Algorithm Design and Analysis

Homework #1

Due: 5pm, Friday, October 4, 2013

TA email: ada@csie.ntu.edu.tw

=== Homework submission instructions ===

- For Problem 1, commit your source code and a brief documentation to the SVN server (katrina.csie.ntu.edu.tw). You should create a new folder "`hw1`" and put these two files in it.
- The filenames of the source code, Makefile, and the documentation file should be "`main.c`" and "`report.txt`", respectively; you will get some penalties in your grade if your submission does not follow the naming rule. The documentation file should be in plain text format (`.txt` file). In the documentation file you should explain how your code works, and anything you would like to convey to the TAs.
- You have to login to judgegirl system (http://katrina.csie.ntu.edu.tw/judgegirl) to initiate the juding process of your program.
- For Problem 2 through 5, submit the answers via the SVN server (electronic copy) or to the TA at the beginning of class on the due date (hard copy).
- Except the programming assignment, each student may only choose to submit the homework in only one way; either all in hard copies or all via SVN. If you submit your homework partially in one way and partially in the other way, you might only get the score of the part submitted as hard copies or the part submitted via SVN.
- If you choose to submit the answers of the writing problems through SVN, please combine the answers of all writing problems into only ONE file in the pdf format, with the file name in the format of "`hw1_[student ID].pdf`" (e.g. "`hw1_b01902888.pdf`"); otherwise, you might only get the score of one of the files (the one that the grading TA chooses).
- Discussions with others are encouraged. However, you should write down your solutions with your own words. In addition, for each problem you have to specify the references (the Internet URL you consulted with or the people you discussed with) on the first page of your solution to that problem.
- **NO LATE SUBMISSION IS ALLOWED** for the homework submission in hard copies - no score will be given for the part that is submitted after the deadline. For submissions via SVN (including the programming assignment and electronic copies of the writing problems), up to one day of delay is allowed; however, the score of the part that is submitted after the deadline will get some penalties according to the following rule (the time will be in seconds):

$$\text{LATE SCORE} = \text{ORIGINAL SCORE} \times (1 - DelayTime/86400)$$

Problem 1. Rectangle Man

(30%)

According to the latest news, the archaeologists have found the monument left from the aliens called Rectangle Man. They have left a puzzle in the monument. People who are able to solve the puzzle may found the treasure left behind by the aliens.

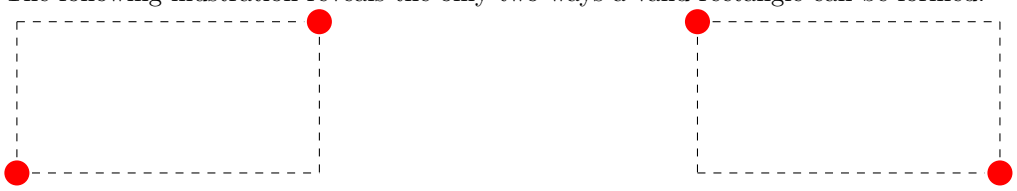
The monument says,

"You should know that a rectangle can be determined by four points. But does it really needs up to four points to determine a rectangle? No! In fact, only two points on the diagonal are needed.

We have left a map of points. As long as you can determine the maximum number of rectangles those points can form, you can find the treasure we left.

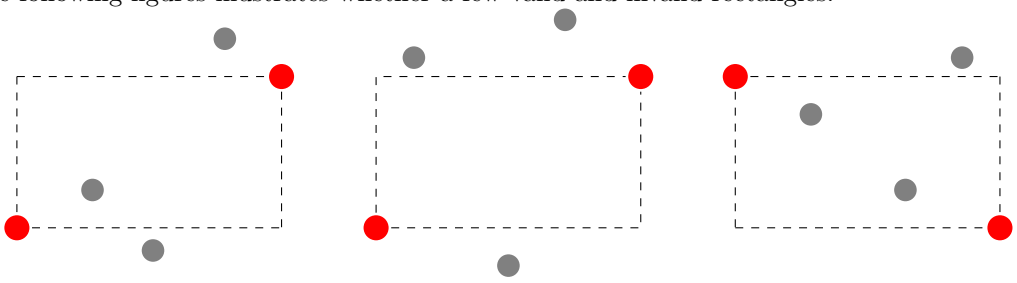
There is one last thing you should know: we only care about rectangles whose edges are parallel or perpendicular to the x axis."

