Congratulations! You passed!
TO PASS 80% or higher

Keep Learning

GRADE
100%

Asymptotic Notation and Complexity

LATEST SUBMISSION GRADE
100%

1. Consider the algorithm below shown as pseudo code:

3 / 3 points

```
1   algorithm compute_array_stuff ( a )
2   # a is an array of size n
3   for k = 1 to n :
4       if a[k] is prime:
5           return 0
6   sum = 0
7   for i = 1 to n:
8       for j = i + 1 to n:
9           sum = sum + a[i] * a[j]
10  return sum
```

Select all the correct facts from the list below. Ensure that no incorrect facts are selected.

- [x] For each n, in the best case, the algorithm exits after a constant number of steps. This is realized specifically by an array whose very first element is a prime number.

Correct
Correct! See lines 4-5

- [ ] For each n, the algorithm executes in time that is linear in n in the worst case.
- [x] The worst case is realized by an array of all composite numbers.

Correct
Correct, otherwise, the algorithm exits early.

- [x] The worst case complexity of this algorithm is (upper bounded by) $O(n^3)$.

Correct

- [x] The worst case complexity of this algorithm is (lower bounded) $\Omega(n)$

Correct
Yes since $\Omega(n)$ is asymptotic lower bound to $\Omega(n^2)$.

- [x] The worst case complexity of this algorithm is $\Theta(n^2)$

Correct
Correct: it is upper bounded by some constant times n^2 plus some constant times n. Also, we can also show that on an array of size n with all composites, it will take time at least $c_1 n^2 + c_2 n + c_3$ for constants c_1, c_2, c_3.

2. Algorithm X programmed by an "ace programmer" runs in time $2.5n^4 + 3n^3 + 1.4n + 2$ for inputs of size n in the worst case. Another algorithm Y programmed by a novice for the same problem runs in time $2000n + 10000$ for inputs of size n in the worst case.
Please select the correct answers

3 / 3 points

- [x] For inputs of size 2, X is much faster than Y.

Correct
Correct. Just plug in n = 2 and check in both the formulas.

- [ ] Algorithm X is going to be much faster than Y for all inputs.
- [x] Algorithm X runs in time $\Theta(n^4)$

Correct
Correct: the leading term is n^4 and it will dominate all the other terms in the running time. We can also ignore the constant factors.

- [x] Algorithm Y runs in time $O(n^2)$

✓ **Correct**

Sure since n^2 is asymptotically larger than $2000n + 10000$.

- [x] Algorithm Y is asymptotically faster than X.

✓ **Correct**

At n = 11, the running time of X will be larger than that of Y and for n >= 11, we can see that Y will remain faster (smaller time) than X

- [x] It is possible for some input of size 100 that algorithm X is faster than algorithm Y

✓ **Correct**

Yes, this is true, since the formulae provided talk about the worst case time. Any individual input may run faster.

3. Suppose an algorithm runs in time $1.5 \times 2^n + 1.2 \times n^2$ in the worst case. Select the correct answer from the choices below.

2 / 2 points

- (●) Its running time can be expressed as $O(3^n)$
- ( ) Since constants are ignored in asymptotic running time, we can write the running time as $\Theta(2^{2n})$
- ( ) The factor $1.2 \times n^2$ dominates the other term 2^n. Thus, the algorithms asymptotic complexity is $O(n^2)$.
- ( ) Running times like 2^n are not possible for algorithms we program.

✓ **Correct**

Correct since 3^n will asymptotically dominate 2^n.

4. Suppose an algorithm runs in time $200 \log_2(n) + 250$ time in the worst case as a function of the input size n, which of the options below are correct?

2 / 2 points

- [x] The running time is $\Theta(\log_{10}(n))$ since $\log_{10}(n) = \frac{\log_2(n)}{\log_2(10)}$ is a constant factor times $\log_2(n)$.

✓ **Correct**

Correct: note that $\log_a(n) = \log_b(n) / \log_a(b)$ since the denominator is a constant, we can ignore the base of the logarithm in asymptotic notation

- [ ] It is impossible for an algorithm to have a running time that is smaller than the input size.
- [x] The running time of the algorithm is $O(n)$

✓ **Correct**

Correct: n is asymptotically larger than $\log(n)$.