Umeå University
Department of Computing Science
TDBC 91 — Analysis of Algorithms
Examination: October 30, 1998

Name and ID Number:____________________________________

Signature:____

Directions:

1. All answers must be written in English.
2. An English-X/X-English dictionary may be used, with X a natural language of your choice. No other help materials are allowed.
3. Solutions to all problems must be written on the special examination paper, which is provided. Write your name, ID number, and the problem number on each solution sheet. Collate the sheets in numerical order of the problems. Please write on only one side of the paper.
4. The examination has a total of 300 points. The point value of each question is shown next to that question.
5. In the table below, place an X in the position for each part of a problem for which you have attempted a solution, and which you wish to have graded. For parts which you do not mark with an X, you will receive 10% of the points for that part, regardless of your answer or lack thereof. (The column indicates the problem number, and the row the subpart. Boxes marked with a huge dot indicate nonexistent problem parts, and have no other significance.)

	1	2	3	4	5	6	7	8
(a)								
(b)	●							
(c)	●	●		●		●		●
(d)	●	●		●		●	●	●
(e)	●	●	●	●		●	●	●

TDBC91, October 30, 1998, Examination, page 1

1. (30 points) Two distinct notions of asymptotic complexity which recapture the notion that function f is at least as complex as function g were defined in the course notes and lectures. They were represented as $f \in \Omega_1(g)$ and $f \in \Omega_2(g)$. Define each of Ω_1 and Ω_2, and briefly discuss the advantages and disadvantages of each. (It is *not* necessary that you remember which was called Ω_1 and which was called Ω_2. Just define and compare the two notions.) For full credit, your answer must include formal mathematical definitions, as well as English-language explanations of their meaning.

2. (30 points total) Consider a variation on the mergesort algorithm in which the array to be sorted is broken into *three* (instead of two) pieces of equal size, and then each piece is recursively processed using the same technique, with the three pieces then merged together.

(a: 15 points) Write down the recurrence relation which characterizes the time complexity of this algorithm. (You do **not** have to write down the algorithm; just give the recurrence relation which governs it.)

(b: 15 points) Solve this recurrence relation, using whatever techniques are appropriate. You need not explicitly solve for constants, but you must produce the correct form of the solution. You must show intermediate work, including the characteristic polynomial and how you obtained it, as well as any variable transformations which were necessary in the solution.

3. (40 points total)

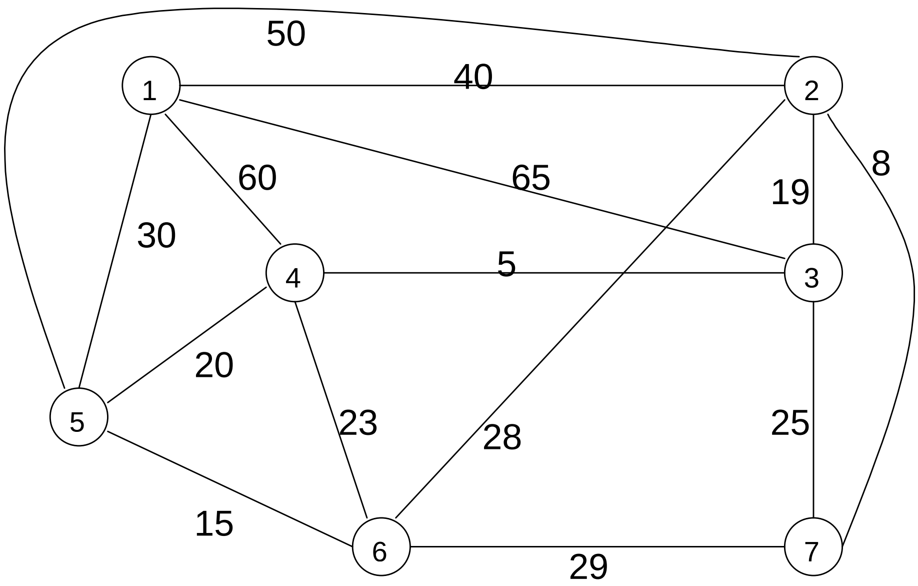
(a: 10 points) Give the asymptotic average-case and worst-case time complexities of quicksort with random pivot selection. You do not need to derive them; just state them.

