Tutorial 2

Prakhar Bindal (17CS10036)

August 4, 2019

1 Problem Statement

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

i) The upper vertex chain of Q is x-monotone (increasing) from pL to pR.
ii) The lower vertex chain of Q is x-monotone (decreasing) from pR to pL.
iii) Perimeter of Q is minimum.

2 Recurrences

The first step is to sort the points acccording to their x coordinate and let $d(i, j)$ be the distance between points i and j.Now, Consider two paths upper and lower (considering a imaginary line joining leftmost and rightmost points).
For i,j in (1..n) and i,j on different paths. Let dp[i][j] be the minimum total cost of two paths (i.e the length of path from 1 to i and the other part from 1 to j). So now dp[n][n] will be the optimal solution. Clearly dp[i][j]= dp[j][i] , since both represent the same polygon , So we are only interested in i,j with 1 ≤ i ≤ j ≤ n.
Since dp[1][1] represent single point only , therfore dp[1][1]=0. Now , there are two cases possible:

Case 1: when i=j-1 or i=j, So for the optimal solution the path which comes from a node k , where 1 ≤ k ≤ j-1, must ends at j. To make the graph, we will need to store that node k. So, let node[i][j] will store that node for vertex pair i,j. The optimal solution for this case will look like:

$$\text{dp[i][j]} = \min_{1 \leq k \leq j-1} (\text{ dp[i][k]} + \text{d(k,j)})$$

Case 2: when i<j-1, in this case the path that is ending at j must also visit j-1 because it is not possible for i to go to j-1 and then again comes to i. Therefore, the optimal solution for this case will look like:

dp[i][j] = dp[i][j-1] + d(j-1,j)   for i < j-1

3 Algorithm

Pseudo code:

```
dp[1][1]=0
For i in 1 to n
    For j in i+1 to n
        if(i=1 and j=2):
            dp[i][j]=d(1,2)
        else if(i<j-1):
            dp[i][j]= dp[i][j-1] + d(j-1,j)
        else:
            min=INT_MAX
            For k in 1 to j-1
                q=dp[i][k] + d(k,j)
                if(q<min):
                    min=q
                    node[i][j]=k
            dp[i][j]=min
i=n, j=n
graph=g
while(i>1 or j>1)
    if(j<i)
        swap(i,j)
else if(i=j-1 or i=j)
    index=node[i][j]
    addEdge(g,index,j)
    j=index
else
    addEdge(g,j-1,j)
    j - -
```

4 Time and space complexities

Time Complexity = O(n^2)
Space Complexity = O(n^2)