Deakin University

Data Structures and Algorithms

OnTrack Submission

Algorithm complexity (pass)

Submitted By:
Peter Stacey
pstacey
2020/08/02 15:52

Tutor:
Maksym Slavnenko

Outcome	**Weight**
Complexity	◆◆◆◆◆
Implement Solutions	◆◇◇◇◇
Document solutions	◆◇◇◇◇

Learning outcome 1 involves evaluating the computational complexity of different algorithms and being able to understand performance in terms of asymptotic notiation. This task aligns very well with the learning outcome, only missing consideration of the memory use of the algorithms. It does not relate to creating data structures (ULO2) or documenting designs and constraints (ULO3)

August 2, 2020

Task 2.1

Student Name: Peter Stacey

Student ID: 219011171

Question 1

1 i.

a)

Case	Assumptions	Operations
Best	• rand() always returns a result very close to 1.0	11
Average	• rand() return averages to 0.5	13
Worst	• rand() always returns a result very close to 0.0	14

b)

Case	Big-Θ Notation
Best	Θ(1)
Average	Θ(1)
Worst	Θ(1)

c) Overall Performance in Big-O: O(1)

d) Overall Performance in Big-Ω: Ω(1)

e) Overall Performance in Big-Θ: Θ(1)

f) *f(n) ~ g(1), O(1), Ω(1), Θ(1), o(logn)*

1 ii.

a)

Case	Assumptions	Operations
Best	• N > 0 • rand() always returns a value >= 0.5	4N + 3
Average	• N > 0 • rand() results average to 0.5	4.5N + 3
Worst	• N > 0 • rand() always returns a value < 0.5	5N + 3

b)

Case	Big-Θ Notation
Best	Θ(n)
Average	Θ(n)
Worst	Θ(n)

c) Overall Performance in Big-O: O(n)

d) Overall Performance in Big-Ω: Ω(n)

e) Overall Performance in Big-Θ: Θ(n)

f) *f(n) ~ Θ(n), O(n), Ω(n), o(nlogn), ω(logn)*

1 iii.

a)

Case	Assumptions	Operations
Best	• N is > 0 • Unlucky is always false	3N + 3
Average	• N > 0 • unlucky is true half of the time	$0.75N^2 + 4.5N + 3$
Worst	• N > 0 • unlucky is always true	$1.5N^2 + 6.5N + 3$

b)

Case	Big-Θ Notation
Best	Θ(n)
Average	$\Theta(n^2)$
Worst	$\Theta(n^2)$

c) Overall Performance in Big-O: $O(n^2)$

d) Overall Performance in Big-Ω: Ω(n)

e) Overall Performance in Big-Θ: Cannot be described

f) *$f(n) = O(n^2), \Omega(n), o(n^3), \omega(\log n)$*

1 iv.

a)

Case	Assumptions	Operations
Best	• unlucky is false	3
Average	• N > 0 • unlucky is true half the time	$1.5\log_2(N) + 5$
Worst	• N > 0 • unlucky is always true	$3\log_2(N) + 7$

b)

Case	Big-Θ Notation
Best	Θ(1)
Average	Θ(logn)
Worst	Θ(logn)

c) Overall Performance in Big-O: O(logn)

d) Overall Performance in Big-Ω: Ω(1)

e) Overall Performance in Big-Θ: Cannot be described

f) *f(n) = O(logn), Ω(1)*

1 v.

a)

Case	Assumptions	Operations
Best	• N > 0 • rand() always return < 0.5	4N + 6
Average	• N > 0 • Half the time, rand() returns a value < 0.5	6N + 6
Worst	• N > 0 • rand() always returns a value < 0.5	8N + 6

b)

Case	Big-Θ Notation
Best	Θ(n)
Average	Θ(n)
Worst	Θ(n)

c) Overall Performance in Big-O: O(n)

d) Overall Performance in Big-Ω: Ω(n)

e) Overall Performance in Big-Θ: Θ(n)

f) *f(n) ~ Θ(n), O(n), Ω(n), o(nlogn), ω(logn)*

1 vi.

a)

