

Comprehensive Exam in Advanced Algorithms : Fall 2004

Instructions: Choose any four of the following six questions to answer. Clearly mark which question you want graded on the front of the exam by placing an X in the appropriate slot.

Problem	1	2	3	4	5	6
Graded?						

Note that you will be graded not just on the answers, but also on work. An answer alone will not be worth much if anything at all. No books, calculators, notes, talking, etc. Using any communications device at all (cell phones, walkie talkies, etc.) will be an automatic failing grade.

WARNING: If you do not mark the questions you want graded or mark them ambiguously (i.e. if you decide to mark 5 questions rather than 4, etc.), we will grade your lowest 4 questions.

1. Algorithm Complexity: For each of the following pieces of pseudocode, determine the asymptotic size of the variable *sum* in terms of the parameter *n*.

- (a)

```
sum=0;
for (i=1; i<=n; i++)
  for (j=1; j<=i; j++)
    sum++;
```
- (b)

```
sum=0;
for (i=1; i<=n; i++)
  for (j=1; j<=n; j=j*2)
    sum++;
```
- (c)

```
sum=0;
for (i=1; i<=n; i++)
  for (j=1; j<=i*i; j++)
    if (j%i==0)
      for (k=1; k<=j, k++)
        sum++;
```

2. NP-completeness: The Edge Cover Problem (ECP) is defined as follows. Given an undirected graph G with vertex set V and edge set E and a positive integer k , does there exist a subset $E' \subseteq E$ such that $\forall v \in V, \exists e \in E$ such that v is an endpoint of e and $|E'| \leq k$? (Put another way, the ECP is the problem of finding the minimum-size subset E' of the edges such that every vertex must be hit by at least one edge in E' . The decision problem asks whether such a subset exists of size smaller than or equal to k .) You may use the fact that ECP is NP-complete.

The Set Cover Problem (SCP) is defined as follows. Given a sequence of sets $S_1, S_2, S_3, \dots, S_n$ and a positive integer k , does there exist a set $D \subseteq \{1, 2, 3, \dots, n\}$ such that $|D| \leq k$ and

$$\bigcup_{x \in D} S_x = \bigcup_{y \in \{1, 2, 3, \dots, n\}} S_y$$

(Put another way, the set cover problem is the problem of finding the smallest group of sets such that when you union the group together, you get the same union as if you had taken the union of all the sets at once.) Prove that SCP is NP-complete.

3. Greedy: Assume that the frequencies of letters in a given message are as follows. T: 1, S: 2, M: 5, R: 7, E: 9, and A: 25. Design a Huffman code for this message and calculate the length of the resulting encoded message.

4. **Dynamic Programming: 0-1 Knapsack Problem:** You need to load several objects into a knapsack that has a maximum weight capacity M . Given objects x_1, x_2, \dots, x_n where object x_i has weight w_i and profit p_i (if it is placed in the knapsack), determine a subset of objects to place into the knapsack in order to maximize your profit.

Solving the 0-1 Knapsack Problem can be done by making the following observations.

Case 1: The optimal load includes x_n . Then the rest of the load is the optimal load for the knapsack in which only objects x_1, x_2, \dots, x_{n-1} are given and the maximum weight capacity is $M - w_n$.

Case 2: The optimal load does not include x_n . Then the load is the optimal load for the knapsack in which only objects x_1, x_2, \dots, x_{n-1} are given and the maximum weight capacity is M .

These observations imply the following dynamic programming recurrence relation, where $P(i, c)$ denotes the maximum profit for a 0-1 Knapsack Problem using objects x_1, \dots, x_i and a sack with maximum weight capacity c . (Of course, we are ultimately interested in the quantity $P(n, M)$.)

$$P(i, c) = \begin{cases} 0 & \text{if } i = 0 \text{ or } c \leq 0 \\ \max\{P(i-1, c), P(i-1, c-w_i) + p_i\} & \text{otherwise} \end{cases}$$

Solve the 0-1 Knapsack Problem using the above table by completing the table below. Assume that the capacity of your knapsack is $M = 10$.

Object	Weight	Profit
1	3	4
2	5	6
3	5	5
4	1	3
5	4	5

	0	1	2	3	4	5	6	7	8	9	10
0											
1											
2											
3											
4											
5											

5. The Master Method: Assume that we are given two competing algorithms for a particular problem. The recursive relation describing the running time of Algorithm A is as follows.

$$T_A(n) = 9T_A(n/3) + n^2$$

The recursive relation describing the running time of Algorithm B is as follows.

$$T_B(n) = 4T_B(n/4) + n^k$$

Find the largest integer value of k such that Algorithm B is asymptotically preferable to Algorithm A.

The Master Theorem: Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined on the nonnegative integers by the recurrence $T(n) = aT(n/b) + f(n)$ where we interpret n/b to mean either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Then $T(n)$ can be bounded asymptotically as follows. 1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$. 2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \log n)$. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n , then $T(n) = \Theta(f(n))$.

6. Divide and Conquer:

- (a) Write the steps of Quicksort assuming that you use the median-of-5 median as the pivot element. (You do not have to write out the steps of the median-of-5 algorithm. Assume that you are given a function that finds the median for you.)
- (b) Write a recurrence relation for the running time of Quicksort assuming that the median-of-5 is used as the pivot element.
- (c) Analyze the running time of Quicksort using the recurrence relation you found above.