Umeå University
Department of Computing Science
TDBC91 — Analysis of Algorithms
Examination: October 23, 2006

Name (printed): ______________________________

Swedish ID number: ______________________________

Computer User-ID: ______________________________

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. 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. 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.
6. If you want to take a future examination to determine your final grade, and so do not want this examination graded, attach a sheet to your examination clearly stating this intention.
7. This examination consists of seven (7) problems on a total of three (3) pages.

(1: 50 points) Two distinct notions of asymptotic complexity were defined in the course notes and lectures, each recapturing the notion that function f is at least as complex as function g. 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 less formal explanations of their respective meanings. Illustrate each with a simple example.

(2: 60 points total) A family of algorithms is characterized by the following parameterized recurrence relation

$$T(n) = a \cdot T(n/2) + k$$

with $a \geq 1$ and $k > 0$ the parameters.

(a) Find a closed form for the complexity of these algorithms, with a as a parameter.

(b) Identify those values of a for which this complexity is greater than $\Theta(\log(n))$. Explain your answer clearly.

(c) Identify those values of a for which this complexity is greater than $\Theta(n)$. Explain your answer clearly.

Hint: If you are confused, first consider the cases $a = 1$ and $a = 2$, which were solved in the course for well-known problems. Then try a few other values for a and observe the pattern. It may be necessary to break your solution into a few distinct cases.

Recall also *l'Hôpital's rule* from Problem Exercise 6, which may come in handy: If $f(x)$ and $g(x)$ are differentiable functions for which

$$\lim_{x \to \infty} \frac{f(x)}{g(x)}$$

is indeterminate; *i.e.*, either both of

$$\lim_{x \to \infty} f(x) = 0 \text{ and } \lim_{x \to \infty} g(x) = 0$$

hold, or else both of

$$\lim_{x \to \infty} f(x) = \infty \text{ and } \lim_{x \to \infty} g(x) = \infty$$

hold, then

$$\lim_{x \to \infty} \frac{f(x)}{g(x)} = \lim_{x \to \infty} \frac{df(x)/dx}{dg(x)/dx}$$

holds as well.

(3: 60 points total) Answer the following questions. For (a)-(c), you are to give **formal** definitions, not "intuitive" ones. Furthermore, for parts (a) and (b), illustrate the concepts using the discrete knapsack problem.

(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 the *formal greedy algorithm* with respect to a combinatorial optimization problem.

(d) Show that for any integers M and n, with $M \geq 3$ and $n \geq 3$, there is a discrete knapsack problem with exactly and n objects and knapsack capacity M such that the associated formal greedy algorithm does not find the optimal solution. In your proof, assume further that the weights and values of objects are positive integers.

(e) Using the notions of (a)-(c), define the concept of *matroid*, and explain clearly the implication of the result of (d) in this context.

(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) Sketch how a min-heap may be created in time linear in the number of elements. A full proof of the complexity is not required.

(e) A sorting algorithm known as *heapsort* uses a min-heap in a straightforward way. Explain briefly how such a sorting algorithm works, and identify both its average-case and worst-case time complexity. Justify these complexities in light of your answers to part (c) and (d).

Note: For part (b) 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) Consider a discrete knapsack problem P with n objects $\{\mathsf{obj}_1, \mathsf{obj}_2, \ldots, \mathsf{obj}_n\}$, each object obj_i possessing a weight w_i and a value v_i. The capacity of the knapsack is M. Answer the following questions in the context of solving such a problem using the branch-and-bound technique studied in the course.

(In the following, a *partial solution* of degree i with $1 \leq i \leq n$ is a vector $\langle x_1, x_2, \ldots, x_i \rangle \in \{0,1\}^i$ which indicates whether or not each of the first i objects is included or not. A *total solution* is a partial solution of degree n.)

(a) Given a partial solution $\langle x_1, x_2, \ldots, x_i \rangle$ of P, a solution to a knapsack problem Q on $\{\mathsf{obj}_{i+1}, \ldots, \mathsf{obj}_n\}$ is used to obtain a **lower** bound on the best total solution to P which extends $\langle x_1, x_2, \ldots, x_i \rangle$. Describe this problem Q clearly.

(b) Given a partial solution $\langle x_1, x_2, \ldots, x_i \rangle$ of P, a solution to a knapsack problem Q' on $\{\mathsf{obj}_{i+1}, \ldots, \mathsf{obj}_n\}$ is used to obtain an **upper** bound on the best total solution to P which extends $\langle x_1, x_2, \ldots, x_i \rangle$. Describe this problem Q' clearly.

(c) Explain clearly how the solutions to the knapsack problems Q and Q', which are identified in parts (a) and (b) respectively, are used to obtain the solution to the total knapsack problem on $\{\mathsf{obj}_1, \mathsf{obj}_2, \ldots, \mathsf{obj}_n\}$. In answering this part, it is appropriate to provide some details on how the branch-and-bound strategy operates.

Note: For parts (a) and (b) above, the special problems Q and Q' must be those which were presented in the book and course notes, or else ones which provides bounds which are at least as good as those which were studied in the course.

(6: 50 points total) Consider the problem of evaluating a polynomial of degree n

$$\sum_{i=0}^{n} a_i \cdot x^i = a_n \cdot x^n + a_{n-1} \cdot x^{n-1} + a_{n-2} \cdot x^{n-2} \ \ldots \ a_i \cdot x_i \ \ldots \ a_2 \cdot x^2 + a_1 \cdot x + a_0$$

at some value $x = b$.

(a) Give high-level pseudocode for an algorithm which solves this problem in worst-case time $O(n)$.

(b) Illustrate the operation of this algorithm on the polynomial

$$4 \cdot x^3 + 5 \cdot x^2 + 6 \cdot x + 10$$

evaluated at $x = 3$.

(7: 60 points total) Consider the following statement:

> No sorting algorithm can have average-case complexity which is better than $O(n \cdot \log(n))$.

Sketch an argument which shows this to be the case. Make sure that you state the assumptions under which this argument holds.