Umeå University
Department of Computing Science
TDBC91 — Analysis of Algorithms
Examination: October 31, 2003

Name and ID number: ____

Signature: ____

1. All answers must be written in English.
2. A pocket engineering calculator (miniräknare) and English/X - X/English dictionary may be used. No other help materials are allowed.
3. You must turn in this cover sheet together with the pages containing your answers for Problems 1–7. Collate the sheets in numerical order of the problems, and write on only one side of the paper.
4. The examination has a total of 400 points, which constitutes 40% of the course grade. The other 60% is based upon the obligatory exercises and problem sets, as explained in the course syllabus.
5. You have the option of skipping a problem for partial credit. In the table below, place an X in the position for any problem for which you have attempted a solution, and which you wish to have graded. It is extremely important that you fill in this table properly, because of the following option. For any box which is left blank, the associated question will not be graded, and you will instead be awarded 15% of the points for that question. Your decision to leave a box blank is definitive, so be very careful. For example, If you leave box 5 blank, your answer to that question will not be graded, even if it is completely correct. On the other hand, if you place an X in box 5, but provide no answer whatsoever to that question, you will not receive 15% of the points for that question. It is strongly recommended that you use a pencil, in case you change your mind! Note that it is not possible to opt in or out of parts of problems; only entire problems may be skipped for partial credit.

Prob	1	2	3	4	5	6	7

6. In many of the problem with multiple parts, the answer to later parts depends upon the answer to earlier parts. If the answer to an earlier part is wrong, and if that error makes the answer to a later part much easier or trivial, then full credit will not be awarded for that part even if the answer is correct relative to the previous error.

(1: 60 points total) The behavior of an algorithm is governed by the following recurrence relation, with initial conditions $T(1) = 2$ and $T(2) = 9$.

$$T(n) = 5 \cdot T(n/2) - 4 \cdot T(n/4) + 0.75 \cdot n^2$$

(a) Using a change of variable, transform this into a linear (inhomogeneous) recurrence.

(b) Give the characteristic polynomial for the answer to part (a).

(c) Find the roots of the polynomial obtained in (b).

(d) Give the solution to this linear recurrence. This answer may contain constants to be determined later.

(e) Translate the answer of (d) back to the original problem.

(f) Solve for all constants in the result of (e), thus obtaining a parameter-free, closed form for the solution of the above geometric recurrence.

Note: The comments in item 6 of the cover page are particularly relevant here. Answers to parts of this question which are based upon incorrect answers to earlier parts may not receive much credit, particularly if the earlier answers result in a much simpler problems for the later parts.

(2: 50 points total)

(a) 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) 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) In a software exercise, 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.

(3: 60 points total) Matroids were used to formalize the notion of a greedy solution to a combinatorial optimization problem. Answer the following fundamental questions on this topic. For (a)-(c), you are to give **formal** definitions, not "intuitive" ones.

(a) Define *subset system* and *profit function*.

(b) Define *combinatorial optimization problem*, *feasible solution*, and *optimal solution*, with respect to your answer of (a).

(c) Define *matroid*.

(d) State the fundamental result characterizing matroids in terms of the comparison of two independent sets of differing sizes. (Just state it; do not prove it.)

(e) State the fundamental result characterizing matroids in terms of maximal independent subsets of a set A. (Just state it; do not prove it.)

(4: 60 points total)

(a) Give a concise but clear definition of the abstract data type *priority queue*. The following information must be provided.

 (i) The underlying type on which it operates.

 (ii) Each of its operations, including the domain and range (*i.e.*, input and output types), as well as a clear description of what the operation does.

 Do not describe implementation here in part (a); that issue is addressed in (b) below.

(b) Describe the data structure *min-heap*, including:

 (i) how it is implemented using arrays; and

 (i) how the operations of a priority queue are represented within it.

(c) State (proof is not required) the *worst case* and *average case* time complexity for each of the operations of a priority queue which is implemented as a min-heap as described in (b) above.

