CS-217 Homework 2

HONGLI YE

September 24th, 2024

Contents

1 Question 2.1

Question text: Express the following running times in Θ-notation. Justify your answer by referring to the definition of Θ (i. e. work out suitable c_1, c_2, n_0).
a) $3n^2 + 5n - 2$
b) 42
c) $4n^2(1 + logn) - 2n^2$

Answer:

- For **question a**: It is obvious to know that the goal is a quadratic function.

$$3n^2 \leq 3n^2 + 5n - 2 \leq 4n^2 \text{ for } n \geq n_0 = 100$$

$$3 \leq 3 + \frac{5}{n} - \frac{2}{n^2} \leq 4 \text{ for } n \geq n_0 = 100$$

so we take $c_1 = 3$; $c_2 = 4$; $n_0 = 100$;

$$3n^2 + 5n - 2 = \Theta(n^2)$$

- For **question b**: It is obvious to know that the goal is a constant function.

$$41 * 1 \leq 42 \leq 43 * 1 \text{ for } n \geq n_0 = 1$$

so we take $c_1 = 41$; $c_2 = 43$; $n_0 = 1$;

$$42 = \Theta(1)$$

- For **question c**: Observing the goal function's structure is not obvious.

$$4n^2 * (1 + \log n)\text{-}2n^2 = 2n^2 + 4n^2 * \log n$$

Since:

$$2n^2 = O(n^2 \log n)$$

So we take $c_1 = 4$, $c_2 = 6$:

$$4n^2 * \log n \leq 2n^2 + 4n^2 * \log n \leq 6n^2 * \log n$$

$$4 \leq \frac{2}{\log n} + 4 \leq 6$$

Obviously, for $n \geq n_0 = 100$, it holds.
So we take $c_1 = 4$; $c_2 = 6$; $n_0 = 100$;

$$4n^2(1 + logn) - 2n^2 = \Theta(n^2 \log n)$$

2 Question 2.2

Indicate for each pair of functions f(n), g(n) in the following table whether f(n) is O, o, Ω, ω, or Θ of g(n) by writing “yes” or “no” in each box.

Answer:

f(n)	g(n)	O	o	Ω	ω	Θ
$\log n$	$\sqrt{n}$	Yes	Yes	No	No	No
n	$\sqrt{n}$	No	No	Yes	Yes	No
n	n$\log n$	Yes	Yes	No	No	No
n^2	$n^2 + (\log n)^3$	Yes	No	Yes	No	Yes
2^n	n^3	No	No	Yes	Yes	No
$2^{n/2}$	2^n	Yes	Yes	No	No	No
$log_2 n$	$log_{10} n$	Yes	No	Yes	No	Yes

Table 1: Question 2.2

3 Question 2.3

State the number of “foo” operations for each of the following algorithms in Θ-notation. Pay attention to indentation and how long loops are run for. Justify your answer by stating constants c_1, c_2, $n_0 > 0$ from the definition of Θ(g(n)) in your answer.

Answer:

- For **Algorithm a**:
  For each (1,j), the algorithm does the ”foo” operations thrice. So:

  $$g(n) = 3 * \text{all } (i, j) \text{ pairs}$$

  Since:

  $$1 \leq i \leq n \text{ and } 1 \leq j \leq n - 2$$

  So:

  $$\text{all } (i, j) \text{ pairs} = n * (n - 2)$$

So:

$$g(n) = 3n(n-2) + 1$$

Obviously, g(n) is a quadratic function.
Take $c_1 = 2$, $c_2 = 3$, $n_0 = 100$,then:

$$2n^2 \leq 3n(n-2) + 1 \leq 3n^2, \forall n \geq n_0$$

$$g(n) = \Theta(n^2)$$

- For **Algorithm b**:
  There are two loops, without loss of generosity, we name them L_1 and L_2 in sequence.
  In L_1: For each i, the algorithm only does one "foo" operation. So the number of "foo" operations in L_1 = n.
  In L_2: For each i, the algorithm does two "foo" operations. So the number of "foo" operations in L_1 = [n/2]*2.
  Above all: $g(n) = n + [n/2] * 2 + 1$
  Since:

$$n/2 - 1 \leq [n/2] \leq n/2$$

  So:

$$2n - 1 \leq g(n) \leq 2n + 1$$

  Take $c_1 = 1$, $c_2 = 3$ and $n_0 = 100$:

$$c_1 n \leq g(n) \leq c_2 n \ \forall n \geq n_0$$

  So:

$$g(n) = \Theta(n)$$

- For **Algorithm c**:
  For Line 1 and Line 6, the "foo" operation only carries out once.
  For line 4, the algorithm does the "foo" operations once for each (i,j) pair.
  For line 5, the algorithm does the "foo" operations once for each i.
  So:

$$g(n) = 1 + \Sigma_{i=1}^{n}(i) + n + 1$$

$$g(n) = \frac{1}{2}n^2 + \frac{3}{2}n + 2$$

  It is obvious that g(n) is a quadratic function. Take $c_1 = \frac{1}{2}$, $c_2 = 2$ and $n_0 = 100$:

$$\frac{1}{2}n^2 \leq g(n) \leq 2n^2 \ \forall n \geq n_0$$

  So:

$$g(n) = \Theta(n^2)$$

4 Question 2.4

Recall from Lecture 2 that a statement like 2n2 + Θ(n) = Θ(n2) is true if no matter how the anonymous functions are chosen on the left of the equal sign, there is a way to choose the anonymous functions on the right of the equal sign to make the equation valid. You might want to think of the Θ(n) on the left-hand side being a placeholder for some (anonymous) function that grows as fast as n. For each of the following statements, state whether it is true or false. Justify your answers.
1. O($\sqrt{n}$) $= O(n)$
2.$n + o$(n^2)=ω(n)
3. 3nlog $n + O(n)$ =Θ(nlog n)

Answer:

- For **Question 2.4.1**:

$$\forall f \in O(\sqrt{n}),\ \text{we have } c_1, n_0,\ \text{s.t.}, 0 \le f \le c_1\sqrt{n}\ \forall n \ge n_0$$

Take $n_1 > n_0$ and $n_1 > 100$.Then:

$$c_1\sqrt{n} \le c_1 n$$

So:

$$\forall f \in O(\sqrt{n}),\ \text{we have } c_1, n_1,\ \text{s.t.}, 0 \le f \le c_1 n\ \forall n \ge n_1$$

i.e.

$$\forall f \in O(\sqrt{n}), f \in O(n)$$

So:

$$O(\sqrt{n}) = O(n)$$

- For **Question 2.4.2**:
It is false:
Take n = o(n^2)

$$n + n = 2n;\ lim_{n\to\infty}\frac{2n}{n} = 2 \ne \infty$$

So there exist $f \in o(n^2)$, s.t. $n + f \ne \omega(n)$
So $n + o(n^2) \ne \omega(n)$.

- For **Question 2.4.3**:

$$\text{take } f \in O(n), \text{we have } c_1, n_0 \text{ s.t. } 0 \le f \le c_1 n\ \forall n \ge n_0$$

We take n_1 s.t. $n_1 > n_0$ and $n_1 > 100$,then we have:

$$0 \le f \le c_1 n \le c_1 n \log n \ \forall n \ge n_1$$

$$3n \log n \le 3n \log n + f \le (3 + c_1) n \log n \ \forall n \ge n_1$$

So now we can take $c_2 = 3$, $c_3 = 3 + c_1$ and n_1:

$$c_2 n \log n \le 3n \log n + f \le c_3 n \log n \ \forall n \ge n_1$$

So:

$$\forall f \in O(n), \text{we have } 3n \log n + f = \Theta(n \log n)$$

$$3n \log n + O(n) = \Theta(n \log n)$$

5 Question 2.5

The following algorithm computes the product C of two n × n matrices A and B, where A[i, j] corresponds to the element in the i-th row and the j-th column.

Algorithm 1: Matrix-Multiply(A, B)

```
1 for i = 1 to n do
2     for j = 1 to n do
3         C[i, j] := 0;
4         for k = 1 to n do
5             C[i, j] := C[i, j] + A[i, k] · B[k, j];
6 return C;
```

Give the running time of the algorithm (number of operations in a RAM machine) in Θ-notation.Justify your answer. Feel free to use the rules on calculating with Θ-notation from the lecture.

Answer:
Let c_i = the i-th line time cost and n_i = how many time the i-th line operates, g(n) = running time of the algorithm.

$$g(n) = \Sigma_{i=1}^{6}(c_i n_i)$$

$$g(n) = c_1(n+1) + c_2(n+1)n + c_3 n^2 + c_4(n+1)n^2 + c_5 n^3 + c_6$$

$$g(n) = (c_4 + c_5)n^3 + (c_2 + c_3 + c_4)n^2 + (c_1 + c_2)n + (c_1 + c_6)$$

Since $\forall c_i > 0$ and when $n \ge 1$, $n^3 \ge n^2 \ge n \ge 1$

So we take $C_1 = c_4 + c_5$, $C_2 = (c_4 + c_5) + (c_2 + c_3 + c_4) + (c_1 + c_2) + (c_1 + c_6)$ and $n_0 = 1$:

$$C_1 n^3 \leq (c_4 + c_5)n^3 + (c_2 + c_3 + c_4)n^2 + (c_1 + c_2)n + (c_1 + c_6) \leq C_2 n^3 \ \forall n \geq n_0$$

So:

$$g(n) = \Theta(n^3)$$

6 Question 2.6

Already finished it on Online Judge.