CSCI-2725 Homework 1

Yitao Tian

2023-01-21

1. **Problem 1:** Give the Big O estimate of the following functions.

   (a) As defined, $f \in O(n^5)$. Given a function $f : \mathbb{N} \to \mathbb{N}$, where

$$\begin{aligned} f(n) &= 5n^5 - 4n^4 + 3n^3 + \sqrt{2}n^2 + n - 1 \\ \text{(which is in turn)} &\leq 5n^5 + 3n^5 + \sqrt{2}n^5 + 1n^5 \\ \text{(which is in turn)} &= (5 + 3 + \sqrt{2} + 1)n^5 \\ \therefore f &\in O(n^5). \end{aligned}$$

   (b) As defined, $f \in O(n^5)$. Given a function f, where

$$\begin{aligned} f(n) &= (n^3 - (\log(n))^3)(n^2 - \log(n)) + 11n^3 \\ &= n^3n^2 - n^3\log(n) - (\log(n))^3n^2 + (\log(n))^4 + 11n^3 \\ &= n^5 - n^3\log(n) - n^2(\log(n))^3 + (\log(n))^4 + 11n^3 \\ &\leq n^5 + (\log(n))^4 + 11n^3 \\ &\leq 1n^5 + 1n^5 + 11n^5 \\ &= (1 + 1 + 11)n^5 \\ \therefore f &\in O(n^5). \end{aligned}$$

(c) As defined, $f \in O(n^2)$. Given a function f, where

$$\begin{aligned}
f(n) &= \frac{5n^4 + 10n^3 - 100n^2 - n - 1}{6n^2} \\
&= \frac{5}{6}n^2 + \frac{10}{6}n - \frac{100}{6} - \frac{1}{6n} - \frac{1}{6n^2} \\
&\leq \frac{5}{6}n^2 + \frac{10}{6}n \\
&\leq \frac{5}{6}n^2 + \frac{10}{6}n^2 \\
&= \left(\frac{5}{6} + \frac{10}{6}\right)n^2 \\
\therefore\ f &\in O(n^2).
\end{aligned}$$

(d) As defined, $f \in O(n^2 \log(n))$. Given a function f, where

$$\begin{aligned}
f(n) &= \log(n^3 + n + 10) + n^2 \log(n + 4) \\
&\leq \log(1n^3 + 1n^3 + 10n^3) + n^2 \log(1n + 4n) \\
&= n^2 \log(5n) + \log(12n^3) \\
&= n^2(\log(5) + \log(n)) + \log(12) + \log(n^3) \\
&= \log(5)n^2 + n^2 \log(n) + \log(12) + 3\log(n) \\
&\leq (1 + \log(5) + \log(12) + 3)n^2 \log(n) \\
\therefore\ f &\in O(n^2 \log(n)).
\end{aligned}$$

(e) As defined, $f \in O(n^2 \log(n)^2)$. Given a function f, where

$$\begin{aligned}
f(n) &= (n\log(n) + 1)^2 + (\log(n) + 1)(n^2 + 1) \\
&= n^2(\log(n))^2 + 2n\log(n) + n^2\log(n) + \log(n) + n^2 + 2 \\
&\leq (1 + 2 + 1 + 1 + 1 + 1 + 1)n^2 \log(n)^2 \\
\therefore\ f &\in O(n^2 \log(n)^2).
\end{aligned}$$

(f) As defined, $f \in O(\log(n))$. Given a function f, where

$$\begin{aligned}
f(n) &= \log((5n^5 + 7n^3 + 10)^2(3n^3 + 4n + 10)) \\
&= \log((25n^{10} + 70n^8 + 100n^5 + 49n^6 + 140n^3 + 100) \\
&\qquad \times (3n^3 + 4n + 10)) \\
&= \log(75n^{13} + 310n^{11} + 250n^{10} + 427n^9 + 1000n^8 \\
&\qquad + 196n^7 + 1310n^6 + 1000n^5 \\
&\qquad + 560n^4 + 1700n^3 + 400n + 10000) \\
&\leq \log((75 + 310 + 250 + 427 + 1000 + 196 + 1310 \\
&\qquad + 1000 + 560 + 1700 + 400 + 10000)n^{13}) \\
&= \log(75 + 310 + 250 + 427 + 1000 + 196 + 1310 \\
&\qquad + 1000 + 560 + 1700 + 400 + 10000) + 13\log(n) \\
&\leq (\log(75 + 310 + 250 + 427 + 1000 + 196 + 1310 \\
&\qquad + 1000 + 560 + 1700 + 400 + 10000) + 13)\log(n)
\end{aligned}$$

$\therefore f \in O(\log(n))$.

2. **Problem 2:** Find constants c and n_0 such that the given statement is true.