The following illustration reveals the only two ways a valid rectangle can be formed:



You can see that two imaginary points and two points selected from the given map of points are sufficient to form a rectangle.

Note: For a rectangle to be valid, there can be no points inside the boundary of that rectangle. The following figures illustrates whether a few valid and invalid rectangles.



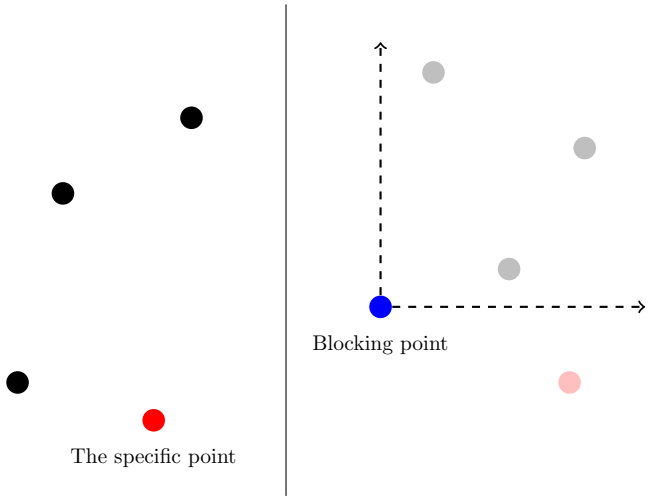
Invalid Rectangle *Valid Rectangle* *Invalid Rectangle*

Your goal is to write a program that utilizes a divide-and-conquer algorithm to solve the problem and retrieve the treasure left by the aliens.

Hint 1: Sorting the points by their coordinates and dividing them into two groups would help a lot.

Hint 2: When you want to find the possible points which can form rectangles with a specific point, you can find that there may be some points blocking all the points behind them from forming rectangles. It may be beneficial to use some kind of data structure to maintain those "blocking

points". In the following figure, when we consider the rectangles the red point can form with points on the right side, those grey points cannot be the ones to form valid rectangles since they are blocked by the blue point, while the pink point is certainly an option.



Report(10%)

Please write down how you derived the algorithm, how you implemented it, and analyze the running time of your algorithm using asymptotic notations.

Coding(20%)

Input format: The first line contains the total number of points, n, $0 < n \leq 50000$. And the following n lines contains each point's coordinate. For any pair of points a, b in the map, $a.x \neq b.x$ and $a.y \neq b.y$ where $a.x$ and $a.y$ are the X and Y coordinates of a, respectively, and $b.x$ and $b.y$ are the X and Y coordinates of b, respectively. In addition, the X and Y coordinates of the points can all be represented in **integer**.

Output format: Please output the maximum possible number of rectangles in a single line.

Sample input:

5
4 4
1 2
2 3
5 5
3 1

Sample output:

6

Problem 2. Solving Recurrences (16%)

a. (2%) $T(n) = 2T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n$
   Use the substitution method to prove that $T(n) = O(n \log n)$

b. (2%) $T(n) = T\left(\left\lceil \frac{n}{2} \right\rceil\right) + T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + 1$
   Use the substitution method to prove that $T(n) = O(n)$

c. (2%) $T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + \Theta(n)$
   Draw a recursion tree, and use it to prove that $T(n) = \Omega(nlgn)$

For the following subproblems, give tight asymptotic bounds (i.e. $\Theta(n^2)$, $\Theta(nlgn)$) and the corresponding proofs. You can use whatever methods you want.

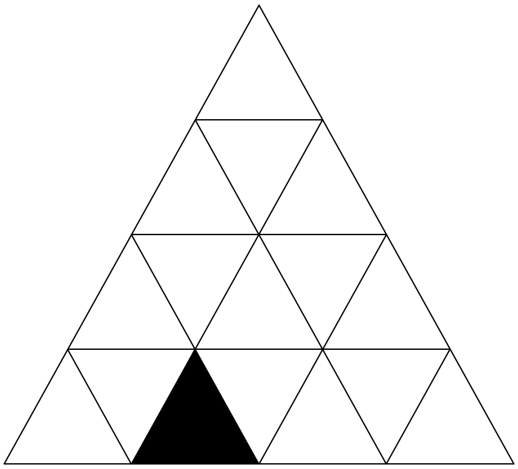
d. (3%) $T(n) = 3T\left(n^{\frac{1}{3}}\right) + \log n$

