CISC 181 Lab 4 Algorithms

Complete and submit this document.

1. (20 marks) In this week's notes, I presented the exchange sort algorithm and used it to sort a collection of books by the surnames of their authors. The time complexity of that algorithm, as measured by the number of comparisons it makes in its inner loop, is given as

where *N* is the number of books (i.e., the number of items to sort). Thus, to sort 10 books, exchange sort makes

comparisons.

A naïve person, such as one who had not read the notes for this unit, might suspect that the job of sorting 20 books would require twice as many comparisons as are needed for 10 books, and would therefore take about twice as much time, but this is not the case.

Assume you can compare the authors of two books and swap their positions if needed at an average rate of one operation per second. Calculate the number of comparisons that would be required and the time, in minutes and seconds, that it would take to sort book collections of the sizes given in the table below.

|  |  |  |
| --- | --- | --- |
| Number of books (*n*) | Number of comparisons | Time in minutes and seconds (mm:ss) |
| 10 | 45 | 00:45 |
| 11 | 55 | 00:55 |
| 12 | 66 | 01:06 |
| 13 | 78 | 01:18 |
| 14 | 91 | 01:31 |
| 15 | 105 | 01:45 |
| 16 | 120 | 02:00 |
| 17 | 136 | 02:16 |
| 18 | 153 | 02:33 |
| 19 | 171 | 02:51 |

Now, let’s say a computer that can make 1000000 (one million) comparison operations per second, on average, is using exchange sort to sort 1000000 (one million) items of data. How many days, hours, minutes, and seconds will the computer require to do that? Express your answer in "d hh:mm:ss" form, as in "2 13:04:18" for 2 days, 13 hours, 4 minutes, and 18 seconds.

5 18:53:20

1. (20 marks) Merge sort, described in lecture, and similar sorts have time complexities that are proportional to

*n* log2(*n*)

(That's *n* times the log2 of *n*.) log2 is described in this week's notes on binary search, as binary search has a time complexity proportional to log2(*n*). Scientific calculators frequently have a way of calculating log2 of a number, and so does Microsoft Excel. I encourage you to use Excel or some compatible spreadsheet program (Google Sheets, perhaps) to help you with the calculations for this problem.

In the unit's notes, you may have seen that I figured out a formula for the maximum number of comparisons a merge sort might need to make:

where the ⌈ and ⌉ brackets imply the ceiling function, rounding values with fractional parts like 1.25 up to the next highest integer, 2. Translated into an Excel formula, with a value representing *n* stored in cell A1, that is

=ceiling.math(a1\*log(a1,2))-a1+1

Let's repeat the exercise from the previous page, but this time imagining that we are using merge sort instead of exchange sort. For this first part, it will still be you sorting books, and you are still capable of one comparison per second on average, but now you'll be applying my formula, above, instead of (*n*2 – *n*)/2 for your middle column calculations. Again, I've done the first one for you.

|  |  |  |
| --- | --- | --- |
| Number of items (*n*) | Number of comparisons | Time in minutes and seconds (mm:ss) |
| 10 | 25 | 00:25 |
| 11 | 29 | 00:29 |
| 12 | 33 | 00:33 |
| 13 | 36 | 00:36 |
| 14 | 41 | 00:41 |
| 15 | 45 | 00:45 |
| 16 | 49 | 00:49 |
| 17 | 54 | 00:54 |
| 18 | 59 | 00:59 |
| 19 | 63 | 00:63 |

Now, let’s say a computer that can make 1000000 (one million) comparison operations per second, on average, is using this more efficient sorting algorithm to sort 1000000 (one million) items of data. How many days, hours, minutes, and seconds (d hh:mm:ss) will the computer require to do that?

0 00:00:19