CS_332 (2005-2006)
DESIGNING ALGORITHMS
(*Attempt 2 questions out of 3*)

Question 1

(a) Define the O, Ω and Θ notations.

[4 marks]

(b) The runtime $T(n)$ of many divide-and-conquer algorithms may be described by a recurrence relation of the form

$$T(n) = aT(n/b) + \Theta(n^c),$$

for constants $a \geq 1$, $b > 1$ and $c \geq 0$, where n/b represents either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$.

State a Θ-expression for the function $T(n)$, distinguishing three cases according to the values of a, b and c.

[3 marks]

In the remaining questions we consider the problem of multiplying two n-bit integers X and Y, where we assume that n is a power of 2.

(c) Explain why the usual method for multiplying two n-digit numbers by hand which we learn at school takes $O(n^2)$ (quadratic) time, while the usual method for adding two n-digit numbers by hand takes $O(n)$ (linear) time.

[3 marks]

Suppose we split the two n-bit integers X and Y into two $(\frac{n}{2})$-bit halves as follows:

$$X = \boxed{A|B} \quad \text{and} \quad Y = \boxed{C|D}$$

That is, $X = A \cdot 2^{n/2} + B$ and $Y = C \cdot 2^{n/2} + D$, where A, B, C and D are $(\frac{n}{2})$-bit integers.

(d) Show that $X \cdot Y = A \cdot C \cdot 2^n + \Big((A-B) \cdot (D-C) + A \cdot C + B \cdot D\Big) 2^{n/2} + B \cdot D$.

[3 marks]

(e) With the formula in part (d) in mind, justify the claim that X and Y can be multiplied using $M(n)$ bit operations where

$$M(1) = 1 \quad \text{and} \quad M(n) = 3M(n/2) + \Theta(n) \quad \text{for } n > 1.$$

[6 marks]

(f) Find the asymptotic runtime of the algorithm hinted at in part (e) (that is, find a Θ expression for $M(n)$).

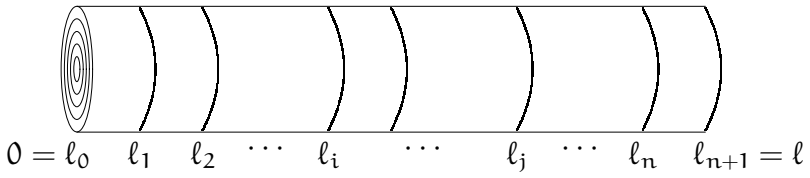
[3 marks]

(g) Why wouldn't we want to use the formula $X \cdot Y = A \cdot C \cdot 2^n + (A \cdot D + B \cdot C) 2^{n/2} + B \cdot D$ in part (d) as the basis of our algorithm for multiplying X and Y?

[3 marks]

Question 2

You bring an ℓ-foot log of wood to your local sawmill. You want it cut at n specific places: $\ell_1, \ell_2, \ldots, \ell_n$ feet from the left end. The sawmill charges £x to cut an x-foot log any place you like.



For example, suppose we wish to cut a log ℓ=12-feet long at lengths ℓ_1=2, ℓ_2=5, and ℓ_3=8 feet from the left end. The first cut (regardless of where it is made) will cost £12, but the cost for the two subsequent cuts will depend on the order in which the cuts are made (as the lengths of the subsequent sublogs to be cut will be different). For example, cutting in the order ℓ_1, ℓ_2, ℓ_3 would cost $12 + 10 + 7 = £29$, while cutting in the order ℓ_2, ℓ_1, ℓ_3 would cost $12 + 5 + 7 = £24$. In this example, it makes sense to start by cutting in the most central spot, to minimize the length of the two remaining pieces.

(a) Consider a greedy algorithm that cuts the log so that the maximum length of the resulting two pieces is always as small as possible; that is, it cuts it in the most central spot. Show that this algorithm does not necessarily achieve the minimal cost, by giving an example in which it fails to do so. (Hint: Consider making three cuts all close to the midpoint.)

[3 marks]

(b) Argue why Dynamic Programming is appropriate for this problem.

