BLG336E - Analysis of Algorithms II
2017-2018 Spring, Project 2 Report
Submission Deadline: 20.04.2018, 20:00

Kadir Emre Oto (150140032)

1) Explain the master theorem with your own words briefly (max: 3 lines). What do a and b mean in divide- and-conquer approach? (max: 3 lines)

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

b means how many parts the algorithm divides, and a means how many times the function is called. Generally, a and b are equal. $f(n)$ means the complexity of conquer.

2) Present your problem formulation in detail.

In this problem, we are trying to find the distance of the closest points in a 3D space. The simplest solution is brute-force solution, which checks all points to determine the closest pair. But the brute-force algorithm has $O(N^2)$ time complexity, which is very slow for the values of huge N's.

The idea that reduces the complexity to $O(nlog^2n)$ is divide and conquer method. The method says "Divide the space into smaller pieces, solve them in separately and finally conquer the results that obtained smaller parts. In order to apply this method, first I sort all points according to an axis like X, then divide the points two almost equal part which is going to solved individually and finally conquer the results. At this step, there is one thing to consider: the result may come from the pairs that are in different parts, therefore some points which are far at most minimum of the answers of the parts from the separation point should be compared too.

3) How does your algorithms work?

- **Write your pseudo-code:**
- **Write the time and space complexity of your algorithm on your pseudo-code:**
  The *space complexity* of the algorithm is $O(N)$, just points are hold in an array.

function Solve(Point** array, int size):	$O(N)$
if (size <= 10)	$O(1)$
distance is infinity initially	$O(1)$
for each point pair in [array, array + size]:	$O(N)$
set distance if distance of current pair is smaller than distance variable	$O(1)$
return distance	$O(1)$
else:	$O(1)$
Divide the point array into two almost equal parts	$O(1)$
Solve both of them separately using the Solve function	$2 * O(N/2)$
Conquer the solutions and check the closest pairs between parts	$O(logN)$
return distance calculated	$O(1)$

- **Write the recurrence relation of your algorithm. (T(base) and T(n)=?)**

$$T(n) = 2T\left(\frac{n}{2}\right) + O(nlogn)$$

As f is asymptotically greater than $n^{log_2 2} = n$ by a logarithmic factor, the equation does not satisfy any case of master theorem. So master theorem cannot say about the recurrence.

4) Analyze and explain the algorithm results in terms of:

- **Total number of distance calculations for each input file**
- **The running time for each input file**

Input	Output	Running Time
data1000.txt	The distance is 16.911535	0 millisecond
	Number of total distance calculations is 21521	
data5000.txt	The distance is 37.363083	0 millisecond
	Number of total distance calculations is 267787	
data10000.txt	The distance is 30.265492	2 milliseconds
	Number of total distance calculations is 717497	
data25000.txt	The distance is 39.673669	8 milliseconds
	Number of total distance calculations is 2883247	

Compilation Command: g++ kod.cpp -o kod -O2 -std=c++11
Running Command: ./kod data/data1000.txt