University of Waterloo
CS 341 — Algorithms
Spring 2014
Assignment 2

Due: Wed, June 4, 2014, at 3:00pm.

Assignment Guidelines

- Use A2-coversheet.pdf for the first page of your assignment or format the first page of your assignment in *exactly* the same manner.
- Answers the questions in the same order that they are given in the assignment.
- Make it very clear to the marker where the answer to one question ends and the answer to the next one begins.
- You may lose up to 20% on a question because it is difficult to read or difficult to understand.
- You may only use the techniques discussed in class.
- Your whole assignment is considered late if either the programming question or the written portion is submitted late.

Questions to think about - do not hand in, answers will not be provided

Consider the algorithm for finding the closest pair of points in the plane described in class.

i At each recursive step, the plane is subdivided using a vertical line and in each combining step, pairs of points that are within a distance of δ from the vertical need to be considered. Justify that this work can be done in $\Theta(n)$ given two lists of points - one sorted by the x-coordinate and the other sorted by the y-coordinate.

ii Suppose a naive student implements the algorithm by performing a sort at each step; give the recurrence relation for the runtime of this implementation.

iii Prove that the general form of the Master Theorem (provided on slide 62) cannot be used to solve the recurrence from part ii).

Assignment Questions

1. **[20 points]** Recursion Tree

   (a) Solve the following recurrence relations using the recursion tree method. Express your solution in terms of a Θ bound on $T(n)$. Show your work clearly. You do not have to draw the tree at incremental steps, however, draw the final tree showing at last 4 levels (including the root and leaves).

      i You may assume that n is a power of 3.

$$T(n) = \begin{cases} 4, & n = 1, \\ 5T(n/3) + n\sqrt{n}, & n > 1. \end{cases}$$

      ii You don't have to show all nodes on the third level but show enough that we know what you are doing.

$$T(n) = \begin{cases} 2, & n \leq 1, \\ 6T(\frac{3}{7}n) + n^2, & n > 1. \end{cases}$$

   (b) Solve part a) using the Master Theorem or explain why the Master Theorem cannot be used.

2. **[10 points]** Guess and Verify

   Use induction to verify the following recurrence with the corresponding guess:

   $T(n) = 2T(\lfloor n/2 \rfloor) + \sqrt{n}$, for $n > 1$ and $T(1) = 3$
   Guess: $T(n) = O(n)$.

3. **[10 points]** Matrix Multiplication

   Using Strassen's divide-and-conquer algorithm for matrix multiplication, that runs in $\Theta(n^{log_2 7})$, as a building block (you do not need to provide the details of this algorithm) describe algorithms to solve the following problems and determine the complexity as a function of k and n.

   (a) Compute the product AB where A is an n by kn matrix and B is a kn by n matrix.

   (b) Compute the product BA where A is an n by kn matrix and B is a kn by n matrix.

4. **[20 points]** Programming Question

   A Nuclear Waste Facility (NWF) stores the nuclear waste in cylindrical barrels, each of radius R. Two barrels are considered to be too close to each other if the distance between their shells is less than or equal to specified constant D. Each barrel position on the yard is given by the x, y coordinates of its center. Given the coordinates of n barrels in the NWF we need to find for each barrel the barrels that are too close to it.

   Input and output The input consists of $n+1$ lines. The first line consists of three numbers: a positive integer n (the number of barrels), R (the radius of all barrels), and D (the minimum separation). The next n lines each have three numbers: a positive integer b_i (the barrel ID), and x_i and y_i (the coordinates of the barrel). Barrel IDs are distinct.

   The output is to have n lines, each having the following: a barrel ID followed by the IDs of barrels that are too close to it. The output has to be in increasing order of barrel IDs.

   Sample input

   ```
   4 2.3 0.2
   7 0.0 0.0
   11 4.6 0.1
   4 0.2 4.7
   6 9.2 0.0
   ```

   Sample output

   ```
   4 7
   6 11
   7 4 11
   11 6 7
   ```

   (a) Design and implement an efficient divide-and-conquer algorithm for this problem. Submit a file `waste.cpp`.

   (b) In comments at the beginning of the file, give a brief description of your algorithm and analyze the runtime complexity. You do not need to do a line by line analysis but cover the main ideas of your algorithm.