Final exam in TDT4120 Algorithms og data structures

Exam date	2 December 2008
Exam time	0900–1300
Grading date	23 December
Language	English
Contact during the exam	Magnus Lie Hetland (tlf. 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.

Problem 1 (47%)

Assume that a problem of size n is to be solved algorithmically.

a) Write examples of the following types of running times as a function of n, expressed in Θ-notation.

Logarithmic (2%) _______________

Linear (2%) _______________

Quadratic (2%) _______________

Polynomial (2%) _______________

Exponential (2%) _______________

b) Why can't you use Θ-notation to describe the general running time of QUICKSORT when it's possible to use this notation to describe both the best-case, average-case and worst-case running times individually? Keep your answer as short as possible.

Answer (8%):

Consider the following algorithm:

for $i = 1 \dots n$
    for $j = i \dots n/100$
        print "Hello, World!"

for $i = 1 \dots n$
    $j = 1 \dots \lg n$
        print "Goodbye, World!"

Note: You can assume that a loop **for** $j = a \dots b$ is not executed if $a > b$.

c) What is the running time of the algorithm, as a function of n, expressed in Θ-notation? Briefly state your reasoning.

Answer (10%):

d) What is the solution to the following recurrence? Give your answer in Θ-notation.

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

Answer (8%):

Consider the following algorithm:

MyAlgorithm(n)

for $i = 1 \dots n$
    print "When will it ever end?"

if $n = 1$
    return True

for $i = 1 \dots 4$
    MyAlgorithm($n/2$)

e) What is the running time of the algorithm, as a function of n, expressed in Θ-notation? Briefly state your reasoning.

Answer (5%):

You are faced with the three problems A, B and C. All three are in the set NP. You know that A is in the set P and that B is in the set NPC. Assume that you are to use polynomial reductions between these problems to show certain properties.

Note: Some of the properties can, of course, be shown in other ways. You may ignore this fact in this problem.

f) Complete the following statements.

To show that C is in P, ____ must be reduced to ____ in polynomial time. (2%)
To show that C is in NPC, ____ must be reduced to ____ in polynomimal time. (2%)
If ____ can be reduced to ____ in polynomial time, it follows that P = NP. (2%)

Problem 2 (26%)

a) Assume that you have a binary heap stored in an array, as described in the textbook. Assume that the root is at index 1. Where (that is, at which position in the array) is the parent node of the element with index *i*?

Answer (6%):

b) How many internal nodes does a binary tree with *n* leaf nodes have, if all the internal nodes have two children?

Answer (7%):

c) What is the difference between a maximum matching and a perfect matching? **Note**: We are talking about bipartite matching here.

Answer (5%):

d) What is a Hamilton cycle?

Answer (8%):

Problem 3 (17%)

a) Describe concicely, in your own words, how RADIX SORT works.

Answer (9%):

In FLOYD-WARSHALL, the expression $d^{(k)}_{ij}$ is used to describe the solution to a subproblem.

b) What is the recursive formula for $d^{(k)}_{ij}$?

Answer (8%):

Problem 4 (10%)

Assume that you have a directed graph with positive integer edge weights. If you are to find the shortest path from *u* to *v* there may be more than one answer; that is, there may be multiple paths with the same (minimal) length.

a) How can you efficiently find the path among the shortest paths from *u* to *v* that consists of the lowest number of edges?

Answer (5%):

b) How can you efficiently determine how many shortest paths (that is, how many paths of minimal length) there are from *u* to *v*?

Answer (5%):