Algorithm Lab

Week 4: Maximum Subarray and Variation

Description

Subarrays are generalizing of substrings, a continuous subdomain of original space. Assume that we have an array $A = [a_1, a_2, \dots, a_n]$; we may have a subarray $S_A = [a_i, a_{i+1}, \dots, a_{j-1}, a_j]$ where $1 \leq i \leq j \leq n$. Note that the minimal size of a subarray is 0 and maximal size is n. If a subarray contains nothing, then its summation is 0.
We say a subarray S_1 is larger than subarray S_2 if summation of elements in S_1 larger than summation of elements in S_2. How can find the largest subarray of a given array in divide and conquer approach?

Instance: An array $A = [a_1, a_2, \dots, a_n], \forall a_i \in A, a_i \in \mathbb{R}$.

Result: Maximum subarray (can only return start index and length like $(3, 2)$ *which shows the subarray* $S_A = [a_3, a_4]$.

Algorithm Design

1 Divide data to 2 same-size parts.
2 Find the maximum subarray of 2 parts recursively.
3 Find the maximum subarray which cross the center line.
  3.1 Find the maximum subarray end with right-most element of left part.
    3.1.1 Set max-length, max-summation, current-length, current-summation by 0.
    3.1.2 Test each summation of each possible length and calculate the temporary result.
    3.1.3 If temporary result is better/larger than result store in max-variables, update them.
  3.2 Find the maximum subarray start at left-most element of right part.
    3.2.1 Doing things like step 3.1.1 to 3.1.3.
  3.3 The result is concatenation of step 3.1 and step 3.2.
4 Compare results of step 2 and step 3 and return the better (larger) 1.

Questions

Assume we have n points set $V = \{v_1 = \{x_1, y_1\}, v_2 = \{x_2, y_2\}, \ldots, v_n = \{x_n, y_n\}\}$ in a 2D Euclidean space. A pair is 2 points. We can measure Euclidean distance of the pair by Pythagoras theorem (i.e., $\forall p = \{v_a = \{x_a, y_a\}, v_b = \{x_b, y_b\}\}$ where $a < b$, then we have distance of pair p, $D(p) = \sqrt[2]{(x_b - x_a)^2 + (y_b - y_a)^2}$). The closest pair is the pair has smallest distance.

Instance: A set of pints $V = \{v_1\{x_1, y_1\}, v_2\{x_2, y_2\}, \ldots, v_n\{x_n, y_n\}\}$.

Result: A closest pair $\{v_a, v_b\}, a < b$ *that* $\forall 1 \leq i < j \leq n, D(\{v_a, v_b\}) \leq D(\{v_i, v_j\})$

---

1 Please modify above algorithm to design a divide and conquer algorithm to solve closest pair problem.

1.1 Hint: you can try sort the points by x-axis before finding.

2 Analysis space complexity and time complexity (should contains parameter n) of your algorithm.

3 Implement your algorithm to solve **ALG03A** on http://oj.csie.ndhu.edu.tw/

1.) Start
Scan n point pairs of (x, y)
Sort pairs by x increasing
Divide into 2 from the middle point M
↳ From here do the step for both halves recursively and find the shortest distance between both halves
Create subarray of points with distance x to the middle point lower than current minimum distance
Sort subarray by y
Find smallest distance in subarray
Return minimum distance

2.) $f(n) = 2\,O(n/2)$ subarrays in half $+ O(n \lg n)$ sorting algorithm $+ O(n)$ transverse the points

$f(n) = O(n \lg n)$ time

At most creates two subarrays of max size $(n/2)$ each

$O(n)$ space