CMPT 307 (2013)
Assignment 4
November 7, 2013

The following problems are related to divide-and-conquer algorithmic paradigm.

Problems from the text (page 246):

5.1 You can assume that the numerical values in each database of size n are already sorted. We are interested determining the median value of $2n$ elements in $O(logn)$ time.

5.2 It is very similar to the problem of counting inversions. Similar approach will work here.

5.3 Divide the bank cards into two piles S_1 and S_2. Recursively check if S_i has an equivalence class of size more than $n/4$. If such a class exists in one pile, check if that class exists in S (only $O(n)$ calls to equivalence tester for the last step).

5.6

The following problems are taken from the book Algorithms by Dasgupta, Papadimitriou and Vazirani.

1. Use the divide-and-conquer integer multiplication algorithm to multiply the two binary integers 10011011 and 10111010.

2. Show that for any positive integers n and any base b, there must some power of b lying in the range $[n, bn]$. (For $n = 1111$ and $b = 2$, $2^{11} = 2048$ lies in the interval $[1111, 2222]$.)

3. Suppose you are choosing between the following three algorithms:

   - Algorithm A solves problems by dividing them into five subproblems of half the size, recursively solving each subproblem, and then combining the solutions in linear time.
   - Algorithm B solves problems of size n by recursively solving two subproblems of size $n - 1$ and then combining the solutions in constant time.

- Algorithm C solves problems of size n by dividing them into nine subproblems of size $n/3$, recursively solving each subproblem, and then combining the solutions in $O(n^2)$ time.

What are the running times of each of these algorithms (in big-O notation), and which would you choose?

4. Solve the following recurrence relations and give a O bound for each of them.

   (a) $T(n) = 2T(n/3) + 1$

   (b) $T(n) = 5T(n/4) + n$

   (c) $T(n) = 7T(n/7) + n$

   (d) $T(n) = 9T(n/3) + n^2$

   (e) $T(n) = T(n-1) + n$

   (f) $T(n) = 2T(n-1) + 1$

5. You are given an array of n elements, and you notice that some of the elements are duplicates; that is, they appear more than once in the array. Show how to remove all duplicates from the array in time $O(nlogn)$. (Try to solve this problem using only $O(1)$ extra space. Using $O(n)$ extra space would be easy.)

6. You are given an infinite array $A[\ .\ ]$ in which the first n cells contain integers in sorted order and the rest of the cells are filled with ∞. You are not given the value of n. Describe an algorithm that takes an integer x as input and finds a position in the array containing x, if such a position exists, in $O(logn)$ time. (If you are disturbed by the fact that the array A has infinite length, assume instead that it is of length n, but that you dont know this length, and that the implementation of the array data type in your programming language returns the error message whenever elements $A[i]$ with $i > n$ are accessed.)

7. Given a sorted array of distinct integers $A[1, \ldots, n]$, you want to find out whether there is an index i for which $A[i] = i$. Give a divide-and-conquer algorithm that runs in time $O(logn)$.

8. The square of a matrix A is its product with itself, AA.

(a) Show that five multiplications are sufficient to compute the square of a 2×2 matrix.

(b) What is wrong with the following algorithm for computing the square of an n n matrix?
Use a divide-and-conquer approach as in Strassens algorithm, except that instead of getting 7 subproblems of size $n/2$, we now get 5 subproblems of size $n/2$ thanks to the first part. Using the same analysis as in Strassens algorithm, we can conclude that the algorithm runs in time $O(n^{log_2 5})$.

(c) In fact, squaring matrices is no easier than matrix multiplication. In this part, you will show that if $n \times n$ matrices can be squared in time $S(n) = O(n^c)$, then any two $n \times n$ matrices can be multiplied in time $O(n^c)$.

i. Given two $n \times n$ matrices A and B, show that the matrix $AB + BA$ can be computed in time $3S(n) + O(n^2)$. (Use $(a + b)^2 = a^2 + b^2 + 2ab$-like identity.)

ii. Given two $n \times n$ matrices$X$and Y, define the $2n \times 2n$ matrices A and B as follows:

$$A = \begin{bmatrix} X & 0 \\ 0 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} Y & 0 \\ 0 & 0 \end{bmatrix}$$

What is $AB + BA$, in terms of X and Y?

iii. Using the above results argue that the product XY can be computed in $3S(2n) + O(n^2)$. Conclude that matrix multiplication takes $O(n^c)$ time.

9. We studied Euclids algorithm for computing the greatest common divisor (gcd) of two positive integers: the largest integer which divides them both. Here we will look at an alternative algorithm based on divide-and-conquer.

(a) Show that the following rule is true ($a \geq b$).

$$gcd(a,b) = \begin{cases} 2gcd(a/2, b/2) & if\ a, b\ are\ even \\ gcd(a, b/2) & if a\ is\ odd,\ b\ is\ even \\ gcd((a-b)/2, b) & if\ a, b\ are\ odd \end{cases}$$

(b) Give an efficient divide-and-conquer algorithm for greatest common divisor.