

CSE548/AMS542 Fall 2008 Analysis of Algorithms

Jie Gao*

September 30, 2008

Due **October 16th** before class. Each problem, unless specified otherwise, has a maximum of 10 points. (i) You write down the solution clearly. If we can not recognize your writing then you may lose points. (ii) Avoid too many details. A succinct and clean proof is the best. You may use the algorithms we covered in class without referring to the details. (iii) If you discuss some of the problems with another fellow student (at most 2 students per group), write down his/her name and the problems. If you consult any books/webpages, cite them.

Homework 3

1. Solve the following recurrences and give a $\Theta()$ bound.

(a) $A(n) = A(\sqrt{n}) + 1$.

(b) $B(n) = B(n - 1) + n^c$, where $c \geq 1$ is a constant.

(c) $C(n) = 49C(n/25) + n^{3/2} \lg n$.

(d) $D(n) = 2D(n - 1) + 1$.

2. **In-place algorithm.** In our median-finding algorithm, a basic primitive is the split operation, which takes as input an array S and a value v and then divides S into three sets: the elements less than v , the elements equal to v , and the elements greater than v . Show how to implement this split operation in place, that is, without allocating new memory.

3. **Square of a matrix.** The square of a matrix A is its product with itself, AA .

(a) Show that 5 multiplications are sufficient to compute the square of a 2×2 matrix. (5pts)

(b) What is wrong with the following algorithm for computing the square of an $n \times n$ matrix? (5pts)

Use a divide-and-conquer approach as in Strassen's algorithm, except that instead of getting 7 subproblems of size $n = 2$, we now get 5 subproblems of size $n = 2$ thanks to part (a). Using the same analysis as in Strassen's 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 $O(n^c)$. (5pts)

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

*Department of Computer Science, Stony Brook University, Stony Brook, NY 11794, USA, jgao@cs.sunysb.edu.

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

$$A = \begin{pmatrix} X & 0 \\ 0 & 0 \end{pmatrix}; B = \begin{pmatrix} 0 & Y \\ 0 & 0 \end{pmatrix}$$

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

- iii. Using (i) and (ii), argue that the product XY can be computed in time $3S(2n) + O(n^2)$. Conclude that matrix multiplication takes time $O(n^c)$.
4. **Greatest common divider.** Compute the greatest common divisor (gcd) of two positive integers: the largest integer which divides them both. Design a divide and conquer algorithm to solve it and analyze the running time.
5. **Foo-sort.** Alice proposed the following sorting algorithm called Foo-sort: If the array has 2 elements in decreasing order then we just swap them. If the array has 3 or more elements, we first sort the first two thirds of the elements of the array recursively with Foo-sort, next sort the last two thirds of the elements of the array recursively, and then sort the first two thirds of the elements of the array recursively again. Notice that the algorithm does not require any externally allocated memory.
- (a) Explain why Foo-sort correctly sorts the input data.
- (b) Analyze the running time of Foo-sort.