e. (2%) $T(n) = 7T\left(\frac{n}{4}\right) + \Theta(n^2)$

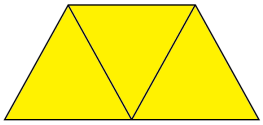
f. (3%) $T(n) = 2T\left(\frac{n}{2}\right) + n \log_2 n$

g. (2%) $T(n) = 2T(n-1) + 1$

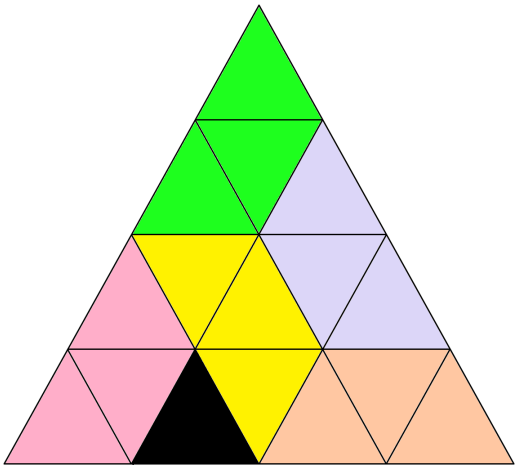
Assume that T(1)=1 for all recurrences.



(a) The board for $n = 4$



(b) The piece to be filled in the board



(c) The board covered by the pieces for $n = 4$

Figure 1: The tiling problem

Problem 3. (15%) Assume that n is a positive integer which is a power of two. We have a triangular board which is composed of equilateral trangles, and one triangle on the edge gets randomly removed (see Figure 1(a)). You are asked to cover the entire board with the trapezoical piece shown in Figure 1(b). See Figure 1(c) for an example. Note that the piece can be rotated. All tiles on the board except the one taken way have to be covered, and no tiles can be left outside of the board. In this problem, we ask you to design an algorithm to solve this problem with the divide-and-conquer strategy. Analyze the running time of your algorithm by using the recurrences.

Problem 4. Selection in two lists (15%)

Given two sorted list with length m and n, design an $O(\log m + \log n)$-time algorithm to compute the k-th smallest element in the union of the two given lists. Please give the pseudo code of your algorithm in detail (10%), and explain why the time complexity of your algorithm satisfies the requirement (5%).

Problem 5. (24%)

1. (8%)The following pseudo code implements a function that multiplies two very large numbers stored in two binary strings.

```
multiply(value1, value2)
    length1 = number of digits of value1
    length2 = number of digits of value2
    if length1 < 5 && length2 < 5
        // It's small enough that we can do
        // arithmetic operations in constant time.
        return value1 * value2
    k = max(length1, length2) / 2 + max(length1, length2) % 2
    value1_high = first length1 - k digits of value1
    value1_low = last k digits of value1
    value2_high = first length2 - k digits of value2
    value2_low = last k digits of value2
    z1 = multiply(value1_high, value2_high)
    z2 = multiply(value1_high, value2_low)
    z3 = multiply(value1_low, value2_high)
    z4 = multiply(value1_low, value2_low)
    add 2*k trailing 0s to z1
    add k trailing 0s to z2 and z3
    return z1+z2+z3+z4
```

Assume that the larger number of the two has n digits. The running time of the procedure would be $T(n) = \begin{cases} 4T(\frac{n}{2}) + O(n), \text{ if } n \geq 5 \\ \Theta(1), \text{ if } n < 5 \end{cases}$. Therefore, the procedure runs in $O(n^2)$ time. Please modify the procedure and justify that your procedure can run in $O(n^{\log_2 3})$ time.

2. (8%)Please give a pseudo code for `power_of_ten_to_binary(n)` which returns a binary string representing 10^n where n is a positive integer. You have to give the recurrences to represent its running time and justify that your procedure runs in $O(n^{\log_2 3})$ time.

3. (8%)Please give a pseudocode for `decimal_to_binary(x)` which returns a binary string representing the n-digit decimal integer x and justify that your procedure runs in $O(n^{\log_2 3})$ time.