Tutorial 2

Divyang Mittal (17CS10012)

4-08-2019

1 Problem Statement

$P[1..n]$ is a list of n points in xy - plane. Assume that no two points have the same x coordinate or y coordinate. Draw a Polygon Q whose vertex set is P such that,
1. The upper vertex chain of Q is x-monotone (increasing) from leftmost vertex to rightmost vertex.
2. The lower vertex chain of Q is x-monotone (deccreasing) from rightmost vertex to leftmost vertex.
3. The perimeter of Q is minimum.

2 Recurrences

We are trying to construct a vertex chain of $P[1..n]$ with minimum length.First we need to sort the array $P[1..n]$ so that we are assured that $P[1]$ will be the leftmost and $P[n]$ will be the rightmost vertex. So,now we will draw edges from $P[1]$ to $P[n]$ then vice versa to get the upper vertex chain and lower vertex chain respectively.To get the minimum path we need to assign all points $P[1..n]$ to any one of the chains except $P[1]$ which is on both of them.
Let $C[i]$ be the minimum length possible for a vertex chain whose vertex set is $P[1..i]$ i.e the rightmost point of the vertex chain is $P[i]$. Let $d(P[i], P[j])$ denote the length of edge ($P[i]$,$P[j]$). Also let $f(i)$ denotes the index of the rightmost point before $P[i]$ that is on a different chain(upper or lower) than point $P[i]$.In this manner our solution will become $C[n]$
Now considering the minimum length vertex chain for $P[1..i]$, the rightmost edge is $(P[j], P[i])$
There are two cases depending on the value of j,

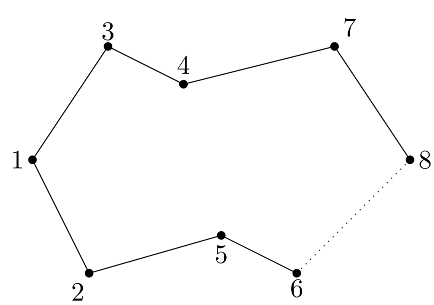

Figure 1: The points are sorted by x-coordinate. The dotted lines complete the polygon and the bold lines are the vertex chain cycle

Case 1 : j=i-1

C[i] = C[i-1] + d(P[i-1],P[i]) + d(P[f(n-1)],P[n])

Case 2 : $1 \leq j$ and $j < i-1$

C[i] = C[j] + d(P[j],P[i]) + d(P[f(j)],P[j+1]) + $\sum_{k=j+1}^{i-2} d(P[k], P[k+1]) + d(P[i-1], P[i])$

Here, the summation part adds all the remaining edges on the chain other than the one on which the rightmost point sits. Also, d(P[f(n-1)],P[n]) completes the polygon. Now we can see that the minimum possible perimeter is the minimum possible value we can get from above two cases by putting i=n.

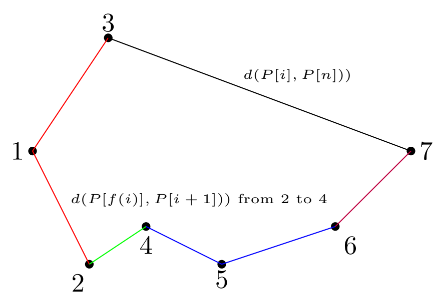

Figure 2: In this example, the rightmost point is 7 and rightmost edge is between 3 and 7. The red lines indicate the optimal solution for $C[i]$ (for points $P[1, \ldots, j]$). The green edge indicate the distance between the point $f(j)$ and $P[j+1]$. The blue edges between the points chain $P[j+1, i-1]$. The purple edge completes the polygon. The black edge indicates the edge between the point $P[j]$ and edge $P[i]$.

Recurrence Relation

Base Case : i=0

C[i] = 0

Recurring relation

C[i] = min[C[i-1]+d(P[f(i-1),P[i]),

min$_{\mathbf{j}}$[C[j]+d(P[j],P[i])+d(P[f(j)],P[j+1])+ $\sum_{k=j+1}^{i-2} d(P[k], P[k+1]) + d(P[i-1], P[i])$]]

3 Algorithm

Due to the overlapping subproblem structure, we can device a dynamic programming solution for this problem. We don't need to calculate sum of edges from some i to j repeatedly we can create a prefix array D for the edge lengths. Also f(i) also follows the following recurrence relation,

f(i) = f(i-1) if (P[i],P[i-1]) is an edge

f(i) = P[i-1] otherwise

4 Time and space complexities

Time Complexity

Require: $P[1,\ldots,n]$, a list of all points.

Sort the points as per x coordinate.
Create arrays $C[1,\ldots,n]$, $F[1,\ldots,n]$, S[1, ..., n] and $D[1,\ldots,n]$.
Fill the array D as per the rule, $D[i] = \sum_{a=1}^{i} d(P[a], P[a+1])$ as:
$D[1] = 0$
$i \leftarrow 2$
while $i \leq n$ **do**
  $D[i] = D[i-1] + d(P[i-1], P[i])$
end while

Computing the subproblems:
$C[1] = 0$
$F[1] = 1$
$i \leftarrow 2$
while $i < n$ **do**
  $arg1 = C[i-1] + d(P[i-1], P[i])$
  $arg2 = \min_{1 \leq k < i-1} \{C[k] + d(P[k], P[i]) + d(P[F[k]], P[k+1]) + D[i-1] - D[k+1]\}$
  if $arg1 < arg2$ **then**
    $F[i] = F[i-1]$
    $C[i] = arg1$
    $S[i] = i$
  else
    $F[i] = P[i-1]$
    $C[i] = arg2$
    $S[i] = \operatorname{argmin}_{k \in [1, i-2]} \{C[k] + d(P[k], P[i]) + d(P[F[k]], P[k+1]) + D[i-1] - D[k+1]\}$
  end if
end while
Computing $C[n]$,
$arg1 = C[n-1] + d(P[n-1], P[n]) + d(P[f(n-1)], P[n])$
$arg2 = \min_{1 \leq k < n-1} \{C[k] + d(P[k], P[n]) + d(P[F[k]], P[k+1]) + D[n] - D[k+1]\}$
if $arg1 < arg2$ **then**
  $F[n] = F[n-1]$
  $C[n] = arg1$
  $S[n] = n$
else
  $F[n] = P[n-1]$
  $C[n] = arg2$
  $S[n] = \operatorname{argmin}_{k \in [1, n-2]} \{C[k] + d(P[k], P[n]) + d(P[F[k]], P[k+1]) + D[n] - D[k+1]\}$
end if

Create lists $C1$ and $C2$ to keep the two vertex chains.
$i \leftarrow n$
while $i > 1$ **do**
  Add $S[i]$ in beginning of $C1$
  $i \leftarrow S[i]$
end while
Form $C2 = P - C1 - P[1] - P[n]$ (in same increasing order).
Using slope we will find the upper and lower chains
$slope1 = \frac{P[1].y - C1[1].y}{P[1].x - C1[1].x}$
$slope2 = \frac{P[1].y - C2[1].y}{P[1].x - C2[1].x}$
if $slope1 < slope2$ **then**
  ($C1$ is upper chain) Polygon $Q = P[1] + C1 + P[n] + C2$
else
  ($C2$ is upper chain) Polygon $Q = P[1] + C2 + P[n] + C1$
end if

First we sort the array which takes O(nlogn) time. Computing of array C takes O(n^2) time as we have to traverse from 1 to i for each C[i]. Hence overall time complexity = O(n^2)

Space Complexity

All arrays P,C and f are of size n so the space complexity comes out to be O(n).