THE UNIVERSITY OF AUCKLAND
NEW ZEALAND
Tamaki Campus

COMPSCI.220.S1.T - Algorithms and Data Structures

ASSIGNMENT 1 – INTRODUCTION TO ALGORITHM ANALYSIS

Out: Thursday 6th March, 2008
Due: Wednesday 2nd April, 2008
Worth: 8.33% of the final course grade

This assignment is worth **50 marks** representing **8.33%** of your total course grade.

Model answers with brief instructions for marking

Requirements

Answers to the questions must be explained in detail: 25% of a total mark for each step of solution could be deducted if detailed comments are absent.

1. (*2 marks*) Work out the time complexity $T(n)$ of the following piece of code in terms of the number of operations for a given integer n:

```
for ( int i = 1; i < n*n*n; i *= n ) {
     for ( int j = 0; j < n; j += 2 ) {
          for ( int k = 1; k < n; k *= 3 ) {
               // constant number C of operations
          }
     }
}
```

   - (*1 out of 2 marks*) For a given n, the inner loop variable successively takes the following values: $k = 1 \equiv 3^0$, 3, 3^2, ..., 3^{m-1} such that $3^{m-1} < n \leq 3^m$, or $m-1 < \log_3 n \geq m$. Therefore, the inner loop performs $C \log_3 n$ operations for each pair of values of the variables j and i.
   - (*0.5 out of 2 marks*) The middle loop variable j takes $n/2$ values, and the outer loop variable i takes three values, 1, n, and n^2 for a given n.
   - (*0.5 out of 2 marks*) The time complexity of the whole piece of code is equal to $T(n) = 3C\frac{n}{2}\log_3 n = 1.5Cn\log_3 n$.

2. (*3 marks*) You have found empirically that the implemented `insertionSort` and `heapSort` methods spent 5 μs and 1 μs, respectively, to sort an array of 1000 objects. Find out how much time each algorithm will spend for sorting an array of $100,000,000$ objects?

   - (*1 mark out of 3*) The worst and average time complexity values of `insertionSort` and `heapSort` are $O(n^2)$ and $O(n \log n)$, respectively. Therefore, for a large n, the time spent by these implementations can be written as $T_{\text{is}}(n) = c_{\text{is}} n^2$ and $T_{\text{hs}}(n) = c_{\text{hs}} n \log_{10} n$.
   - (*1 out of 3 marks*) The scaling factors are respectively $c_{\text{is}} = \frac{T_{\text{is}}(1000)}{1000^2} = 5 \times 10^{-6}$ μsec per object and $c_{\text{hs}} = \frac{T_{\text{is}}(1000)}{1000 \cdot 3} = \frac{1}{3000}$ μsec per object.
   - (*1 out of 3 marks*) Therefore, the processing time is as follows: $T_{\text{is}}(10^8) = \frac{5}{10^6}10^{16} = 5 \times 10^{10}$ μsec and $T_{\text{hs}}(10^8) = \frac{1}{3000}8 \times 10^8 = \frac{8}{3} \times 10^5$ μsec, respectively.

3. (*4 marks*) Suggest which of the two software packages, **A** and **B**, of the same price should be bought to maintain large data bases having each up to 10^9 records. It is known that the average time to process n records with **A** and **B** is $T_{\mathbf{A}}(n) = 0.1n\log_2 n$ milliseconds and $T_{\mathbf{B}}(n) = 2.5n$ milliseconds, respectively. Decide which package is better in the "Big-Oh" sense, work out exact conditions, in terms of the database size n, when this package outperforms the other, and recommend the best choice in your case.

- (*1 mark out of 4*) In the "Big-Oh" sense, the package $\mathbf{B}$ is better (due to its linear vs. $n \log n$ time complexity).
- (*2 marks out of 4*) The package $\mathbf{B}$ will begin to outperform the package $\mathbf{A}$ when $T_{\mathbf{B}}(n) \leq T_{\mathbf{A}}(n)$, that is, when $2.5n \leq 0.1n \log_2 n$, or $\log_2 n \geq 25$, or $n \geq 2^{25}$.
- (*1 mark out of 4*) Therefore, to maintain the databases of size $10^9 \approx 2^{30}$, the package $\mathbf{B}$ is still the best choice.

4. (*5 marks*) Prove that $T(n) = 5n \log_2 n + 500n$ is $\Omega(n)$ and $O(n^{1+\varepsilon})$ where $\varepsilon > 0$ is an arbitrary small positive constant.

(a) (*1.5 marks out of 5*) $T(n)$ is $\Omega(n)$ because $\log_2 n \geq 1$ for all $n \geq 2$ and therefore $T(n) = 5n \log_2 n + 500n = (500 + 5 \log_2 n) \cdot n \geq 505n$ for all $n \geq 2$.

