

COMP3600/COMP6466 in 2009 – Assignment One Answers to Assignment One

No programming is needed for this assignment. Put your work in the COMP3600 box on the ground floor of the CSIT Building. The total marks for this assignment are 50 points for COMP3600 and 60 points for COMP6466.

Question 1 (5 points).

Given the following sequence, order them into a sorted sequence in the order of growth when n approaches infinity.

$$n^{3.5}, n \log \log n, n \log^3 n, n^{3/2} \log n, 2^{\ln n}, 3^{0.3n}, n^5, 2^{1/n}, 3^{100} \log n.$$

The order of growth from small to larger is as follows.

$$2^{1/n}, 3^{100} \log n, 2^{\ln n}, n \log \log n, n \log^3 n, n^{3/2} \log n, n^{3.5}, n^5, 3^{0.3n}.$$

Question 2 (12 points).

Let $f(n)$, $g(n)$ and $h(n)$ be three positive function. For each of the following statements, either prove that the statement is true or give an example for f , g , h showing that the statement is false.

(a) $f(n) = O(g(n))$ and $g(n) = o(h(n))$ together imply that $f(n) = o(h(n))$;

$f(n) = O(g(n))$ implies:

there exist constants $c_0 > 0$ and $n_0 > 0$ such that for any $n \geq n_0$, the following inequalities hold

$$0 \leq f(n) \leq c_0 g(n), \tag{1}$$

while $g(n) = o(h(n))$ implies that there exists a constant $n_1 > 0$ for any constant $c_1 > 0$ such that for any $n \geq n_1$, the following inequalities hold

$$0 \leq g(n) < c_1 h(n), \quad (2)$$

Combine inequalities (1) and (2), there exists a constant $n_2 = \max\{n_0, n_1\} > 0$ for any constant $c_2 = c_0 c_1 > 0$ such that for any $n \geq n_2$, the following inequalities hold

$$0 \leq f(n) \leq c_0 g(n) \leq c_0 c_1 h(n) = c_2 h(n), \quad (3)$$

i.e.,

$$f(n) = o(h(n)).$$

The claim is true.

$$(b) \min\{f(n), g(n), h(n)\} = O(f(n) + g(n) + h(n));$$

$$\min\{f(n), g(n), h(n)\} \leq f(n), \quad (4)$$

$$\min\{f(n), g(n), h(n)\} \leq g(n), \quad (5)$$

$$\min\{f(n), g(n), h(n)\} \leq h(n), \quad (6)$$

From Inequalities (4), (5), and (6), we have

$$\min\{f(n), g(n), h(n)\} \leq (f(n) + g(n) + h(n))/3. \quad (7)$$

In other words, there are constants $c = 1/3$ and $n_0 = 1$ such that for any $n \geq n_0$, the following inequalities hold.

$$0 \leq \min\{f(n), g(n), h(n)\} \leq c(f(n) + g(n) + h(n)), \quad (8)$$

So, $\min\{f(n), g(n), h(n)\} = O(f(n) + g(n) + h(n))$.

The claim is true.

(c) $f(n) = \Omega(g(n))$ and $g(n) = O(h(n))$ together imply that $f(n) = \Theta(h(n))$.

This claim is wrong, for example, $f(n) = n^2$, $g(n) = n$, $h(n) = n^{1.5}$, clearly, $f(n) = \Omega(g(n))$ and $g(n) = O(h(n))$ but $f(n) \neq O(h(n))$.

Question 3 (8 points for COMP3600, 12 points for COMP6466).

Give an expression for each of the following sums using the $\Theta()$ notation. In each case explain your reasoning clearly.

(a) $\sum_{k=1}^n k^{5/3}$.

$$\sum_{k=1}^n k^{5/3} < \sum_{k=1}^n n^{5/3} = n^{8/3}, \text{ thus, } \sum_{k=1}^n k^{5/3} = O(n^{8/3}).$$

$$\begin{aligned} \text{Also, } \sum_{k=1}^n k^{5/3} &\geq \sum_{k=\lceil n/2 \rceil}^n k^{5/3} \geq \sum_{k=\lceil n/2 \rceil}^n (n/2)^{5/3} \\ &= (n - \lceil n/2 \rceil + 1) \left(\frac{n}{2}\right)^{5/3} \geq \left(\frac{n}{2}\right)^{8/3}, \text{ i.e., } \sum_{k=1}^n k^{5/3} = \Omega(n^{8/3}). \text{ Thus,} \end{aligned}$$

$$\sum_{k=1}^n k^{5/3} = \Theta(n^{8/3}).$$

(b) $\sum_{k=1}^n k^4/3^k$.

