091M4041H - Algorithm Design and Analysis

Assignment 1

September 27, 2018

Notice:

1. The assignment contains two parts.
   (a) For problems 1-6, please submit your answer in hard copy AND submit a digital version to UCAS website `http://sep.ucas.ac.cn`.
   Hard copy should be submitted before 9 am. October 12 and digital version should be submitted before 11 pm. October 12.
   (b) For problems 7-8, you need finish them on the website `http://theory.ict.ac.cn/grad_oj` before 11 pm. October 12.
2. You can choose **three** from problems 1-6.
3. For problems 1-6, you should do at least the following things:
   (a) Describe your algorithm in natural language **AND** pseudo-code;
   (b) Draw a "subproblem reduction graph", where nodes represent subproblems, and edges describe the "reduction relationship" between them for every problem you choose in problems 1-6;
   (c) Prove the correctness of your algorithm;
   (d) Analyse the complexity of your algorithm.
4. For problems 7-8, you can implement your algorithm in C/C++/Java/Python/Pascal.

1 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.

2 Divide and Conquer

Given a binary tree, suppose that the distance between two adjacent nodes is 1, please give a solution to find the maximum distance of any two node in the binary tree.

3 Divide and Conquer

Consider an n-node complete binary tree T, where $n = 2^d - 1$ for some d. Each node v of T is labeled with a real number x_v. You may assume that the real numbers labeling the nodes are all distinct. A node v of T is a *local minimum* if the label x_v is less than the label x_w for all nodes w that are joined to v by an edge.

You are given such a complete binary tree T, but the labeling is only specified in the following *implicit* way: for each node v, you can determine the value x_v by *probing* the node v. Show how to find a local minimum of T using only $O(\log n)$ *probes* to the nodes of T.

4 Divide and Conquer

Suppose now that you're given an $n \times n$ grid graph G. (An $n \times n$ grid graph is just the adjacency graph of an $n \times n$ chessboard. To be completely precise, it is a graph whose node set is the set of all ordered pairs of natural numbers (i, j), where $1 \leqslant i \leqslant n$ and $1 \leqslant j \leqslant n$; the nodes (i, j) and (k, l) are joined by an edge if and only if $|i - k| + |j - l| = 1$.)

We use some of the terminology of problem 3. Again, each node v is labeled by a real number x_v; you may assume that all these labels are distinct. Show how to find a local minimum of G using only $O(n)$ probes to the nodes of G. (Note that G has n^2 nodes.)

5 Divide and Conquer

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

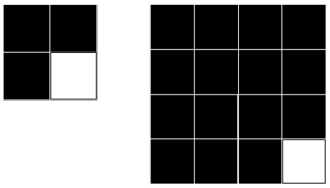
We motivated the problem of counting inversions as a good measure of how different two orderings are. However, one might feel that this measure is too sensitive. Let's call a pair a $significant\ inversion$ if $i < j$ and $a_i > 3a_j$. Given an $O(n \log n)$ algorithm to count the number of significant inversions between two orderings.

6 Divide and Conquer

Given a table M consisting of $2^n * 2^n$ blocks, we want to fill it with a L-shaped module (consisting of three blocks). The L-shaped module is shown below.

Please give a fill method, so that the last element of the table ($M_{2^n,2^n}$) is empty.

For example:



7 Divide and Conquer

Find the k^{th} largest element in an unsorted array. Note that it is the kth largest element in the sorted order, not the k^{th} distinct element.

INPUT: An unsorted array A and k.

OUTPUT: The k^{th} largest element in the unsorted array A.

8 Divide and Conquer

Implement the algorithm for the closest pair problem in your favourite language.

INPUT: n points in a plane.

OUTPUT: The pair with the least Euclidean distance.