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 $O(n^3 \lg \lg \lg n)$

Answer: An algorithm for a problem is optimal if its worst running time (in the big-O sense) matches a lower bound of the problem, the only difference between the upper bound and the lower bound is a constant coefficient. Thus, it is (1).

(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

Answer: (1) The optimal algorithm is the merge sort algorithm, the rest are not. The lower and upper bounds on its running time are $\Omega(n \lg n)$ and $O(n \lg n)$ respectively. Thus, its running time is tight, i.e., $\Theta(n \lg n)$.

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.

Answer: Mary's solution is better than John's solution in terms of the quality of solutions, as the estimated accuracy of Mary's solution is more accurate (to the exact solution – the tight bound) than that of John's solution, as

$$\lim_{n\to\infty} \frac{n^{1.1} \lg n}{2^{10} n \lg^2 n} = \lim_{n\to\infty} \frac{n^{0.1}}{\lg n} = \infty$$

(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.

Answer: Following the time complexity analysis of the three algorithms, the upper bounds of their running times from the highest to the lowest are: Z's algorithm, X's algorithm, and Y's algorithm, as the estimate (over estimation to the exact running time) of Y's solution is more accurate (to the tight bound) than those of X and Z.

Question 3 (3 points).

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

Algorithm 1 Iter(v_1,v_2)

```
1: n ← v1.size();
2: m ← v2.size();
3: if m ≤ n then return error
4: for i ∈ {1, ..., n} do
5:     for j ∈ {i + 1, ..., m} do
6:         w[i] ← w[i] + v1[i] * v2[j]
7: return w
```

Answer: Let $T(n, m)$ be the time complexity of algorithm 1, then

$$T(n,m) = c_1 + \sum_{i=1}^{n} \sum_{j=i+1}^{m} c_2 = c_1 + c_2 \sum_{i=1}^{n} (m-i) \tag{1}$$

$$= c_1 + c_2 \left(nm - \sum_{i=1}^{n} i \right) \tag{2}$$

$$= c_1 + c_2 \left(nm - \frac{n(n+1)}{2} \right), \tag{3}$$

where $c_1 > 0$ and $c_2 > 0$ are constants. Since $m > n$, this implies that $m \geq n+1$ and $\frac{n+1}{2m} \leq \frac{1}{2}$.
Since

$$\lim_{n\to\infty} \frac{nm - \frac{n(n+1)}{2}}{nm} = \lim_{n\to\infty} 1 - \frac{n+1}{2m} = c^*,\ 0 < c^* < \infty,$$

we have

$$T(n,m) = \Theta(nm).$$

using the theorem presented in Lecture 3 slides 3 and 4.

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: $h \leftarrow \text{Rec1}(a, d, m)$;
11: **return** $2^{2m}e + 2^m(g + h) + f$.

Answer: Let $T(n)$ be the time complexity of algorithm 2, then

$$T(n) = c + 4T\left(\frac{n}{2}\right).$$

The running time of an intermediate level i in the recursion tree equals

$$c4^i,\ i = 0, 1, \ldots, \lg n - 1.$$

The total running time of all intermediate levels gives:

$$c\sum_{i=0}^{\lg n-1} 4^i = c\left(\frac{4^{\lg n}-1}{3}\right) = c\left(\frac{n^{\lg 4}-1}{3}\right) = c\left(\frac{n^2-1}{3}\right) = \Theta(n^2).$$

Note that the first equation is based on the compact representation of geometric sums (Lecture 4, slide 12).

The last level of the recursion tree has $4^{\lg n} = n^{\lg 4} = n^2$ leafs. We thus have

$$T(n) = \Theta(n^2).$$

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., $c = 39$.

Answer: Let R be the set of n elements. Recall that we need the find the median of a median sequence x, and we partition the n elements into three disjoint subsets using the value of x, i.e., $R_1 = \{x' \mid x' < x, x' \in R\}$, $R_2 = \{y \mid y = x, y \in R\}$, and $R_3 = \{x'' \mid x'' > x, x'' \in R\}$, $R_i \cap R_j = \emptyset$ if $i \neq j$ for $1 \leq i, j \leq 3$, and $R_1 \cup R_2 \cup R_3 = R$. We assume that the ith smallest element that we are looking for is in R_3 (if it is in R_1, we can proceed similarly), we now show that the size of R_1 is bounded. Otherwise (i.e., the element is in R_1 or R_2), we can show that the size of R_3 is bounded as well.

$$|R_1| \geq \frac{c+1}{2}(\frac{\lceil n/c \rceil}{2} - 2) \geq \frac{(c+1)n}{4c} - (c+1).$$

Thus, the size of R_3 is

$$|R_3| = n - |R_1| - |R_2| \leq n - |R_1| \leq \frac{(3c-1)n}{4c} + (c+1).$$

The time complexity is thus represented as the following recurrence.

$$T(n) \leq \begin{cases} \Theta(1) & \text{if } n \leq 128 \\ T(\lceil n/c \rceil) + T(\frac{(3c-1)n}{4c} + (c+1)) + O(n) & \text{if } n > 128 \end{cases}$$

We can prove by the method of substitution that for any rational number $n > 0$, $T(n) \leq bn$ for some constant $b > 0$. Choose b large enough that $T(n) \leq bn$ for all $n \leq 128$.

Also suppose the $O(n)$ term in the recurrence is bounded by an for $n > n_0$.

For $n > n_0$, the recurrence says

$$\begin{aligned} T(n) &\leq T(\lceil n/c \rceil) + T(\frac{(3c-1)n}{4c} + (c+1)) + an && (4)\\ &\leq b\lceil n/c \rceil + b\frac{(3c-1)n}{4c} + b(c+1) + an && (5)\\ &\leq bn/c + b + 3bn/4 - \frac{bn}{4c} + b(c+1) + an && (6)\\ &= \frac{3bn + 3bcn}{4c} + 2b + bc + an && (7)\\ &= bn + (\frac{3b}{4c}n - bn/4 + 2b + bc + an) && (8)\\ & && (9) \end{aligned}$$

The expression $\frac{3b}{4c}n - bn/4 + 2b + bc + an$ is negative if $n > n_0$ and $b \geq \frac{4acn}{cn - 3n - 8c - 4c^2} = \frac{4a}{1 - 3/c - 1/16 - c/32}$ when $n_0 = 128$. Therefore, for some b, $T(n) \leq bn$ always.