Case	Assumptions	Operations
Best	• N > 1 • a[j] < a[j + 1] always (i.e. already sorted ascending)	$1.5N^2 + 2.5N - 2$
Average	• N > 1 • a[j] > a[j + 1] half of the time	$1.75N^2 + 2.25N - 2$
Worst	• N > 1 • a[j] > a[j + 1] always (i.e. maximum unsorted)	$2N^2 + 2N - 2$

b)

Case	Big-Θ Notation
Best	$\Theta(n^2)$
Average	$\Theta(n^2)$
Worst	$\Theta(n^2)$

c) Overall Performance in Big-O: $O(n^2)$

d) Overall Performance in Big-Ω: $\Omega(n^2)$

e) Overall Performance in Big-Θ: $\Theta(n^2)$

f) *$f(n) = \Theta(n^2)$, $O(n^2)$, $\Omega(n^2)$, $o(n^3)$, $\omega(n\log n)$*

Question 2

Big-O notation bounds a function on the upper end, providing an expression for which the expression grows no faster than even under the worst conditions.

Since we are often interested in the impact on performance for large data sets and hence large set of inputs, understanding how the function will perform at its worst, generally provides more useful information when designing and architecting software, than other notations.

Question 3

No.

While an algorithm that is bounded by *$\Theta(n^3)$* will always eventually be slower than an algorithm that is bounded by *$\Theta(\log n)$*, there may be some constant $c_1, c_2 > 0$ that affects the performance of one or both algorithms up to some value of n, for which the *$\Theta(n^3)$* algorithm performs faster.

In figure 1, a constant c_1 is affecting the *$\Theta(\log n)$* algorithm (green line) so that at low values of n, it's performance is slower.

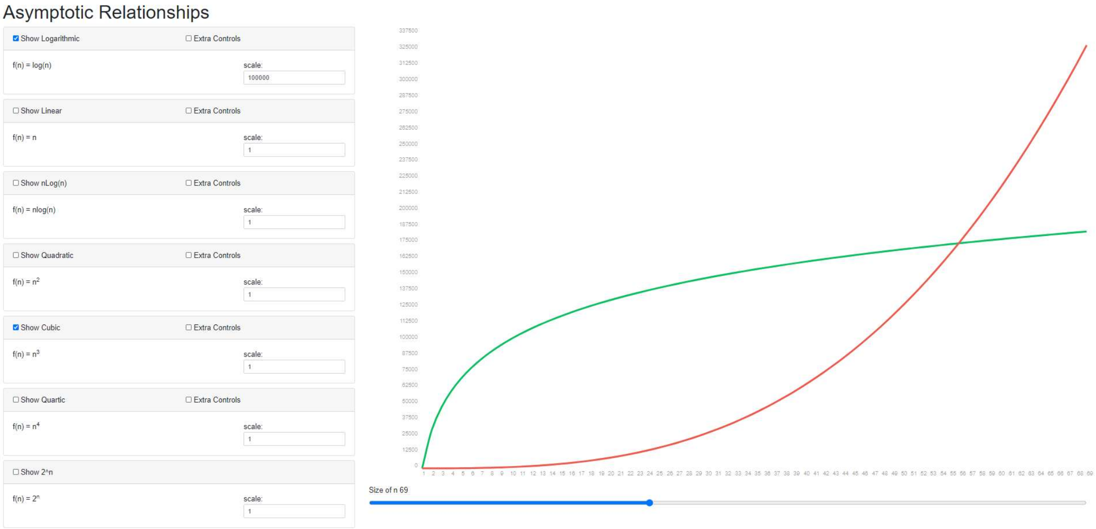

Figure 1: n^3 v log n performance at low n, with constant c_1 affecting the (log n) performance

Source: https://asymptoticnot.z13.web.core.windows.net/ (created by Peter Stacey as part of studying for SIT221)

Question 4

	Statement	Right/Wrong
1	$2n^2 + 6^{13}n = O(n^2)$	Right
2	$n\log n = O(n)$	Wrong
3	$n^3 + n^2 + 10^{16}n = \Theta(n^4)$	Wrong
4	$n\log n = \Omega(n)$	Right