Final exam in
TDT4120 Algorithms og data structures

Exam date	30 November 2009
Exam time	0900–1300
Grading date	21 December
Language	English
Contact during the exam	Magnus Lie Hetland (ph. 91851949)
Aids	No printed/handwritten; specific, simple calculator

Please read the entire exam before you start, plan your time, and prepare any questions for when the teacher comes to the exam room. Make assumptions where necessary. Keep your answers short and concise. Long explanations that do not directly answer the questions are given little or no weight.

You may describe your algorithms in text, pseudocode or program code, as long as it is clear how the algorithm works. Short, abstract explanations can be just as good as extensive pseudocode, as long as they are precise enough. Running times are to be given in asymptotic notation, as precisely as possible.

Problem 1 (56%)

a) What is the difference between a *problem* and a *problem instance*?

Answer (7%):

b) What requirements must normally be met by the input of bucket sort?

Answer (7%):

c) A flow network has an edge from a vertex u to a vertex v, with capacity 7 and flow 3. By how much can you increase the flow from v to u? What will the flow from v to u be then?

Answer (7%):

d) Finding a shortest weighted simple path in a graph with negative cycles is NP-Hard. How does BELLMAN–FORD handle this?

Answer (7%):

e) If you are faced with an unknown problem A in NP, how would you show it to be NP-Complete?

Answer (7%):

f) Is it possible to prove that *some* NP-Complete problems may only be solved in exponential time, without ruling out that some other NP-Complete problems can be solved in polynomial time? Give a short explanation.

Answer (7%):

g) Give a lower and an upper bound for T_P, using *work*, *span* and P, on an ideal parallel machine with greedy scheduling.

Answer (7%):

A k-regular tree is a tree where all internal nodes have k children. It need not be balanced. You may assume that the tree has at least one internal node.

h) How many leaf nodes does a k-regular tree with n internal nodes have? Write a recurrence equation for the problem and solve it. The answer should be exact (that is, without asymptotic notation or unknown constants), as a function of k and n. Show your reasoning.

Answer (7%):

Problem 2 (20%)

Assume you have a set **U** of images and a function $s(\cdot, \cdot)$ describing the similarity between two images as a real number. You may also assume that $s(x, y) = s(y, x)$. Now you want to perform a simple "clustering": You want to split **U** into two subsets **X** and **Y**, such that the sum of the distances from all the objects in **X** to all the objects in **Y** is minimized. Neither **X** nor **Y** is to be empty.

a) Show that the problem can be solved polynomially, *or* show that the problem is NP-hard.

Answer (6%):

You now want to show your images in a slide show, such that two consecutive slides are as similar possible — and for some added dramatic effect you want the first image to also be the last. Thus you want to maximize the sum of the distance between pairs of consecutive images

for the whole slide show, while showing each image exactly once, except for the first/last one, which is shown twice.

b) Give a short description of an algorithm that efficiently solves this problem, with running time, *or* show that the problem is NP-Hard.

Answer (7%):

You want to make a website with your images, where you from each image can navigate to some "neighboring images." If you can navigate directly from a image x to a image y, you also want to be able to navigate from y to x. It must be possible to get from any image to any other, by successive jumps to neighbors, but it should not be possible to navigate in circles. You want to set up this navigational structure such that the sum of the distances over all pairs of neighboring images is maximized.

c) Give a short description of an algorithm that efficiently solves this problem, with running time, *or* show that the problem is NP-Hard.

Answer (7%):

Problem 3 (24%)

Assume that the following holds: For a problem size of n we may process numbers with $O(\log n)$ digits in constant time, and we may use them as indices in tables, with constant look-up time. (This is the default assumption in the textbook.)

You are given an undirected graph with n nodes and m edges, represented by neighbor lists stored in a table indexed by node numbers (standard neighbor list representation). You want to sort all the neighbor lists such that for each node, the neighbor list contains the node's neighbors in ascending order.

a) Explain how you can sort the lists as described, with total running time $O(n + m)$.

Answer (8%):

Assume you have an algorithm A that finds a minimum cost circulation in a directed graph $G = (V, E, w, l, u)$ with vertices V, edges E, weights w, and lower and upper bounds on flow of l and u, respectively, or that informs that no such circulation exists.

b) Show how you can use A to determine is there exists a path from a node u to a node v in an undirected graph $G' = (V', E')$.

Answer (8%):

Let $T = (V, E)$ be an undirected tree, and let $h(u) = \max \{ d(u, v) \mid v \in V \}$, where $d(u, v)$ is the distance from u to v, given by the number of edges in the path from u to v. Define a *midpoint* of T to be a node $u \in V$ such that $h(u) = \min \{ h(v) \mid v \in V \}$.

c) Explain (without any detailed code) how you can find the midpoints of T using a topological sort.

Answer (8%):