(b) (*3 marks out of 5*) By the Limit Rule, $T(n)$ is $O(n^{1+\varepsilon})$ if $\lim_{n\to\infty} \frac{T(n)}{n^{1+\varepsilon}} = \lim_{n\to\infty} \frac{5n \log_2 n + 500n}{n^{1+\varepsilon}} \equiv \lim_{n\to\infty} \frac{5 \log_2 n + 500)}{n^{\varepsilon}} = 0$. For the first term, this condition holds for $n \to \infty$ because by the L'Hopital's rule of calculus, $\lim_{n\to\infty} \frac{5 \log_2 n}{n^{\varepsilon}} = 0$.
The following proof of the latter statement can be but is not expected in the student's solution: for $x \to \infty$:

$$\lim_{x\to\infty} \frac{5 \log_2 x}{x^{\varepsilon}} = \lim_{x\to\infty} \frac{\frac{d}{dx} 5 \log_2 x}{\frac{d}{dx} x^{\varepsilon}} = \lim_{x\to\infty} \frac{5x^{-1} \log_2 e}{\varepsilon x^{\varepsilon - 1}} = \lim_{x\to\infty} \frac{5 \log_2 e}{\varepsilon x^{\varepsilon}} = 0$$

where $\mathrm{e} = 2.71828\ldots$ is the base of the natural logarithms.
The solution remains valid if instead of the Limit Rule, it directly analyses the Big-Oh condition: $T(n) < cn^{1+\varepsilon}$ for $n > n_0$ and finds that such constant and threshold exist using the L'Hopital's rule.

(c) (*0.5 marks out of 5*) The second term $\frac{500}{n^{\varepsilon}}$ tends to zero for $n \to \infty$, too.

5. (*5 marks*) Let the processing time for two algorithms, $\mathbf{A}$ and $\mathbf{B}$, comply to the conditions $T_{\mathbf{A}}(n)$ is $\Omega(n)$ and $T_{\mathbf{B}}(n)$ is $O(n)$, respectively. Decide whether both the algorithms can be of the same time complexity (i.e. $T_{\mathbf{A}}(n) = T_{\mathbf{B}}(n)$) and, if yes, determine under which conditions this can happen.

Both algorithms can be of the same time complexity if $T_{\mathbf{A}}(n)$ is simultaneously $O(n)$, i.e. is $\Theta(n)$, and $T_{\mathbf{B}}(n)$ is simultaneously $\Omega(n)$, i.e. is $\Theta(n)$, too.

6. (*6 marks*) Let the processing time $T(n)$ of a certain optimisation algorithm depend on the problem size n as follows: $T(n) = 0.01n^2 \log_2 n + 6n (\log_2 n)^2$. Determine the "Big-Theta" time complexity bounds for this algorithm, decide whether the statements "$T(n)$ is $O(n^3)$" and "$T(n)$ is $\Omega(n^2)$ hold for this algorithm, and prove your decisions.

(a) (*2 marks out of 6*) Because $n > log_2 n$ and $6n\,(\log_2 n)^2 > 0$ for all $n \geq 2$, $0.01n^2 \log_2 n < T(n) < 6.01n^2 \log_2 n$ for $n \geq 2$, so that $T(n)$ is $\Theta(n^2 \log n)$.

(b) (*2 marks out of 6*) The statement $T(n)$ is $O(n^3)$ holds because $n > \log_2 n$ and thus $T(n) < 6.01n^3$ for all $n \geq 2$.

(b) (*2 marks out of 6*) The statement $T(n)$ is $\Omega(n^2)$ holds because $\log_2 n \geq 1$ and thus $T(n) > 0.01n^2$ for all $n \geq 2$.

7. (*6 marks*) You have theoretically derived that the processing time $T(n)$ of a certain algorithm is both $\Omega(n)$ and $O(n^3)$. Decide whether you can conclude that $T(n)$ is $\Theta(n^2)$.

No, to conclude that $T(n)$ is $\Theta(n^2)$ the processing time has to be simultaneously $\Omega(n^2)$ and $O(n^2)$. The derived bounds are insufficient for such a conclusion: actually $T(n)$ can be anywhere within the range between $\Theta(n)$ to $\Theta(n^3)$ but not necessarily $\Theta(n^2)$.

8. (*7 marks*) Determine a recurrence relation for the time $T(n)$ of processing a string of the length n with the Java method:

```
public static String reverse( String s ) {
   int n = s.length();
   if ( n <= 1 ) return s;
   else return reverse(s.substring(n/2, n)) + reverse(s.substring(0, n/2));
}
```

assuming that the Java method `String substring(int i,int j)` spends $j-i$ time units for copying a substring of elements with indices from i to $j-1$ from the string `s`, and derive a closed-form formula for $T(n)$ by solving this recurrence relation with the base condition is $T(1) = 0$ under the simplifying assumption that $n = 2^m$ with the integer m.

