PRIFYSGOL CYMRU; UNIVERSITY OF WALES

DEGREE EXAMINATIONS JANUARY 2003

SWANSEA

Computer Science

CS 332 Designing Algorithms

Attempt 2 questions out of 3

Time allowed: 2 hours

Students are permitted to use the dictionaries provided by the University

Students are NOT permitted to use calculators

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

Question 1

(a) Give asymptotic bounds, expressed as $T(n) = \Theta(\cdot)$, for each of the following recurrences. In each case, explain how you get your solution, for example, by naming the particular case of the (Simplified) Master Theorem.

(i) $T(n) = T(n-2) + 1$.

[2 marks]

(ii) $T(n) = 2T(\lfloor n/2 \rfloor) + 3n$.

[2 marks]

(iii) $T(n) = 3T(n-2)$.

[2 marks]

(iv) $T(n) = 2T(\lceil 3n/4 \rceil) + n^2$.

[2 marks]

(v) $T(n) = \sqrt{8}T(\lfloor n/2 \rfloor) + n\sqrt{n}$.

[2 marks]

(b) Let A be an array of n ***distinct*** integers sorted in increasing order.

Describe a $O(\lg n)$ divide-and-conquer algorithm which either finds an index i such that $A[i] = i$, or else determines that no such i exists.

[Hint: If $A[i] > i$, can we have $A[i+1] = i+1$? Or $A[i+2] = i+2$?]

Justify the correctness and running time of your algorithm.

[5 marks]

(c) You are given a collection of n bolts of different widths, and n corresponding nuts. Each bolt matches precisely one nut. You are allowed to try to match a nut with a bolt, from which you can determine whether the nut is too large, too small, or an exact match. But there is no way to compare two nuts, nor to compare two bolts.

(i) Describe an efficient algorithm for finding the smallest pair of matching nut and bolt. How many comparisons does your algorithm perform in the *worst* case before finding the smallest pair ?

[5 marks]

(ii) Describe an efficient algorithm based on QUICKSORT which matches up all pairs of nuts and bolts.

[5 marks]

Question 2

(a) Dynamic Programming is appropriate for problems which satisfy the ***optimal substructure property*** and the ***overlapping subproblems property***. Define these two properties.

[4 marks]

(b) The year is 1670, and you are a fur trader for the newly-founded ***Hudson's Bay Company***. There are n trading posts along the Churchill River, numbered 1 through n with trading post $i+1$ downstream from trading post i (for $1 \leq i < n$). At any of these posts you may rent a canoe which may be returned at any other trading post downstream. (It is impossible to paddle against the current and travel upstream.) For each possible departure point i and each possible arrival point j ($i < j$), you know the cost $c[i, j]$ of renting a canoe. However, different rental companies operate at different trading posts, so it may be more expensive to travel from i to j by renting a canoe at i and returning it at j than to rent a canoe at i, return it at some intermediate trading post, and rent another canoe there (and possibly return that at some further intermediate post and rent yet another one there, and so on). For example, $c[1, 3]$ may be greater than $c[1, 2] + c[2, 3]$. You wish to travel all the way from trading post 1 to trading post n with the least cost.

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

[3 marks]

Let $m[i, j]$ (for $1 \leq i \leq j \leq n$) be the least cost of travelling from trading post i to trading post j. This problem thus comes down to computing $m[1, n]$.

(ii) What is the value of $m[i, i]$? Explain.

[1 marks]

(iii) What is the value of $m[i, i+1]$? Explain.

[2 marks]

(iv) What is the value of $m[i, i+2]$? Explain.

[3 marks]

(v) Write a recursive definition for $m[i, j]$.

[5 marks]

(vi) Write pseudo-code for a dynamic programming algorithm which computes the optimal cost of travelling from post 1 to post n.

[5 marks]

(vii) Analyse the running time and space requirement of your algorithm.

[2 marks]

Question 3

(a) The ***best-case*** complexity of a comparison-based sorting algorithm is the ***minimum*** number of comparisons performed by the algorithm when sorting n elements; its ***worst-case*** complexity is the ***maximum*** number of comparisons performed by the algorithm when sorting n elements.

(i) Argue why any comparison-based sorting algorithm must have a best-case complexity of at least $(n-1)$ comparisons.

[3 marks]

(ii) Argue why any comparison-based sorting algorithm must have a worst-case complexity of at least $\lceil \lg(n!) \rceil$ comparisons.

[6 marks]

(iii) Describe a <u>*single*</u> sorting algorithm which has best-case complexity $(n-1)$ and worst-case complexity $O(n \lg n)$.

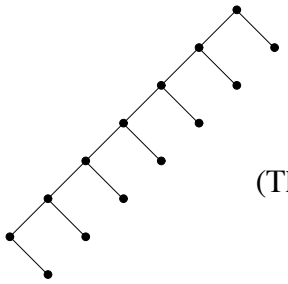
[3 marks]

(iv) By using an information theoretic lower bound argument based on comparison trees, show that there is no algorithm for sorting 5 elements which simultaneously achieves best-case complexity 4 and worst-case complexity 7.

[6 marks]

(b) This question concerns the Ford-Johnson sorting algorithm, as applied to a 14-element list.

(i) Show that $13 \times 11 \times 9 \times 7 \times 5 \times 3 \times 1 = 135\,135$ is the number of permutations of 14 elements which are consistent with the following partial order:



(The 'comb' of length 7.)

[Hint: First express P_n in terms of P_{n-1}, where P_n denotes the number of permutations of $2n$ elements which are consistent with a 'comb' of length n.]

[4 marks]

(ii) Deduce from (e) above that if Steps 1 and 2 of the Ford-Johnson algorithm are applied to a 14 element set, then step 3 is an optimal way to complete the sort.

[Hint: $2^{17} < 135\,135 < 2^{18}$.]

[3 marks]