The University of Melbourne

Department of Computer Science and Software Engineering

433–521

Algorithms and Complexity

Semester 1, 2009

Identical Examination papers: None.

Common Content: None.

Exam Duration: Three hours.

Reading Time: Fifteen minutes.

Length: This paper has 4 pages including this cover page.

Authorised Materials: None.

Instructions to Invigilators:

Instructions to Students: Answer all of the questions in the script book provided. Suggested times for each question are given (these are indicative of the number of marks).

Calculators: Calculators are not permitted.

Baillieu Library: Exam paper to be held by the library.

Question 1 [15 minutes].

This question is about sorting algorithms.

(a) Name four sorting algorithms mentioned in this subject.

(b) Worst case time complexity is one important criterion which may influence which sorting algorithm we choose. Name three others.

(c) Give a four by four table which show how each of the criteria (including worst case time complexity) apply to each of the algorithms.

Question 2 [5 minutes].

Your friend shows you two algorithms for managing a database of people's names. They try to convince you that one algorithm will be faster in practice than the other by testing each of the algorithms on sets of of twenty, forty, eighty, and one hundred and sixty randomly-generated strings. Describe at least two objections that you could make to your friend's attempt to convince you.

Question 3 [20 minutes].

For each of the following expressions, write the order of growth, using Θ notation, in the simplest form. For example, $3n^2 + \sqrt{n} \in \Theta(n^2)$.

(a) $\sum_{i=1}^{n}(2i+1)$

(b) $\sum_{i=0}^{n}(2^i - 1)$

(c) $\sum_{i=1}^{n}(2\log i + 1)$

(d) $\sum_{i=1}^{n-1}(n-(i+1))$

(e) $\sum_{i=1}^{n-1}(\sum_{j=1}^{i}(j+1)))$

Question 4 [20 minutes].

Consider the following recursive algorithm to find the smallest element in an array:

```
If the array has just one element then return it
Divide the array into two halves, as equally as possible
Find the smallest element in each half, recursively
Compare the two returned elements and return the smaller one
```

(a) Write a recurrence relation for this algorithm, in terms of the number of comparisons $C(n)$, where n is the number of elements.

(b) Solve the recurrence relation.

(c) How does this method compare with the usual loop-based method of finding the smallest element, in terms of number of comparisons?

Question 5 [50 minutes].

This question is about an abstract data type for sets of integers. The representation used will be sorted lists without duplicates. To help answer the questions, the following (pseudo)coding of merge sort for lists may be useful. It is assumed `head(l)` and `tail(l)` return the head (first element) and tail (remaining elements) of a list, respectively, and a procedure is available for splitting a list into two lists of approximately equal size.

```
// returns sorted version of list l
mergesort(l):
    if l is empty or tail(l) is empty then return l
    split l into lists l1 and l2
    s1 = mergesort(l1)
    s2 = mergesort(l2)
    return merge(s2, s2)

// returns merge of sorted lists s1 and s2
merge(s1, s2):
    if s1 is empty then return s2
    if s2 is empty then return s1
    if head(s1) < head(s2) then
        h = head(s1)
        s3 = merge(tail(s1), s2)
        add h to front of s3
        return s3
    else
        h = head(s2)
        s3 = merge(s1, tail(s2))
        add h to front of s3
        return s3
```

(a) Describe an algorithm (preferably using pseudo-code) to determine if a number n is an element of a set s (a sorted list without duplicates) and give the worst case time complexity.

(b) Describe an algorithm to convert an arbitrary list of integers (not necessarilty sorted or free of duplicates) to a set (sorted and without duplicates) and give the worst case time complexity.

(c) Describe an algorithm to compute the *union* of two sets and give the worst case time complexity (the union of two sets is the set of elements which are in either of the two sets).

(d) Describe an algorithm to compute the *intersection* of two sets and give the worst case time complexity (the intersection of two sets is the set of elements which are in both of the two sets).

(e) Briefly argue why this representation of sets is better than a representation which uses arbitrary lists.

Question 6 [30 minutes].

You are asked to find the *median* (middle value) of an array of numbers, which we will assume are all distinct. For example, if there are 27 elements, then you must return the 14th smallest (which is also the 14th largest). The strategy used to solve this problem is somewhat like the quicksort algorithm, but has some important differences.

Consider choosing a pivot element and partitioning the input so that the elements less than the pivot are to its left, and those larger than the pivot to its right.

(a) Approximately how many comparisons are required to perform this partitioning?

(b) How can we efficiently determine which partition contains the median?

(c) Having answered part (b), describe a recursive algorithm to find the median.

(d) Now assume that you can guarantee that the pivot partitions the input into one part that has at most 3/4 of the elements, and one part that has at least 1/4 of the elements. Write a recurrence relation for the number of comparisons in this algorithm, $C(n)$, assuming the worst partition, subject to the guarantee.

(e) Solve this recurrence relation, assuming $C(1) = 0$. What do you conclude about the running time required to find the median element?

(f) How can choosing the pivot element uniformly at random help you to obtain the guarantee mentioned in part (d)?