CS5800: Algorithms Spring 2018
Assignment 3.1

Saptaparna Das

February 11, 2018

(a) To prove the problem exhibits optimal substructure property, we can prove by contradiction.
Lets say, OPT(i) be the minimum cost for points p1...pi and the last segment in the optimal partition contains points pj...pi.
Now, if we take out the last segment from the solution OPT(j-1) will be the cost so far for points p1...pj-1. Now, suppose there exists an more optimal solution at this point OPT'(j-1) $[OPT'(j-1) <= OPT(j-1)]$and we still have $pj...pi$ points left. So, if it is true, we can add pj to pi segment back to the solution.
It gives us an cost of $OPT'(j-1) + C + e(j,i) <= OPT(j-1) + C + e(j,i)$
Putting OPT'(j-1)+C+e(j,i)=OPT'(i) we can write, $OPT'(i) <= OPT(i)$ which contradicts our original assumption that OPT(i) is optimal solution.
So, we can say that, an optimal partition contains optimal partitions for sub-problems.

(b) Let OPT(i) be the penalty for the optimal partition of points p1,p2,...,pi. Now, if we know the points that consist the last segment in optimal partition, we can calculate OPT(i) using recursion. Say, the last segment has points from pj to pi. So,OPT(j-1) is the penalty for the optimal partition of points p1,p2,...,pj-1. Hence, we can deduce the following recursive relation:
if i=0 OPT(i)= 0
else $\min_{1 \leq j \leq i} (e(j,i) + C + OPT(j-1))$
where e(j, i) denotes sum of the minimum squared error for points pj, pj+1, , pi and C is penalty associated with each segment in the partition.

(c) **Pseudo code:**

```
//Assume we are given a sorted set of points (by x co-ordinate value)
getPartition(points[p1,p2..., pn])

// opt_track[n] tracks the last segment in the optimal solution
// for the points p1,p2,..pn
opt_track[n]
// OPT[j] is the minimum cost for the points p1,p2,..., pj
OPT[n]
//to store error values
e[n][n]

for i = 1 to n
  for j = 1 to i
      calculate e[j][i] for the segment pj, pj+1, ..., pi
//find optimal partition
//Base case
opt_track[0]=0
OPT[0]=0
min=infinity

for i = 1 to n
   for j = 1 to i
       x=e[j][i]+OPT[j-1]
       if(x<min)
            z=j
            min=x
   OPT[i]= min + C
   opt_track[i]=z
```

```
return opt_track
```

(d) **Analysis:**
The error calculation for each pair of j,i takes linear time. There are n such pairs, we need to take into consideration. So, the time taken is $O(n^3)$.For calculating the minimum cost and populating the tracker array, it takes another $O(n^2)$ time. So, Total time taken would be $O(n^3)$.
For space, The minimum cost is stored in one 1D array and we are using a 2D array to store the error or each pair of point, which takes up $O(n^2)$. So, space complexity is $O(n^2)$