(a) (*2 marks out of 7*) Assuming that the `if`-statement takes c processing steps, the recurrent relation for the time is as follows: $T(n) = 2T(n/2) + c$ where $T(n/2)$ is the time for the first and the second recurrent call, respectively.

(a) (*0.5 marks out of 7*) For $n = 2^m$, the recurrence is reduced to $T(2^m) = 2T(2^{m-1} + c$.

(c) (*3.5 marks out of 7*) Telescoping the latter recurrence gives:

$$\begin{array}{rcl} T(2^m) & = & 2T(2^{m-1} + c \\ 2T(2^{m-1}) & = & 2^2T(2^{m-2} + 2c \\ \ldots & \ldots & \ldots \\ 2^{m-1}T(2) & = & 2^mT(1) + 2^{m-1}c \end{array}$$

(d) (1 mark out of 7) The above telescoping yields the following solution: $T(2^m) = c(1 + 2 + ... + 2^{m-1}) = c(2^m - 1)$, or $T(n) \approx cn$.

If the sum $1+2+\ldots+2^{m-1}$ in parentheses is not reduced to 2^m-1, but the solution indicates clearly that $T(n)$ is linear by n, the last step should receive its full mark.

9. (*6 marks*) Assuming $n = 5^m$ with the integer $m = \log_5 n$, derive a closed-form formula for $T(n)$ by solving the recurrence relation $T(n) = 5T\,(n/5) + 5$ with the base condition $T(1) = 0$.

(a) (*0.5 marks out of 6*) For $n = 5^m$, the recurrence is reduced to $T(5^m) = 5T(5^{m-1}) + 5$.

(b) (*3.5 marks out of 6*) Telescoping the latter recurrence gives:

$$\begin{array}{rcl} T(5^m) & = & 5T(5^{m-1} + 5 \\ 5T(5^{m-1}) & = & 5^2T(5^{m-2} + 5^2 \\ \ldots & \ldots & \ldots \\ 5^{m-1}T(5) & = & 5^mT(1) + 5^m \end{array}$$

(c) (2 marks out of 6) The above telescoping yields the following closed-form formula $T(5^m) = 5^mT(1) + 5(1 + 5 + 5^2 + \ldots + 5^{m-1})$, that is, $T(5^m) = 5\frac{5^m-1}{5-1} = \frac{5}{4}(5^m - 1)$, or $T(n) \approx 1.25n$.

If the sum $1 + x + \ldots + x^{p-1}$ in parentheses is not reduced to $\frac{x^p-1}{x-1}$, but the solution indicates clearly that $T(n)$ is linear by n, the last step should receive its full mark.

10. (*6 marks*) You need to select $k = 15$ most successful lottery winners from an unordered array of winning records for $n = 100,000,000$ web lottery participants all over the world. Each record contains the winner's UPI and the winnings specifying how successful this winner is. You know (possibly, from the course COMPSCI 220) two options for selecting the k higher-rank individuals:

    (a) to run k times `quickselect` with linear processing time, $T_{\text{qselect}}(n) = cn$, in order to sequentially fetch the participants having the desired ranks $n, n-1, \ldots, n-k+1$), or

    (b) to run once `quicksort` with $n \log n$ processing time, $T_{\text{qsort}} = cn \log_2(n)$, to sort the array in ascending order of winnings and fetch the k higher-rank participants.

    Providing the factors c are the same for both the algorithms and the fetching time is negligibly small comparing to the sorting time, which option does result in the fastest selection?

- (*3.5 marks*) The option (a) is better if $kT_{\text{qselect}} \leq T_{\text{qsort}}$, that is, $ckn \leq cn \log_2 n$, or $k < \log_2 n$.
- (*2.5 marks*) Because in this particular case $k = 15 < \log_2 10^8 \approx 26.7$, the option (a) ensures the fastest selection.

Submission

The due date is Wednesday, 2nd April 2008, 8:30 p.m. (ADB time) [the penalty 20% till 3rd April, 8:30 p.m., and 50% till 4th April, 8:30 p.m.; no submission afterwards].

Submit your assignment to the Assignment Dropbox (ADB) system. You have to electronically hand in a pdf-file `Assignment1.pdf` with your answers to Questions 1–10. You may use any platform and text editor to prepare your submission, but you should check at the Faculty of Science Tamaki Computer Labs that your final pdf-file is readable. In particular, you should use only most common fonts like Times New Roman or Arial. Scanned hand-written notes embedded as images into a pdf file **will not be accepted** for marking.

Marking scheme

	% of marks
Correctness of your answers to Questions 1–10	up to 40
Correctness of intermediate steps in deriving the answers	up to 30
Detailed explanations of these steps with the reference, if necessary, to the textbook	up to 30
	100