(d) In certain applications, such as Prim's algorithm for the minimum spanning-tree problem and Dijkstra's algorithm for single-source shortest path problem, it is necessary to augment the definition of a priority queue to one in which the priority of a given element may be increased (*i.e.*, the value may be decreased). Explain how such an operation is realized in the min-heap model which is described in (b) above. Your explanation of how the algorithm works must be complete and accompanied by at least one non-trivial example.

(e) State (proof is not required) the *worst case* and *average case* time complexity for the operation described in (d) above.

Note: For parts (b) and (d) above, the implementation must be the one presented in the book and course notes, or else an implementation for which all operations have a worst-case and average case time complexity which is no worse than for that of the implementation presented in the course.

(5: 60 points total) Shown below is the incidence matrix for a labelled, directed graph of six vertices.

$$\begin{bmatrix} \infty & 7 & 3 & 12 & 8 & 15 \\ 9 & \infty & 8 & 14 & 22 & 5 \\ 17 & 31 & \infty & 6 & 18 & 12 \\ 9 & 3 & 5 & \infty & 11 & 13 \\ 18 & 14 & 9 & 8 & \infty & 14 \\ 13 & 27 & 19 & 1 & 21 & \infty \end{bmatrix}$$

Suppose that the travelling salesman problem is to be solved for the associated graph using least-cost branch-and-bound, with dynamic reduction, as well as full row and column reduction, used to direct the search, as in Problem Exercise 5.

(a) For the initial matrix given above, give the row reduction, then the column reduction of this row reduction, and finally the lower bound $\hat{c}$ on the cost of a tour which is obtained as a result of these reductions.

(b) The next step in the algorithm is to obtain a reduced cost matrix for each of the paths $1 \to 2$, $1 \to 3$, $1 \to 4$, $1 \to 5$, $1 \to 6$. For this question, compute the reduced cost matrix for the single path $1 \to 5$ only; do not do the computation for the other paths. In your answer, show four values:

  (i) The matrix obtained from that in part (a) by using *dynamic reduction*; *i.e.*, by replacing the appropriate entries with ∞.

  (ii) The matrix obtained by doing a row reduction on the matrix obtained in part (i).

  (iii) The matrix obtained by doing a column reduction on the matrix obtained in part (ii).

  (iv) The value of $\hat{c}$ for this configuration, and how it was obtained.

(c) Repeat part (b), this time for the path $1 \to 5 \to 3$. In other words, show the following four values.

  (i) The matrix obtained from that in part (b) by using *dynamic reduction*.

  (ii) The matrix obtained by doing a row reduction on the matrix obtained in part (i).

  (iii) The matrix obtained by doing a column reduction on the matrix obtained in part (ii).

  (iv) The value of $\hat{c}$ for this configuration, and how it was obtained.

(6: 60 points total) Recall that an *abstract problem* is a pair $P = (I, \rho)$ in which I is a set of instances and $\rho : I \to \{0, 1\}$ is a function which assigns a decision value (1 = *accept* and 0 = *reject*) to each problem instance in I. Continue with this definition the items which follow.

(a) Define what it means for an abstract problem $P = (I, \rho)$ to be a *language problem* over the finite alphabet Σ.

(b) Given two language problems $P_1 = (I_1, \rho_1)$ and $P_2 = (I_2, \rho_2)$ over the same finite alphabet Σ, define what it means for P_1 to be *polynomially reducible* to P_2 (written $P_1 \propto P_2$).

(c) Using the definition given in (b), define what it means for a language problem P to be NP-complete. You may take for known what it means for the a language problem Q to be in NP, but you must explain clearly what the completeness condition means. Your definition must be formal, not just wordy.

(d) Suppose that $P_1 = (I_1, \rho_1)$ and $P_2 = (I_2, \rho_2)$ are language problems, with P_1 NP-complete. Explain clearly what must be shown to establish that P_2 is also NP-complete, in terms of the concepts developed in (a)-(c) above.

(7: 50 points total) Three fully polynomial-time relative approximation strategies were discussed in the course:

(a) *rounding*,

(b) *interval partitioning*, and

(c) *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.