(a) Show, for a function f, where $f(n) = n^2 + 10$, $f \in O(n^3)$:

$$\begin{aligned}
\because f(n) &= n^2 + 10 \\
&\leq 1n^2 + 10n^2, n_0 = 1 \\
&= 11n^2
\end{aligned}$$

$\therefore$ if $c = 11 \wedge n_0 = 1$, then $f \in O(n^3)$. ■

(b) Show, for a function f, where $f(n) = n!$, $f \in O(n^n)$:

$$\begin{aligned}
\because f(n) &= n! \triangleq \prod_{i=1}^{n} i \\
&= \underbrace{(n)(n-1)\cdots(2)(1)}_{n \text{ factors}} \\
&\leq \underbrace{(n)(n)\cdots(n)(n)}_{n \text{ factors}}, n_0 = 1 \\
&= 1^n
\end{aligned}$$

$\therefore$ if $c = 1 \wedge n_0 = 1$, then $f \in O(n^n)$. ■

(c) Show, for a function f, where $f(n) = 3n^3 + 7n + 10$, $f \in \Theta(n^3)$:

$$\begin{aligned}
\because\ f(n) &= 3n^3 + 7n + 10 \\
&\leq 3n^3 + 7n^3 + 10n^3, n_0 = 1 \\
&= 20n^3 \\
\therefore\ \text{if } c_1 &= 20 \land n_0 = 1,\ \text{then } f \in O(n^3). \\
\because\ f(n) &= 3n^3 + 7n + 10 \\
&\geq 3n^3, n_0 = 1 \\
\therefore\ \text{if } c_2 &= 3 \land n_0 = 1,\ \text{then } f \in \Omega(n^3). \\
\because\ \text{if } f &\in O(n^3) \land f \in \Omega(n^3),\ \text{then } f \in \Theta(n^3) \\
\therefore\ \text{if } c_1 &= 20 \land c_2 = 3 \land n_0 = 1,\ \text{then } f \in \Theta(n^3).
\end{aligned}$$

■

(d) Show, for a function f, where $f(n) = 1^2 + 2^2 + 3^2 + \cdots + n^2$, $f \in O(n^3)$:

$$\begin{aligned}
\because\ f(n) &= 1^2 + 2^2 + 3^2 + \cdots + n^2 \\
&= \textstyle\sum_{i=1}^{n} i^2 \\
&= \frac{n(n+1)(2n+1)}{6} \\
&= \frac{1n + 2n^3 + 3n^2}{6} \\
&\leq \frac{(1+2+3)n^3}{6}, n_0 = 1 \\
&= n^3 \\
\therefore\ \text{if } c &= 1 \land n_0 = 1,\ \text{then } f \in O(n^3).
\end{aligned}$$

■

3. **Problem 3:** Find the Big O estimate or worst case complexity of the following functions (given in pseudo-code). For the following code snippets, the work for the problem is contained in code comments. The function algorithms ($T(n)$) are measured in units of assignment operations.

   (a) The Big O estimate of the below function is $O(N^3)$.

```
int fun (int N) {                           // T(N) =
  for (i = 0; i < N; i++) {                 // N × (
    if (x < 10) {
      for (j = 0; j < N * N; j++) {  // (N^2 ×
        a = a + 10;                         // 1)+
      } // for (level 3)
    } // if (2)
    else if (y < 10) {
      for (p = 0; p <= N * N; p++) {// ((N^2 + 1)×
        b = b + 10;                         // 1)+
      } // for (level 3)
    } // elif (level 2)
    else {
      for (k = 0; k <= 10; k++) {           // (11×
        c = c + 10;                         // 1))
      } // for (level 3)
    } // else (level 2)
  } // for (level 1)
    // = 2N^3 + 12N assignment operations
} // fun (level 0)                          // ∴ T ∈ O(N^3)
```

(b) The Big O estimate of the below function is $O(N^2)$.

```
int fun (int N) {                           // T(N) =
  for (i = N; i > 0; i--) {                 // N × (
    for (j = 0; j < i; j++) {               // i × (
      a = a + 10;                           // 1))
    } // for (2)
  } // for (1)
    // = 1/2 N(N − 1) assignment operations
} // fun (0)                                // ∴ T ∈ O(N^2)
```

(c) The Big O estimate of the below function is $O((\log(N))^2)$.

```
int fun (int N) {                                   // T(N) =
  for (i = N; i > 0; i = i / 2) {                   // (log(N) + 1) × (
    j = 1;                                          // 1+
    while (j < N) {                                 // log(N) × (
      a = a + 10;                                   // 1+
      j = 2 * j;                                    // 1))
    } // while (2)
  } // for (1)
    // = 2 log(N)^2 + 3 log(N) + 1 assignment operations
} // fun (0)
  // ∴ T ∈ O((log(N))^2)
```

(d) The Big O estimate of the below function is $O(N(\log(N))^2)$.

```
int fun (int N) {                                   // T(N) =
  i = N;                                            // 1+
  while (i > 0) {                                   // (log(N) + 1) × (
    j = 1;                                          // 1+
    while (j < N) {                                 // log(N) × (
      k = 0;                                        // 1+
      while (k < N) {                               // 1/2 N × (
        k = k + 2;                                  // 1)+
      } // while (3)
      j = j * 2;                                    // 1)+
    } // while (2)
    i = i / 2;                                      // 1)
  } // while (1)
    // = 1/2 N log(N)^2 + 2 log(N)^2 + 4 log(N) + 1/2 N log(N) + 3
    ↪ assignment operations
} // fun (0)
  // ∴ T ∈ O(N(log(N))^2)
```