CS 584/684 Spring 2017 Homework 2 – due noon, Wednesday, April 19 2017

Your solutions to problems 1-3 should be type-set in LaTeX and submitted in both `.tex` and `.pdf` form, with file names `hw2.tex` and `hw2.pdf`. These two files, plus any additional source files invoked from your `.tex` file (such as pictures), should be bundled together into a single `.zip` file named *your-last-name*`-tex-hw2.zip`. Your code for problem 4 should be submitted in a single, separate `.zip` file named *your-last-name*`-code-hw2.zip` with contents as described below in the description of problem 4.

Submit by emailing to `hamialex@pdx.edu` including the two zip files as separate attachments and including "`CS584 HW2`" in the subject line.

All algorithms must be accompanied by proofs of correctness and of running time. In some cases, use of appropriate pictures may greatly improve your proofs. Note that hand-drawn pictures can be photographed or scanned and then incorporated into your LaTeX file.

1. Do CLRS 33-2.3.

2. Prove the following statement. Given a convex polygon with n vertices, it is possible to build a data structure of size $O(n)$ with the following characteristics: Given an arbitrary point (the query), whether or not it lies inside the polygon (including its boundary) can be decided in $O(\log n)$ time.

3. Design an algorithm that, given n horizontal line segments sorted by increasing y-coordinate, determines whether or not there exists a line that can pass through all of them. Note: Your algorithm does *not* have to compute such a line if it exists, just answer whether one exists or not. Your algorithm must run in $O(n)$ time. **Prove that your algorithm is correct and has the required running time.** Hint: There is a line separating any two non-intersecting convex polygons. (You may use this fact without further proof.)

4. Implement the Closest Pair algorithm from lecture (or from CLRS Section 33.4), with one change: rather than measuring distance using the ordinary Euclidean metric, use Manhattan distance. The *Manhattan distance* between $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$ is defined to be $|x_1 - x_2| + |y_1 - y_2|$. (Why "Manhattan"? Think about the effective walking distance between two locations on a rectangular street grid.) Your implementation should continue to take $O(n \log n)$ time. This time you may use a library sorting function, if you need one. You should include a comment at the top of the program describing the impact (if any) of changing from Euclidean to Manhattan distance measurement.

Your program should take one command line argument, which is the name of an input file. The format of that file will be as follows:

- First line contains a number $C \geq 0$ of test cases in the file.
- Then come C test cases, each of the following form:
  - Line containing number $n \geq 2$ of points in the test
  - Line containing the $2n$ coordinates of the n points in the order $x_1\ y_1\ \ldots\ x_n\ y_n$, with exactly one space separation between each number.

You can assume that counts are integers $\leq 10^6$ and that the point coordinates are integers in the range $[-10^9 \,..\, 10^9]$.

Your program should output (to `stdout`) one line for each test case, of the form "`Case i:` n" where n is a non-negative integer representing the shortest Manhattan distance between any two points in the input for that case.

Example Input:

```
2
5
0 0 2 0 0 2 2 2 2 2
3
0 0 5 5 100 100
```

Corresponding Example Output:

```
Case 1: 0
Case 2: 10
```

Warning: If your output format is not correct (even spacing), you will get no credit; this problem will be graded by doing a `diff` against a standard output file.

Place your code (one or more source files) together with a `Makefile` in a fresh subdirectory with the name *your-last-name*`-code-hw2`. Then create a single `zip` archive with the name *your-last-name*`-code-hw2.zip` containing just that directory and its contents. It should be possible to build an executable file called `hw2` and test it on an input file `/path/to/foo` by the following steps:

1. `unzip` *your-last-name*`-code-hw2.zip`
2. `cd` *your-last-name*`-code-hw2`
3. `make`
4. `./hw2 /path/to/foo`