If the ratio between adjacent terms is always greater than a constant greater than 1, or always less than a constant less than 1, the series has geometric type and the sum is $\Theta(\text{largest term})$. In this case,

$$\frac{\text{term } k+1}{\text{term } k} = \frac{(k+1)^4/3^{k+1}}{k^4/3^k} = \frac{1}{2} \left(\frac{k+1}{k}\right)^3 \leq \frac{1296}{1875},$$

when $k \geq 5$. so the sum is $\Theta(\text{largest term}) = \Theta(1)$.

(c) (COMP6466 only) $\sum_{k=1}^n \log^4 k/k$.

You might care to know that $\int (\log x)^4/x dx = C + \frac{1}{4}(\log x)^5$.

$$\sum_{k=1}^n \log^4 k/k \leq \int_1^n (\log x)^4/x dx = \frac{1}{4}(\log n)^5;$$

$$\sum_{k=1}^n \log^4 k/k \geq \sum_{k=2}^n \log^4 k/k = \frac{1}{4}(\log n)^5,$$

$$\text{thus, } \sum_{k=1}^n \log^4 k/k = \Theta(\log^5 n).$$

Question 4 (12/50).

Give an asymptotic upper bound for $T(n)$ using the $O(\cdot)$ notation in each of the following recurrences. Justify your answers. Don't use *the Master Theorem*.

(a) $T(n) = 5T(n/3) + n^2$

We apply the iteration method (apply the recurrence to itself repeatedly). It is obvious that $k = \log_3 n$ when $\frac{n}{3^k} = 1$. Then,

$$\begin{aligned} T(n) &= 5T(n/3) + n^2 \\ &= T(n/3^2) + n^2 + 5(n/3)^2 \\ &= \dots \\ &= T(1) + n^2 + 5(n/3)^2 + 5^2(n/3^2)^2 + \dots + 5^k(n/3^k)^2. \end{aligned}$$

This is a series of geometric type so its sum is $\Theta(n^2)$. The term $T(1)$ is negligible in comparison, so the answer is $T(n) = \Theta(n^2)$.

(b) $T(n) = T(n/4) + n \log n$

We use mathematical induction, starting with an educated guess that the answer might be $T(n) = \Theta(n \log n)$. We start the induction at $n = 4$.

Assume that for some constant $c > 1$, and all $n \geq 4$, we have

$$T(n') \leq cn' \log n'$$

for $4 \leq n' < n$. (This is the induction hypothesis.) Then, applying the recurrence, we have

$$\begin{aligned} T(n) &= T(n/4) + n \log n \\ &\leq c(n/4) \log(n/4) + n \log n \\ &\leq \frac{cn}{4} \log n + n \log n \\ &= \frac{c+4}{4} n \log n \end{aligned}$$

For $n \geq 4$, we have $\frac{c+4}{4} \leq c$ provided c is large enough, for example when $c \geq 4/3$. Therefore, by induction, $T(n) \leq cn \log n$ for $n \geq 4$.

We also have $T(n) \geq n \log n$ obviously (look at the recurrence), so the answer is $T(n) = \Theta(n \log n)$.

(c) $T(n) = T(n/5) + T(2n/3) + n$

We use mathematical induction, starting with an educated guess that the answer might be $T(n) = \Theta(n)$. We start the induction at $n = 15$.

Assume that for some constant $c > 1$, and all $n \geq 15$, we have

$$T(n') \leq cn'$$

for $15 \leq n' < n$. (This is the induction hypothesis.) Then, applying the recurrence, we have

$$\begin{aligned} T(n) &= T(n/5) + T(2n/3) + n \\ &\leq c(n/5) + c(2n/3) + n \\ &= \left(\frac{13c}{15} + 1\right)n \\ &= \frac{13c + 15}{15}n \end{aligned}$$

For $n \geq 15$, we have $\frac{13c+15}{15} \leq c$ provided c is large enough, for example when $c \geq 15/2 = 7.5$. Therefore, by induction, $T(n) \leq cn$ for $n \geq 15$.

