
Homework 8 – Due Friday, October 24, 2008

Please refer to the general information handout for the full homework policy and options.

Reminders

- Your solutions are due before the lecture. Late homework will not be accepted.
- Collaboration is permitted, but you must write the solutions *by yourself without assistance*, and be ready to explain them orally to a member of the course staff if asked. You must also identify your collaborators. *Getting solutions from outside sources such as the Web or students not enrolled in the class is strictly forbidden.*
- To facilitate grading, please write down your solution to each problem on a separate sheet of paper. Make sure to include all identifying information and your collaborators on each sheet. Your solutions to different problems will be graded separately, possibly by different people, and returned to you independently of each other.
- For problems that require you to provide an algorithm, you must give a precise description of the algorithm, **together with a proof of correctness** and an analysis of its running time. You may use algorithms from class as subroutines. You may also use any facts that we proved in class or from the book.

Exercises These should not be handed in, but the material they cover may appear on exams:

1. Chapter 6 exercises.
2. Give an algorithm which takes as input a weighted DAG (directed acyclic graph) $G = (V, E)$ with weight function $w : E \rightarrow \mathbb{R}$ together with two nodes $u; v$ and outputs two quantities:
 - (a) The number of distinct paths from u to v ;
 - (b) The number of distinct shortest paths from u to v . Your algorithm should run in time $O(m + n)$.
3. Give a $O(n)$ -time algorithm that takes an array A of n numbers (some negative) and returns a subarray $A[i..j]$ for which the sum of elements is maximal. Next, modify the algorithm so that it finds a subarray with maximal average value.

Problems to be handed in

<p>Page limits: The answer to each problem should fit in 2 pages (or one double-sided sheet) of paper. Longer answers will be penalized.</p>
--

1. Give as efficient an algorithm as you can which takes as input a weighted graph (not necessarily acyclic) $G = (V, E)$ with weight function $w : E \rightarrow \mathbb{R}$ together with two nodes v, w and outputs the number of distinct shortest paths from v to w . Edges in G may have negative weights (lengths), but you may assume that all cycles in G have net weight strictly greater than 0.

2. KT, Chapter 6, problem 8 (EMP).
3. * (Extra credit) Give as efficient an algorithm as you can which takes as input a DAG (directed acyclic graph) $G = (V, E)$ and returns the number of distinct topological sorts of G .