[4 marks]

Let $c[i, j]$ (for $0 \leq i < j \leq n{+}1$) be the optimal (i.e., least) cost of completely cutting the sublog whose left endpoint is at ℓ_i and whose right endpoint is at ℓ_j. We thus wish to compute $c[0, n{+}1]$.

(c) What is $c[i, i{+}1]$? Explain.

[2 marks]

(d) What is $c[i, i{+}2]$? Explain.

[2 marks]

(e) Give a recursive definition for $c[i, j]$.

[4 marks]

(f) Give pseudocode for a dynamic programming algorithm which computes $c[0, n{+}1]$.
(Note: You do *not* need to compute the optimal order itself, just the optimal cost.)

[6 marks]

(g) Analyze the run time and space requirement of this algorithm.

[4 marks]

Question 3

(a) Explain the phrase "information-theoretic lower bound" as it applies to comparison-based algorithms.

[3 marks]

(b) Consider the problem of merging a sorted list of m elements with a sorted list of k elements to produce a sorted list of $(m+k)$ elements, using only pairwise comparisons.

Show that the information-theoretic lower bound on the number of comparisons performed in the worst case is $\left\lceil \lg \binom{m+k}{k} \right\rceil$.

$\Big[$ *Recall that* $\binom{n}{c}$ *("*n choose c*") denotes the number of ways of* choosing c *items from a collection of* n *items (where* $c \leq n$*), and is given by* $\frac{n!}{c!(n-c)!}$. $\Big]$

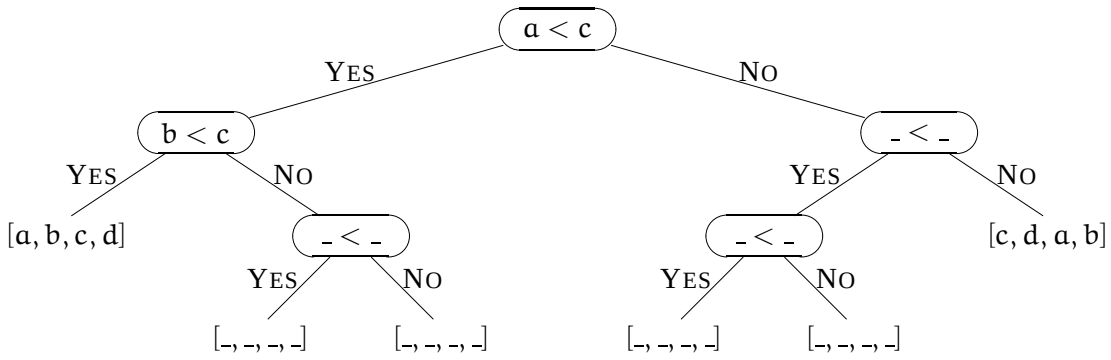
(Hint: The merged list has $m+k$ elements, k of which are taken from the second list.)

[4 marks]

(c) Briefly describe an algorithm which achieves the information-theoretic lower bound derived in part (b) for the special case $k=1$ (i.e., when inserting a single element into a sorted list).

[3 marks]

(d) The following is a skeleton of the comparison tree of an algorithm for merging the two sorted lists $[a, b]$ and $[c, d]$ using at most $\left\lceil \lg \binom{4}{2} \right\rceil = 3$ comparisons.



Draw this comparison tree in full, filling in the missing entries.

[4 marks]

(Please turn over.)

(e) Explain the phrase "adversary strategy" as it applies to comparison-based algorithms.

[3 marks]

(f) Demonstrate, using an adversary strategy, that any comparison-based algorithm for merging two sorted lists of n elements must use at least $2n-1$ comparisons in the worst case.

(Hint: Suppose that merging the two lists $[a_1, a_2, \ldots, a_n]$ and $[b_1, b_2, \ldots, b_n]$ results in the list $[a_1, b_1, a_2, b_2, \ldots, a_n, b_n]$.)

[5 marks]

(g) Given that $\binom{10}{5} = 252$, demonstrate that there is no algorithm for merging two sorted lists of length $m{=}k{=}5$ which achieves the information-theoretic lower bound derived in part (b).

[3 marks]