We also have $T(n) \geq n$ obviously (look at the recurrence), so the answer is $T(n) = \Theta(n)$.

Question 5 (13 points).

You are going on a long trip. You start on the road at mile post 0. Along the way there are n hotels, at mile posts $a_1 < a_2 < \dots < a_n$, where each a_i is measured from the starting point. The only places you are allowed to stop are at these hotels, but you can choose which of the hotels you stop at. You must stop at the final hotel (at distance a_n), which is your destination.

You would ideally like to travel 200 miles a day, but this may not be possible (depending on the spacing of the hotels). If you travel x miles during a day, the *penalty* for that day is $(200 - x)^2$. You want to plan your trip so as to minimize the total penalty – that is, the sum, over all travel days, of the daily penalties.

Describe an efficient algorithm that determines the optimal sequence of hotels at which to stop, and analyse the running time of your algorithm. *Hint: use dynamic programming.*

Let a_1, a_2, \dots, a_n be the given sequence. The key property is: for any j , if you stop at hotel a_j , then the minimum penalty so far is the penalties from the previous hotel a_i you stopped plus the penalty applied to the distance from a_i to a_j , $0 < i < j$.

This gives a recurrence. Define $P(j)$ to be the penalty so far when stopping at hotel a_j , for $1 \leq j \leq n$. Then, for $1 \leq i \leq n$,

$$P(j) = \min_{0 < i < j} \{P(i) + (200 - (a_j - a_i))^2\}.$$

Use the recurrence to compute $P(1), P(2), \dots, P(n)$, in that order. Then $P(n)$ is the answer to the problem. Clearly the computation of all $P(j)$ s takes $O(n^2)$ time.

To find the detailed tour of hotels stayed, we use another table $I()$ to remember the index, i.e., for $1 \leq j \leq n$,

$$P(j) = \min_{0 < i < j < n} \{P(i) + (200 - (a_j - a_i))^2\}.$$

$$I(j) = i,$$

if $P(j) = P(i) + (200 - (a_j - a_i))^2$.

Question 6 (6 points for COMP6466 only).

A subsequence is *palindromic* if it is the same whether read from left to right or right to left. For instance, the sequence

$A, C, G, T, G, T, C, A, A, A, A, T, C, G$

has many palindromic subsequences, including A, C, G, C, A and A, A, A, A (on the other hand, the subsequence A, C, T is not palindromic).

Devise an algorithm that takes a sequence $x[1 \dots n]$ and return the (length of the) longest palindromic subsequence. Its running time should be $O(n^2)$.

Let $X = x_1, x_2, \dots, x_n$ be the given sequence, and let $Y = x_n, x_{n-1}, \dots, x_2, x_1$ be another sequence, which is the reverse sequence of X . Then, we have an important observation, that is,

the longest palindromic subsequence in X is exactly **the longest common subsequence** (LCS) between X and Y , which can be verified from the definition of palindromic sequences. Thus, the original problem is reduced to find a longest common subsequence between X and Y . The solution can be found from the textbook. For the sake of convenience, in the following we reproduce the solution.

Let $X_i = x_1, x_2, \dots, x_i$ and $Y = x_n, x_{n-1}, \dots, x_{j+1}, x_j$ be the subsequences of X and Y respectively. Let $c(i, j)$ be the length of an LCS between X_i and Y_j , $0 \leq i, j \leq n$. Then,

$$c(i, j) = \begin{cases} 0, & i = 0, j = 0 \\ c(i - 1, j - 1) + 1, & x_i = y_j \\ \max\{c(i - 1, j), c(i, j - 1)\}, & x_i \neq y_j \end{cases}$$

Thus, $c(n, n)$ is the longest palindromic subsequence of X . The calculation of $c(n, n)$ takes $O(n^2)$ because there are n^2 cells in 2-D array c to be filled, while the calculation of each cell takes $O(1)$ time. To find the longest palindromic subsequence, you need another 2-D array to keep track of the choice you made during the calculation of each cell.