Date: 20/04/2018

BLG 336E - Analysis of Algorithms II, Spring 2018

CRN: 21548

Project - II

Orçun Özdemir

150140121

Development Environment

Operating System:	Elementary OS
Language:	C++
Paradigm:	Object Oriented
IDE:	Atom
Compiler:	GCC
Data Structures:	Vector, Array
Libraries:	iostream, cstring, chrono, cmath, algorithm, string, fstream, cstdlib, vector

I used chrono library to measure running time and it requires ISO C++ 2011 standard, so you have to compile it with these commands.

- g++ main.cpp -o main -std=c++11
- g++ main.cpp -o main -std=gnu++11

Otherwise you will get this error on ITU SSH Linux Server!

"/usr/include/c++/4.8.2/bits/c++0x_warning.h:32:2: error: #error This file requires compiler and library support for the ISO C++ 2011 standard. This support is currently experimental, and must be enabled with the -std=c++11 or -std=gnu++11 compiler options."

Questions

Q1) Divide and conquer algorithm in which a problem with input size **n** always divided into **a** subproblems. f(n) represents the divide and combine time. The **a** and **b** are constant integers in master theorem.

Q2) In this project, we have to use Euclidean distance to calculate 3D points' distance. Also, the problem space isn't on 1D line, we have to sort all points regarding to one direction that we select. For example, I selected X line to sort all the points. After sorting operation, we should divide all points into equal parts recursively. Dividing and calculating left and right parts' distances is not enough! Because, the wanted two closest pair can be on different parts. To exemplify, there is a possibility to the closest pairs' points located one of them on the left part and the other one located on the right part. To look for these points we have to define a strip area and after sorting these coordinates regarding to Y value, if the closest pair distance in strip area closer than the other areas we should select this distance. Finally, when combining part finished the last returned value will be the closest pair distance.

Q3)

Pseudo-Code

```
divideandConquer of (vx, vy)
        where vx is v(1) .. v(N) sorted by x coordinate, and
           vy is v(1) .. v(N) sorted by y coordinate (ascending order)
if N ≤ 3 then
 return closest points of vx using bruteforce
else
 yl ← points of vx from 1 to [N/2]
 yr ← points of vx from [N/2]+1 to N
 midPoint ← vx([N/2])
 (dl, leftpart) ← divideandConquer of (xl, yl)
 (dr, rightpart) ← divideandConquer of (xr, yr)
 (dmin, pairMin) ← (dR, rightpart)
 if dl < dr then
   (dmin, pairMin) ← (dl, leftpart)
 endif
 nS ← number of points in yS
 (closest, divideandConquer) ← (dmin, pairMin)
 for i from 1 to nS - 1
  k ← i + 1
  while k ≤ nS and yS(k) - yS(i) < dmin
   if |yS(k) - yS(i)| < closest then
    (closest, divideandConquer) ← (|yS(k) - yS(i)|, {yS(k), yS(i)})
   endif
   k ← k + 1
  endwhile
 endfor
 return closest, divideandConquer
endif
```

Complexity

I used sort() STL function to sort points and its time complexity is O(nlogn). This algorithm divide all the points into 2 parts with recursive calls. When divided it finds the mid lane area in O(n) time complexity. Moreover, it takes O(n) time to divide mid lane points into 2 sets. Finding closest pairs in mid lane area takes O(n) time, because it's bounded to linear time. So the final time complexity is O(nlogn) + O(n) + O(n) + O(n) = O(nlogn). Also, I used vector to store the values for algorithm. It has linear space complexity O(n).

Recurrence Relation

$a = 2, b = 2$

$T(n) = 2T(n/2) + O(n) + O(n) + O(n)$
$T(n) = T(n\log n)$

Q4)

	Data Size	Total Number of Calculations	Running Time
Brute-Force	1000	499509	18393 microseconds
Divide & Conquer	1000	5203	488 microseconds
Brute-Force	5000	12497511	409179 microseconds
Divide & Conquer	5000	32496	2603 microseconds
Brute-Force	10000	49995018	1.59646 seconds
Divide & Conquer	10000	67175	5409 microseconds
Brute-Force	25000	312487524	10.4292 seconds
Divide & Conquer	25000	183396	14497 microseconds

While brute-force technique's runtime increases exponentially, divide and conquer approach stays much more efficient. Even with 25K points, divide and conquer can solve the problem under 0.2 seconds.