Assignment 2

Institute of Computing Technology,
Chinese Academy of Sciences, Beijing, China

November 2, 2012

Notice

1. Due date
   - Nov. 2, 2012 for CS711008Z.
   - Nov. 14, 2012 for CS6012.
2. Please send your answer in hard copy.
3. You can choose one problem from Problem 1-4, and one problem from Problem 5-6.
4. When you're asked to give an algorithm, you should do at least the following things:
   - Describe your algorithm in words or pseudo-code;
   - Prove that your algorithm can give the right answer;
   - Analyse the complexity of your algorithm.
5. You can present your answer in English or in Chinese.

1 Divide and Conquer

A group of n Ghostbusters is battling n ghosts. Each Ghostbuster is armed with a proton pack, which shoots a stream at a ghost, eradicating it. A stream goes in a straight line and terminates when it hits the ghost. The Ghostbusters decide upon the following strategy. They will pair off with the ghosts, forming n Ghostbuster-ghost pairs, and then simultaneously each Ghostbuster will shoot a stream at his chosen ghost. As we all know, it is very dangerous to let streams cross, and so the Ghostbusters must choose pairings for which no streams will cross. Assume that the position of each Ghostbuster and each ghost is a fixed point in the plane and that no three positions are collinear.

- Argue that there exists a line passing through one Ghostbuster and one ghost such the number of Ghostbusters on one side of the line equals the number of ghosts on the same side. Describe how to find such a line in $O(n \log n)$ time.
- Give an $O(n^2 \log n)$-time algorithm to pair Ghostbusters with ghosts in such a way that no streams cross.

2 Divide and Conquer

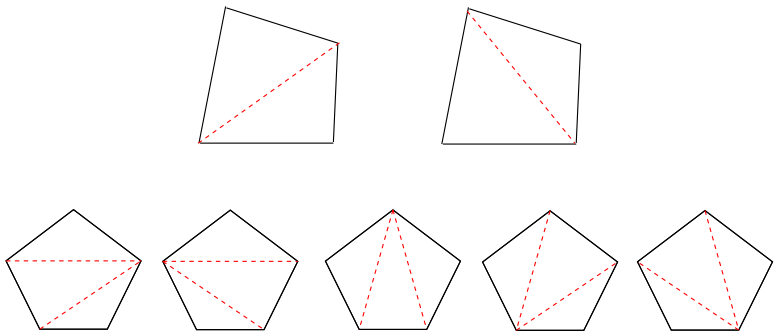
You are interested in analyzing some hard-to-obtain data from two separate databases. Each database contains n numerical values-so there are $2n$ values total-and you may assume that no two values are the same. You'd like to determine the median of this set of $2n$ values, which we will define here to be the n^{th} smallest value.

However, the only way you can access these values is through *queries* to the databases. In a single query, you can specify a value k to one of the two databases, and the chosen database will return the k^{th} smallest value that it contains. Since queries are expensive, you would like to compute the median using as few queries as possible.

Give an algorithm that finds the median value using at most $O(\log n)$ queries.

3 Divide and Conquer

Given a convex polygon with n vertices, we can divide it into several separated pieces, such that every piece is a triangle. When $n = 4$, there are two different ways to divide the polygon; When $n = 5$, there are five different ways.



Give an algorithm that decides how many ways we can divide a convex polygon with n vertices into triangles.

4 Counting Inversions

Recall the problem of finding the number of inversions. As in the course, we are given a sequence of n numbers $a_1, ..., a_n$, which we assume are all distinct, and we difine an inversion to be a pair $i < j$ such that $a_i > a_j$.

We motivated the problem of counting inversions as a good measure of how different two orderings are. However, one might feel that this measure is too sensitive. Let's call a pair a *significant inversion* if $i < j$ and $a_i > 3a_j$.

Given an $O(n \log n)$ algorithm to count the number of significant inversions between two orderings.

5 Counting Inversions

The attached file *Q5.txt* contains 100,000 integers between 1 and 100,000 (each row has a single integer), the order of these integers is random and no integer is repeated.

1. Write a program to implement the SORT-AND-COUNT algorithms in your favorite language, find the number of inversions in the given file.

2. In the lecture, we count the number of inversions in $O(n \log n)$ time, using the MERGE-SORT idea. Is it possible to use the QUICK-SORT idea instead ?
   If possible, implement the algorithm in your favorite language, run it over the given file, and compare its running time with the one above. If not, give a explanation.

6 Sorting

The MERGE-SORT, QUICK-SORT and MODIFIED-QUICK-SORT algorithms in the lecture slides are all recursive. However, by using a stack, all of them can be implemented in a iterative way.

1. Implement both the recursive version and iterative version of the three algorithms in your favorite language.
2. Run your implementation over three integer arrays whose sizes are $100,000$, $1,000,000$, $10,000,000$ respectively. Record their running times and make a comparison.

Note: You can show the results using a diagram or a table.