Tutorial 2

Akash Tiwari (17CS10003)

24-07-2019

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.
You have to answer the following. Provide necessary figures/diagrams for explanations.
1. Develop the recurrences needed for DP, with clear arguments.
2. Design the algorithm and write its main steps.
3. Derive the time and space complexities of your algorithm.

2 Assumptions/Intro

Let the **shortest Path Distance** be denoted as **P(i,j)** where i¡=j and which includes all the points p1,....pj(i.e it starts at pi, goes strictly left to p1 and then strictly right to pj) and let d(p1,p2) represent the euclidean distance between the 2 points. .Let us consider a matrix B of dimensions $n * n$. in which B[i,j] represents P(i,j , thus our required shortest distance is B[n]n).Let r[n][n] be another matrix in which r[i][j] represents the immediate predecessor of pj on the shortest path P(i,j).

If the points in P[1...n] are not sorted according to x-coordinate , we sort them first wrt x-co-ordinate.

3 Recurrences

The value of $B[i][j]$ can be **founded by recurrence**, here i¡j :
Case 1: i=1 and j=2
$B[1,2] = dist(p1,p2)$
Case 2; i¡j-1
$B[i,j] = B[i,j-1] + d(p_{j-1},pj)$
Case 3:i=j-1
$B[j-1,j] = min_{1<=k<j-1} B[k,j-1] + d(pk,pj)$
Note/Observations:

- Case 1 is a base case.
- Case 2 occurs when we take the shortest path P(i,j-1) and then add the distance d[j-1,j] to get P(i,j).
- Case 3: we assume k to be the predecessor of pj and add d(pk,pj) and add to P(l,j-1), we take the min across all k and get B[j-1,j]. Also we store the predecessor index in r[j-1,j].

4 Algorithm

Explanation:

- Then using the recurrence relation we fill the matrix B and the matrix r.
- The B[n][n]th element will be the length of the shortest path that can be obtained.
- , reconstruct the polygon, we start from r[n,n] and get the predecessor of pn and backtrack. The next predecessor point will be r[n, index of predecessor of r[n][n]] and so on.
- We construct the polygon using backtracking on matrix r.

5 Demonstration

In this example, the shortest possible such path is p1-p2-p3-p5-p4-p1 with a total path length of 31.01, on applying the recursion steps we get the same least distance of such a path, and the same polygon on backtracking.

6 Time and space complexities

Time complexity (capital N used for better readability)
Time complexity for sorting=O(Nlog(n))
Time complexity for filling the matrix = O(N*N)(Since each element is evalu-

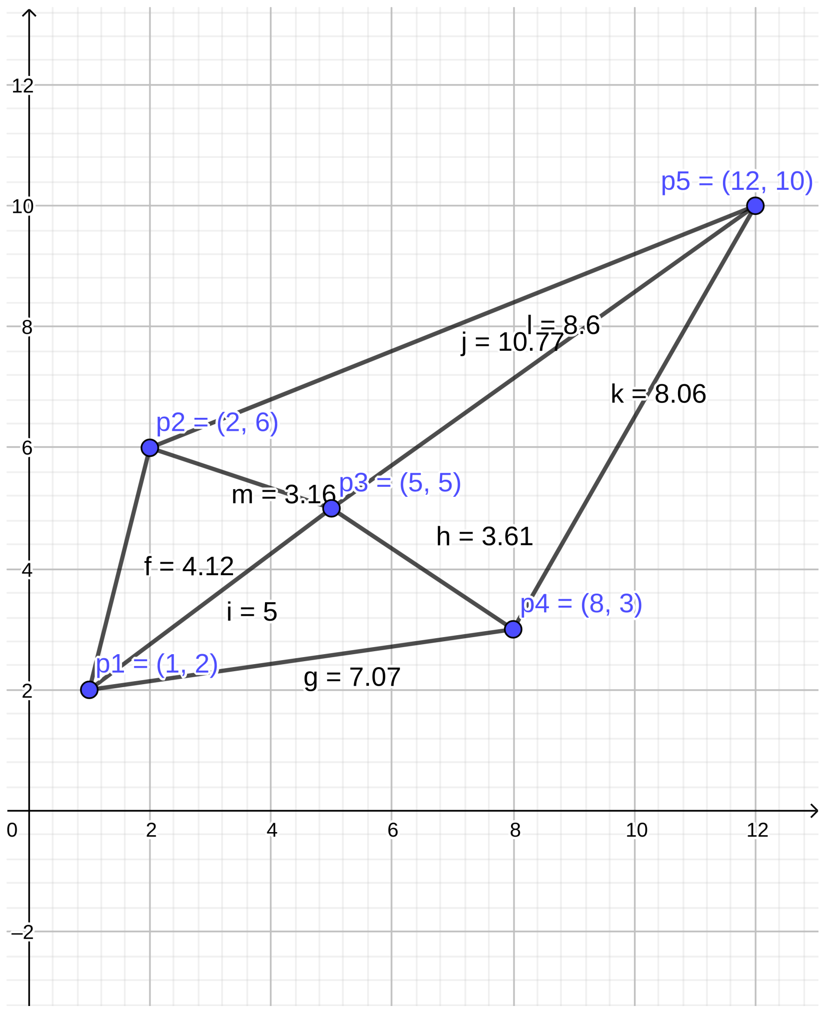

Figure 1:

ated in O(1) time)
Therefore **overall Time complexity**(adding since loops are independently running) = **O(N*N)**
Space complexity(capital N used for better readability)
Space complexity for matrix= O(N*N)
Overall space complexity = O(N*N)