Australian National University
Research School of Computer Science

COMP3600/COMP6466 in 2016 – Quiz Two

Due: 23:55pm Friday, August 12

Submit your work electronically through Wattle. The total mark of this quiz worths 10 points which is worth 3.5 of the final mark.

Question 1 (2 points).

(a) In the following definitions of optimal algorithms for a problem, which one is correct, assuming that the problem size is n and its lower bound is $\Omega(n^3 \lg^2 n)$.

1. The difference between the lower bound and the upper bound of its running time is a constant factor, i.e., its lower bound is $\Omega(n^3 \lg^2 n)$, while its upper bound is $O(n^3 \lg^2 n)$
2. The lower bound of its running time is $\Omega(n^2 \lg \lg n)$
3. The upper bound of its running time is $O(n^3 \lg \lg \lg n)$

(b) In the following comparison-based sorting algorithms, (1) which ones are the optimal algorithms and which ones are not? (2) Express the running time of the optimal algorithm, using asymptotically lower and upper bounds notations, assuming that there are n elements to be sorted.

- insertion sort
- merge sort
- quick sort
- shell sort

Question 2 (2 points).

(a) Given a problem $\mathcal{A}$ with problem size n, John shows that its lower bound is $2^{10} n \lg^2 n$, while Mary proves that its lower bound is $n^{1.1} \lg n$ for $\mathcal{A}$. Whose solution is better in terms of the quality of the solutions? Justify your answer.

(b) Given a problem $\mathcal{A}$ with problem size n, student X devised an algorithm for the problem with time complexity $O(n^{3/2} \lg n)$, student Y proposed an algorithm for the problem with time complexity of $O(n \lg^5 n)$, and student Z devised an algorithm for it

with time complexity of $O(n^2 \lg\lg n)$. Order these three algorithms in terms of their upper bounds on the running time from the highest to the lowest? Justify your answer.

Question 3 (3 points).

Provide the best lower and upper bounds on the running time of Algorithm 1 andAlgorithm 2, where mod is the modulo operator that takes constant time.

Algorithm 1 Iter(v_1,v_2)

1: $n \leftarrow v_1.size()$;
2: $m \leftarrow v_2.size()$;
3: **if** $m \leq n$ **then return** error
4: **for** $i \in \{1, \ldots, n\}$ **do**
5: **for** $j \in \{i+1, \ldots, m\}$ **do**
6: $w[i] \leftarrow w[i] + v_1[i] * v_2[j]$
7: **return** w

Algorithm 2 Rec1(x,y,n)

1: **if** $n = 1$ **then return** $x \times y$
2: $m \leftarrow \lceil \frac{n}{2} \rceil$;
3: $a \leftarrow \lfloor \frac{x}{2^m} \rfloor$;
4: $b \leftarrow x \mod 2^m$;
5: $c \leftarrow \lfloor \frac{y}{2^m} \rfloor$;
6: $d \leftarrow y \mod 2^m$;
7: $e \leftarrow \text{Rec1}(a, c, m)$;
8: $f \leftarrow \text{Rec1}(b, d, m)$;
9: $g \leftarrow \text{Rec1}(b, c, m)$;
10: $e \leftarrow \text{Rec1}(a, d, m)$;
11: **return** $2^{2m}e + 2^m(g + h) + f$.

Question 4 (3 points).

In the linear-time selection algorithm, the n elements in the input sequence are divided into groups of size 5 except the last group. Does the algorithm still run in time $O(n)$ if the input elements are divided into groups of size c instead? Formulate the time complexity of the algorithm as a recurrence and solve the recurrence, where $c > 0$ is an odd integer, e.g., $c = 39$.