

CSE548/AMS542 Spring 2008 Analysis of Algorithms

Jie Gao*

February 1, 2008

Due **Feb 12th** 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 1

1. List the following functions in increasing asymptotic order. Between each adjacent functions in your list, indicate whether they are asymptotically equivalent ($f(n) \in \Theta(g(n))$), you may use the notation that $f(n) \equiv g(n)$ or if one is strictly less than the other ($f(n) \in o(g(n))$) and use the notation that $f(n) \prec g(n)$.

$$\begin{array}{ccccc} 5n^3 + \log n & 2^n & 3^{n/2} & 2^{n/3} & \sqrt{\lg n} \\ \ln n & 2^{\sqrt{\lg n}} & \min\{n^2, 1045n\} & \sum_{i=1}^n i^{77} & n^{\ln 4} \\ \lfloor n^2/45 \rfloor & \lceil n^2/45 \rceil & n^2/45 & \lg \sqrt{n} & \lg \lg n \\ \sum_{i=1}^n 1/i & \sum_{i=1}^n 1/i^2 & \sum_{i=1}^n (i^2 + 5i)/(6i^4 + 7) & \ln(n!) & (\lg n)^{\sqrt{\lg n}} \end{array}$$

2. An in-place algorithm is one that is given an array of length n as input, but it is allowed to use only $O(1)$ additional working storage. In other words, it can use any number of regular variables, but it is not allowed to declare arrays, to dynamically allocate new storage, and it is not allowed to make recursive calls (since doing so would allow it to use the recursion stack for working storage). The in-place restriction is often important in applications where the inputs are very large arrays, that barely fit into memory. For example, heapsort is an in-place algorithm as it can be implemented by using just an array and a few variables to move the numbers around.

Suppose that you are given an array $X[1 :: n]$ of numbers, and you are given an index i , $1 \leq i \leq n - 1$. This implicitly splits X into two subarrays, $X[1 :: i]$ and $X[i + 1 :: n]$. These subarrays are not necessarily of the same size. Give an $O(n)$ time in-place algorithm which, given the array X and i , swaps these two subarrays within X , without changing the order of elements within each subarray. Your algorithm should run in $O(n)$ time.

3. Let G be a directed graph represented using an adjacency list. So, each node $G[i]$ has a list of all nodes reachable in 1 step from i (all out-neighbors of i). Suppose each node of G also has a value: e.g., node 1 might have value \$100, node 2 might have value \$50, etc.

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

Give an algorithm that computes, for every node, the highest value reachable from that node, i.e. that you can get to by some path from that node. For instance, if it is possible to get to any node from any other node (G is “strongly-connected”), then for every node this will be the maximum value in the entire graph. (You also need to prove the correctness of the algorithm.)

Grading: The grade for this problem will fall into one of the below categories:

(n = number of vertices, m = number of edges)

5pts for a correct algorithm

10pts for for $O(m + n \log n)$

5pts extra for $O(m + n)$.

4. Let T be a tree and let T_1, T_2, \dots, T_k be subtrees of T , every two of which have a vertex in common. Show that some vertex of T belongs to all of the subtrees.
5. A node is called a cut node if the removal of which will partition a connected undirected graph into multiple connected components.
 - (a) Give a graph with minimum number of edges on n vertices that does not have a cut node. (3pts)
 - (b) Give an $O(m + n)$ algorithm to find a cut node in a graph, if it exists; and answer NO if otherwise. (You also need to prove the correctness of the algorithm.) (7pts)