i,k) + dis(k,j), & \text{otherwise} \end{cases} \tag{1}$$

The first and second cases are trivial and the other cases are explained in the next section.

3 Algorithm

Let us consider the third case, i.e, $i < j - 1$. In this case, the path ending in $R[j]$ must also contain the point $R[j-1]$ as the other path cannot visit this point and backtrack to $R[i]$. Hence, $dp(i,j)$ will be equal to the sum of minimal cost of $dp(i,j-1)$ and the distance between $R[j-1]$ and $R[j]$.

In the final case, the optimal path must end in node $R[j]$ and come from some node $R[k]$. So the value of $dp(i,j)$ is minimized by iterating over all possible k, where $i \le k < j$. Hence, $dp(i,j) = \min\limits_{1 \le k < j}$ dp(i,k) + dis(k,j).

Finally, the optimal polygon with minimal perimeter can be derived when calculating the function dp, by storing the previous nodes for the points in the upper chain. The rest of the points must be in the lower chain.

From the pseudo-code,retrace the nodes from $Poly[n][n]$ to get the upper chain of the polygon and the remaining nodes are the lower chain of the polygon.

Pseudo-Code

4 Time and space complexities

Lines	Time Complexity
5	$O(n^2)$
6..7	$O(1)$
8..19	$O(n^3)$
20	$O(1)$
	Total: $O(n^3)$

Lines	Space Complexity
4	$O(n)$
5	$O(n^2)$
	Total: $O(n^2)$

Algorithm 1 Minimal Monotonic Polygon

Input
$P[1....n]$, an array of n points

func minDistance(P, n)
$R \leftarrow Sort_x(P)$ ▷ Sort according to their abscissa
Create a 2D array $dp[1..n][1..n]$, $Poly[1..n][1..n]$

$dp[1][1] \leftarrow 0$
$Poly[1][1] \leftarrow 1$

for $j \leftarrow 1$ to n **do**
    for $i \leftarrow 1$ to j **do**
        if $i < j - 1$ **then**
            $dp[i][j] \leftarrow dp[i][j-1] + dis(j-1, j)$
            $Poly[i][j] \leftarrow j - 1$
        else
            $minK \leftarrow$ INT_MAX
            for $k \leftarrow i$ to $j - 1$ **do**
                if $dp[i][k] + dis(k, j) < minK$ **then**
                    $dp[i][j] \leftarrow dp[i][k] + dis(k, j)$
                    $Poly[i][j] \leftarrow k$
            $dp[i][j] \leftarrow minK$
$MinimumPerimeter = dp[n][n]$ ▷ The minimum perimeter of the polygon