W1D1

Learn and Repeat

Pattern 1. "for all" and "there exists" pattern

Given an array A, if all elements of A are positive, return 1. Else return 0.

Algorithm 1. (Counting method)

```
count = 0;
for (i  = 0; i < n; i++)
        if (A[i] > 0) count++;
(count == n)? true : false;
```

Check the condition. If true, increment the counter. If the counter value is n, return true. Else return false.

Algorithm 2. (Test the opposite condition method)

```
for (i  = 0; i < n; i++)
        if (A[i]  <=  0) return false;
return true;
```

Check the "opposite condition". If it's true, return false. Return true outside the loop.

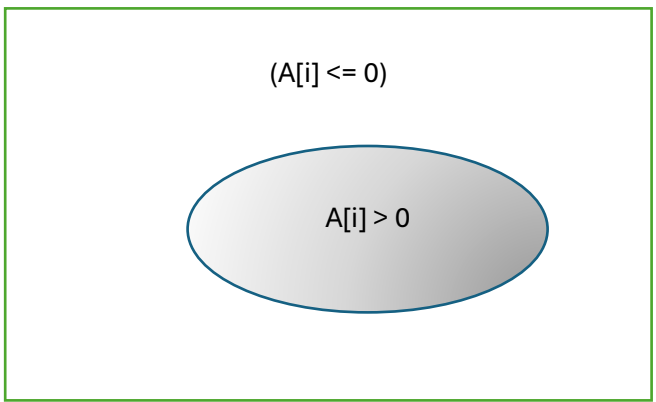
	Best case	Average case	Worst case
Algorithm 1	O(n)	O(n)	O(n)
Algorithm 2	O(1)	O(n)	O(n)

Lower bound of the problem

What is the minimum information required to conclude the problem statement in the affirmative? (This is a very confusing idea for students with less exposure to logic.)

You need to check all elements of A. Hence, the lower bound of the problem is $\Omega(n)$.

Since the lower bound of the problem is equal to the **worst-case time complexity** in Solution 1 (Solution 2), Solution 1 (Solution 2) is optimal.



Exercise 1

An array is defined to be a **235 array** if the number of elements divisible by 2 plus the number of elements divisible by 3 plus the number of elements divisible by 5 plus the number of elements not divisible by 2, 3, or 5 is equal to the number of elements of the array.

Write a method named is123Array that returns 1 if its array argument is a 235 array, otherwise it returns 0.

Note: A number can be divisible by more than one number. For example, 10 is divisible by both 2 and 5.

Algorithm 3. (Counting method)

```
is235Array(A):
  n = length(A)
  c2 = c3 = c5 = cN = 0
  for x in A:
    if x % 2 == 0: c2++
    if x % 3 == 0: c3++
    if x % 5 == 0: c5++
    if (x % 2 != 0 && x % 3 != 0 && x % 5 != 0): cN++
  return 1 if (c2 + c3 + c5 + cN == n) else 0
```

Algorithm 4. (Test the opposite condition method)

```
is235Array_fast(A):
  for x in A:
    divCount = 0
    if x % 2 == 0: divCount++
    if x % 3 == 0: divCount++
    if x % 5 == 0: divCount++
    if divCount >= 2: return 0      // violates the rule
  return 1                         // no violations found
```

Fill in the cells.

	Best case	Average case	Worst case
Algorithm 1	O(n)	O(n)	O(n)
Algorithm 2	O(1)	O(n)	O(n)

What is the lower bound of the problem?

To be certain the array satisfies the property, in the worst case, we must inspect all elements (we might find no violators until the last element). So, the problem's lower bound is $\Omega(n)$; our O(n) worst-case matches that lower bound → optimal.