CSE 5311 Lab Assignment 1

Due July 6, 2011

Goals:

1. Review of binary search trees.

2. Understanding of randomized search trees (treaps).

Requirements:

1. Write (and test) a C/C++ program that uses a treap to implement the sweepline algorithm for 2-d closest pairs (Notes 17) in expected O(*n* log *n*) time. Your program must compile and execute on at least one of `omega.uta.edu` or Visual Studio.

   The input (`stdin`) to your program will a single line with the number of points (*n*) followed by the *n* points as pairs of integer coordinates, one pair per line. Do not prompt for an input file name! All coordinates will be in the range `-16000 . . . 16000`, inclusive.

   Besides outputting the coordinates of the two closest points, your program should provide performance metrics such as: maximum BST size, CPU time, and the number of rotations.

2. Email your code (as attachments) to `miao.zhang@mavs.uta.edu` before 3:15 pm on July 6. The subject should include your name as recorded by the University.

Getting Started:

1. You may borrow from the code at `http://www.cs.fiu.edu/~weiss` (or other places - besides each other), but be sure to give appropriate credit in your comments.

2. `n` will not exceed 100,000,000.

3. Be careful with your randomly generated priorities.

4. To help assure the correctness of your code, it is convenient to compare results with the obvious $O\left(n^2\right)$ method for $n \le 40{,}000$.

5. The range restriction for coordinates allows this assignment to be done without floating-point arithmetic (e.g. `sqrt()` is not needed) by comparing squares of distances rather than the usual $\sqrt{\Delta x^2 + \Delta y^2}$. `short` integers can be used, but are not required.

6. Compiling with `-O3` optimizations can be helpful when working with large *n*.

7. Even though Notes 17 describes the processing in terms of predecessor and sucessor navigation, it is convenient to code the attempt to improve δ by an efficient recursive range search (that avoids unnecessary y-coordinate comparisons) ***before*** inserting point *k* + 1.

8. The preprocessing sort may be done using the library `qsort()`.