CpSc 2120: Algorithms and Data Structures

Instructor: Dr. Brian Dean
Webpage: `http://www.cs.clemson.edu/~bcdean/`
Handout 6: Lab #4

Fall 2014
MWF 9:05-9:55
Vickery 100

1 Finding the Closest Pair of Points

In this lab exercise, you will use "spatial hashing" to find the closest pair among 1 million points spread uniformly across a unit square in the 2D plane. Although this problem is easily solved in $\Theta(n^2)$ time by comparing all pairs of points, this solution is too slow for input sizes n on the order of 1 million, as is the case here.

The input file for this lab is:

`/group/course/cpsc212/f14/lab03/points.txt`

It contains 1 million lines, with two space-separated real numbers per line giving the x and y coordinates of a point. All points (x, y) live strictly inside the unit square described by $0 \leq x < 1$ and $0 \leq y < 1$. Remember that the distance between two points (x_1, y_1) and (x_2, y_2) is given by

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}.$$

There are no source code files to start with; you will need to write your program starting from scratch this time.

2 The Approach

To find the closest pair of points quickly, you will divide the unit square containing the points into a $b \times b$ grid of square "cells", each of size $1/b \times 1/b$. Each point should be "hashed" to the cell containing it. Afterward, each point needs only to be compared to the other points within its cell, and the 8 cells immediately surrounding its cell – this should result in many fewer comparisons than if we simply compared every point to every other point. You will need to select an appropriate value of b as part of this lab exercise. You may want to consider what are the dangers in setting b too low or too high.

Internally, the grid of cells should be stored as a 2-dimensional array of pointers to linked lists, with each linked list containing the set of points belonging to a single cell. The array of cells must be *dynamically* allocated (with the *new* command) and subsequently de-allocated at the end of your program (with the *delete*) command. This means that, since each cell of the table is itself a pointer to a node of a linked list, the top-level variable representing the 2D array of pointers will be of type "Node ***" (with three *'s!), assuming "Node" is the name of the structure representing a node in your linked lists.

Your program should consist of the following major steps:

1. Allocate and initialize 2D array of pointers to linked lists.
2. Read input file, inserting each point into the appropriate linked list based on the cell to which it maps.
3. For each point, compare it to all the points within its cell and the 8 adjacent cells; remember the smallest distance obtained during this process.
4. De-allocate 2D array and linked lists.
5. Print out minimum distance.

Part of this lab also involves figuring out a good choice for the value of b. Please include in a comment in your code a brief description of why you think your choice of b is a good one.

3 Grading

For this lab, you will receive 8 points for correctness and 2 points for having well-organized, readable code. Zero points will be awarded for code that does not compile, so make sure your code compiles on the lab machines before submitting!

Final submissions are due by 11:59pm on the evening of Friday, September, 12. No late submissions will be accepted.