CS 180 Homework 5

Prithvi Kannan

405110096

1. Consider the divide and conquer algorithm for finding the closest pair of points. Analyze the time complexity of the algorithm . Include and discuss a detailed discussion of how to manage points in the x-dimension and how to manage (and search) points in the y-dimension. (You should do this without consulting the book or your notes)

The divide and conquer algorithm goes as follows:

As a pre-processing step, the input array is sorted according to x coordinates.

1) Find the middle point in the sorted array, we can take P[n/2] as middle point.

2) Divide the given array in two halves. The first subarray contains points from P[0] to P[n/2]. The second subarray contains points from P[n/2+1] to P[n-1].

3) Recursively find the smallest distances in both subarrays. Let the distances be dl and dr. Find the minimum of dl and dr. Let the minimum be d.

4) From the above 3 steps, we have an upper bound d of minimum distance. Now we need to consider the pairs such that one point in pair is from the left half and the other is from the right half. Consider the vertical line passing through P[n/2] and find all points whose x coordinate is closer than d to the middle vertical line. Build an array strip[] of all such points.

5) Sort the array strip[] according to y coordinates. This step is O(nLogn). It can be optimized to O(n) by recursively sorting and merging.

6) Find the smallest distance in strip[]. This is tricky. From the first look, it seems to be a O(n^2) step, but it is actually O(n). It can be proved geometrically that for every point in the strip, we only need to check at most 7 points after it (note that strip is sorted according to Y coordinate). See this for more analysis.

2. Exercise 3 on page 314

3. Exercise 5 on page 316

<https://web.cs.wpi.edu/~cs2223/b05/HW/HW6/SolutionsHW6/>

The objective is to maximize the sum of the qualities of words. We start by writing the optimal solution recursively. OPT(j) = max from 1 ≤ k ≤ j of ( OPT(k-1) + quality(yk...yj)). From this recursion, we can write an algorithm as follows:

for (j = 1 to n) {

temp = -infinity

for (k = 1 to j) {

if temp < OPT[k-1] + quality(yk...yj)) then {

temp := OPT[k-1] + quality(yk...yj))

}

OPT[j] := temp

}

4. Exercise 10 on Page 321

5. Given a rod of length n inches and an array of prices that contains prices of all pieces of size smaller than n. Determine the maximum value obtainable by cutting up the rod and selling the pieces. For example, if length of the rod is 8 and the values of different pieces are given as follows, then the maximum obtainable value is 22 (by cutting in two pieces of lengths 2 and 6)

length j 1 2 3 4 5 6 7 8

---------------------------------

price j 1 5 8 9 10 17 17 20

<https://www.geeksforgeeks.org/cutting-a-rod-dp-13/>

6. Consider a row of n coins of values v1,…,vn, where n is even. We play a game against an opponent by alternating turns (you can both see all coins at all times). In each turn, a player selects either the first or last coin from the row, removes it from the row permanently, and receives the value of the coin. Determine the maximum possible amount of money we can win if we move first.

Example 1: [5; 3; 7; 10] : The user collects maximum value of 15 (10 + 5)

- Sometimes the greedy strategy works.

Example 2: [8; 15; 3; 7] : The user collects maximum value of 22 (7 + 15)

{ In general the greedy strategy does not work.