CS 3364 - Summer I/2004 - Test 2

June 18, 2004

Name:

1 Searching

Assume you have a data set with one million elements. Comparing two entries is very slow, so slow that every other operation doesn't matter. You are to write a function to search for a specific entry in the data set.
Which algorithm would you use if you know you have less than 40 lookups before the data set changes? Why? (approx. 2 sentences, 1pt)

Which algorithm would you use if you know you have more than 40 lookups? Why? (approx. 2 sentences, 1pt)

2 Sorting

2.1 Lower Bound

What is the lower bound of operations for a comparison-based sort? Is this the lower bound of operations for any sort? If so, why? If not so, What principle can be used to sort even faster? (approx. 5 sentences, 2pt)

Which of the sorting algorithms named in this test guarantee a worst-case performance which is within this lower bound? (just names, 1pt)

2.2 Insertion

As you might have found out while doing the homework, there are different ways of doing the search part on the insertion sort. The major difference between the book and the class notes were that the algorithm from the book searched from back to front, while in class we search from front to back.
Given an already sorted list, which variant is more efficient? Why? (1-2 senteces, 1 pt)

2.3 Binary Insertion

We know that binary insertion has less data comparisons than regular insertion sort. But what about data read-writes? Are the different? If yes, which one has less? Are they about the same? Are they exactly the same? Why? (1-2 sentences, 1 pt)

If you use the regular insertion sort algorithm from the book (searching backwards on the insertion) and you have an already sorted list as input, how will binary insertion perform compared to regular insertion? Why? (1-2 sentences, 1 pt)

2.4 Bubble

Please construct a simple worst-case for bubble-sort (the most comparisons, the most exchanges). Either describe or give an example (1 sentence or 10 elements, 1 pt).

"Nowadays computers are so fast that the performance of an algorithm doesn't really matter anymore, and bubble sort is so simple, so I am using that." What do you, as a good computer scientist have to say about that? (2-3 sentences, 1 pt).

2.5 Selection

In which respect is selection sort the best of the discussed sort algorithms (1-2 sentences, 1pt)?

2.6 Merge

Consider the following attempt to implement mergesort:

```
mergesort (min,max):

  mid = (min+max/2)
  mergelist(min,med,max)
  mergesort(min,med)
  mergesort(med+1,max)
```

Two things are terribly wrong. Which? (2 sentences or corrections, 1 pt)

Why is the in-place variant of the merge-sort much better than the one we had last week? (1-2 sentences, 1 pt)

2.7 Quick

We have talked about optimizations of quicksort. Why are they needed? Give an example of a worst-case for the unoptimized quick sort (3-4 sentences, 2 pt)

2.8 Heap

Draw a binary heap-tree that contains the elements 8 4 7 3 2 1 6 5 9. (1 pt)

Which trick is used to store this binary heap-tree in an array? Give the array representation of the tree you have drawn. (1-2 sentences+1 graph, 1pt)

2.9 Shell

Shell sorts performance depends very much on the sequence used. As a matter of fact, the sequence originally proposed (powers of two: 1,2,4,8,16,32) gives a very bad performance. Why? (about 2 sentences, 1 pt).