(b: 10 points) Explain how the asymptotic *worst-case* time complexity of quicksort can be improved by choosing a different algorithm for pivot selection. Identify both the new method of pivot selection and the new worst-case complexity. Also, explain why this strategy is seldom used in practice.

(c: 10 points) In a course experiment, the complexity of a modified quicksort algorithm in which insertion sort was used for small lists was investigated. To what extent would such a modification improve the asymptotic average-case and worst-case complexities identified in part (a)? Provide a clear justification for your answer.

(d: 10 points) Although quicksort is the "preferred" sequential sorting algorithm in practice, mergesort does have its advantages. Identify a significant advantage of mergesort over quicksort, and give a specific example application for which mergesort would be preferred due to that advantage.

4. (40 points total) Depicted below is an undirected graph with weighted edges.



(a: 20 points) Show the sequence of edges which are considered, and which are selected, in applying Kruskal's algorithm for identifying a minimum spanning tree. Thus, you must show the edge considered, and whether or not it is selected, at each step. Do not give just the final solution. Represent each edge in your solution by the numerical labels on the vertices which are its endpoints.

(b: 20 points) Repeat part (a) for Prim's algorithm.

5. (50 points total) Let **P** be an optimization problem in which the optimal solution is a maximum (as opposed to a minimum) value. Assume that **P** is to be solved using the branch-and-bound technique.

(a) (10 points) In the context of general state-space search, define what is meant by the following terms: (i) *E-node*; (ii) *live node*: and (iii) *dead node:*

(b) (10 points) In *branch-and-bound* search, state the rule governing the generation of the children of the E-node.

(c) (10 points) Let $\hat{c}$ be an approximation to the true-cost function c. State how $\hat{c}$ is used to guide the search process. In other words, describe both the data structure which is used to maintain the list of possible next candidates for expansion, and how $\hat{c}$ is used to determine which candidate is selected.

(d) (10 points) Identify a *practical* sufficient conditions on $\hat{c}$ so that the search will always terminate at an optimal leaf node.

(e) (10 points) Using an approximation function $\hat{c}$ as identified in part (d), give the precise conditions under which it is known that an optimal solution has been found.

6. (30 points total)

(a: 10 points) Given a directed graph with weighted edges, define the *travelling salesman* optimization problem.

(b: 20 points) Suppose that the travelling salesman problem is to be solved using the branch-and-bound technique. Describe an effective method for obtaining a good approximate cost function $\hat{c}$ which is based upon the techniques of *full reduction* and *dynamic reduction* of rows and columns which are no longer relevant to the solution. (This technique is discussed in the textbook and the lecture notes.) Illustrate this technique by expanding the the search-space graph one level for the problem defined by the matrix below. In other words, find the associated reduced matrix, as well as the approximate cost $\hat{c}(n)$, for each of the four nodes directly below the root node.

$$\begin{bmatrix} \infty & 11 & 10 & 9 & 6 \\ 8 & \infty & 7 & 3 & 4 \\ 8 & 4 & \infty & 4 & 8 \\ 11 & 10 & 5 & \infty & 5 \\ 6 & 9 & 9 & 5 & \infty \end{bmatrix}$$

7. (30 points total)

(a: 10 points) Give a concise definition of what is meant by a *nondeterministic algorithm*. Particularly, explain how such an algorithm differs from one which is deterministic.

(b: 10 points) Briefly define the classes P and NP.

(c: 10 points) State *Cook's Theorem*, and explain its significance in the quest for efficient algorithms.

8. (50 points total)

(a: 20 points) In the context of an optimization problem, define the following concepts. (i) *absolute approximation algorithm*; (ii) ε-*relative approximation algorithm*; and (iii) *fully polynomial-time* (ε-*relative) approximation algorithm*.

(b: 30 points) Three fully polynomial-time relative approximation strategies were discussed in the course: *rounding*, *interval partitioning*, and *separation*. In the context of the $0/1$-knapsack optimization problem, explain how each of these techniques works. You need not show detailed computations, but your explanation should be precise enough to show clearly the ideas behind these strategies.